Package 'dash'

November 7, 2021

Title An Interface to the Dash Ecosystem for Authoring Reactive Web Applications

Version 0.9.3

Description A framework for building analytical web applications, Dash offers a pleasant and productive development experience. No JavaScript required.

Depends R (>= 3.0.2)

Imports R6, fiery (> 1.0.0), routr (> 0.2.0), plotly, reqres (>= 0.2.3), jsonlite, htmltools, assertthat, digest, base64enc, mime, crayon, brotli, glue, magrittr, methods, rlang, utils

Suggests testthat (>= 3.0.0), rstudioapi

License MIT + file LICENSE

Encoding UTF-8

KeepSource true

RoxygenNote 7.1.1

URL https://github.com/plotly/dashR

BugReports https://github.com/plotly/dashR/issues

NeedsCompilation no

Author Chris Parmer [aut],
Ryan Patrick Kyle [aut] (<https://orcid.org/0000-0001-5829-9867>),
Carson Sievert [aut] (<https://orcid.org/0000-0002-4958-2844>),
Hammad Khan [aut, cre] (<https://orcid.org/0000-0003-2479-9841>),
Plotly Technologies [cph]

Maintainer Hammad Khan <hammadkhan@plotly.com>

Repository CRAN

Date/Publication 2021-11-07 08:00:05 UTC

R topics documented:

dash-package *An Interface to the Dash Ecosystem for Authoring Reactive Web Applications*

Description

Dash is a productive framework for building web applications in R, Python, and Julia.

Written on top of Fiery, Plotly.js, and React.js, Dash for R is ideal for building data visualization apps with highly custom user interfaces in pure R. It's particularly suited for anyone who works with data in R.

Through a couple of simple patterns, Dash abstracts away all of the technologies and protocols that are required to build an interactive web-based application. Dash is simple enough that you can bind a user interface around your R code in an afternoon.

Dash apps are rendered in the web browser. You can deploy your apps to servers and then share them through URLs. Since Dash apps are viewed in the web browser, Dash is inherently cross-platform and mobile ready.

There is a lot behind the framework. To learn more about how it is built and what motivated Dash, watch our talk from Plotcon or read our announcement letter.

Dash is an open source package, released under the permissive MIT license. Plotly develops Dash and offers a platform for easily deploying Dash apps in an enterprise environment. If you're interested, please get in touch.

Author(s)

Maintainer: Hammad Khan <hammadkhan@plotly.com>

Authors:

- Chris Parmer <chris@plotly.com>
- Ryan Patrick Kyle <ryan@plotly.com>
- Carson Sievert
- Hammad Khan <hammadkhan@plotly.com>

Other contributors:

- Plotly Technologies [copyright holder]

See Also

Useful links:

- https://dash.plotly.com/r/
- https://github.com/plotly/dashR
- Report bugs at https://github.com/plotly/dashR/issues

add_callback *Add a callback to a Dash app*

Description

Add a callback to a Dash app

Usage

```
add_callback(app, outputs, params, callback)
```

Arguments

app	A dash application created with dash_app().
outputs	Unnamed list. The output argument provides the component id and property which will be updated by the callback; a callback can target one or more outputs (i.e. multiple outputs).
params	Unnamed list; provides input and state statements, each with its own defined id and property. For pattern-matching callbacks, the id field of a component is written in JSON-like syntax and provides fields that are arbitrary keys which describe the targets of the callback. See selectors for more details.
callback	Function; must return output provided input or state arguments. callback may be any valid R function, or a character string containing valid JavaScript, or a call to clientsideFunction, including namespace and function_name arguments for a locally served JavaScript function.

add_meta *Add <meta> tags to a Dash app*

Description

Add <meta> tags to a Dash app

Usage

```
add_meta(app, meta)
```

Arguments

app	A dash application created with dash_app().
meta	A single meta tag or a list of meta tags. Each meta tag is a named list with two elements representing a meta tag. See examples below.

Examples

```
app <- dash_app()

# Add a single meta tag
app %>% add_meta(list(name = "description", content = "My App"))

# Add multiple meta tags
app %>% add_meta(list(
  list(name = "keywords", content = "dash, analysis, graphs"),
  list(name = "viewport", content = "width=device-width, initial-scale=1.0")
))
```

`add_script` *Add external (JavaScript) scripts to a Dash app*

Description

Add external (JavaScript) scripts to a Dash app

Usage

```
add_script(app, script)
```

Arguments

`app`	A dash application created with `dash_app()`
`script`	A single script or a list of scripts. Each script is either a string (the URL), or a named list with `src` (the URL) and any other valid <script> tag attributes. See examples below. Note that this is only used to add **external** scripts, not local.

Examples

```
app <- dash_app()

# Add a single script with URL
app %>% add_script("https://stackpath.bootstrapcdn.com/bootstrap/4.4.0/js/bootstrap.min.js")

# Add multiple scripts with URL
app %>% add_script(list(
  "https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js",
  "https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"
))

# Add a single script with a list
app %>% add_script(
  list(
    href = "https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js",
   integrity = "sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
  )
```

```
)

# Add multiple scripts with both URL and list
app %>% add_script(
  list(
    "https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js",
    "https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js",
    list(
      href = "https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js",
    integrity = "sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
    )
  )
)
```

`add_stylesheet` *Add external (CSS) stylesheets to a Dash app*

Description

Add external (CSS) stylesheets to a Dash app

Usage

```
add_stylesheet(app, stylesheet)
```

Arguments

`app`	A dash application created with `dash_app()`.
`stylesheet`	A single stylesheet or a list of stylesheets. Each stylesheet is either a string (the URL), or a named list with `href` (the URL) and any other valid <link> tag attributes. See examples below. Note that this is only used to add **external** stylesheets, not local.

Examples

```
app <- dash_app()

# Add a single stylesheet with URL
app %>% add_stylesheet("https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css")

# Add multiple stylesheets with URL
app %>% add_stylesheet(list(
  "https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css",
  "https://code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css"
))

# Add a single stylesheet with a list
app %>% add_stylesheet(
  list(
```

```
    href = "https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css",
   integrity = "sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x"
  )
)

# Add multiple stylesheets with both URL and list
app %>% add_stylesheet(
  list(
    "https://code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css",
    "https://fonts.googleapis.com/css?family=Lora",
    list(
      href = "https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css",
    integrity = "sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x"
    )
  )
)
```

`callback_context` *In addition to event properties like n_clicks that change whenever an event happens there is a global variable dash$callback_context, available only inside a callback. It has properties:*

Description

`triggered`: list of changed properties. This will be empty on initial load, unless an input prop got its value from another initial callback. After a user action it is a length-1 list, unless two properties of a single component update simultaneously, such as a value and a timestamp or event counter.

Usage

```
callback_context()
```

Details

`inputs` and `states`: allow you to access the callback params by id and prop instead of through the function arguments.

Examples

```
if (interactive()) {
    dash_app() %>%
      set_layout(
        button('Button 1', id='btn1'),
        button('Button 2', id='btn2'),
        button('Button 3', id='btn3'),
        div(id='container')
      ) %>%
      add_callback(
        output("container", "children"),
```

```
    list(
      input("btn1", "n_clicks"),
      input("btn2", "n_clicks"),
      input("btn3", "n_clicks")
    ),
    function(btn1, btn2, btn3) {
      ctx <- callback_context()
      prevent_update(is.null(ctx))
      sprintf("Triggered: %s, btn1: %s, btn2: %s, btn3: %s",
              ctx$triggered$prop_id, btn1, btn2, btn3)
    }
  ) %>%
  run_app()
}
```

`clientsideFunction` *Define a clientside callback*

Description

Create a callback that updates the output by calling a clientside (JavaScript) function instead of an R function. Note that it is also possible to specify JavaScript as a character string instead of passing `clientsideFunction`. In this case Dash will inline your JavaScript automatically, without needing to save a script inside `assets`.

Usage

```
clientsideFunction(namespace, function_name)
```

Arguments

`namespace`	Character. Describes where the JavaScript function resides (Dash will look for the function at `window[namespace][function_name]`.)
`function_name`	Character. Provides the name of the JavaScript function to call.

Details

With this signature, Dash's front-end will call `window.my_clientside_library.my_function` with the current values of the `value` properties of the components `my-input` and `another-input` whenever those values change. Include a JavaScript file by including it your assets/ folder. The file can be named anything but you'll need to assign the function's namespace to the `window`. For example, this file might look like:

```
window.my_clientside_library = {
my_function: function(input_value_1, input_value_2) {
   return (
     parseFloat(input_value_1, 10) +
       parseFloat(input_value_2, 10)
```

```
    );
}
}
```

Examples

```
## Not run:
app$callback(
  output('output-clientside', 'children'),
  params=list(input('input', 'value')),
  clientsideFunction(
  namespace = 'my_clientside_library',
  function_name = 'my_function'
  )
)

# Passing JavaScript as a character string
app$callback(
 output('output-clientside', 'children'),
 params=list(input('input', 'value')),
 "function (value) {
       return 'Client says \"' + value + '\"';
 }"
)
## End(Not run)
```

`Dash` *R6 class representing a Dash application*

Description

A framework for building analytical web applications, Dash offers a pleasant and productive development experience. No JavaScript required.

Format

An R6::R6Class generator object

Public fields

`server` A cloned (and modified) version of the fiery::Fire object provided to the `server` argument (various routes will be added which enable Dash functionality).

`config` A list of configuration options passed along to dash-renderer. Users shouldn't need to alter any of these options unless they are constructing their own authorization front-end or otherwise need to know where the application is making API calls.

Methods

Public methods:

- Dash$new()
- Dash$server_route()
- Dash$redirect()
- Dash$layout_get()
- Dash$layout()
- Dash$react_version_set()
- Dash$callback()
- Dash$callback_context()
- Dash$callback_context.record_timing()
- Dash$get_asset_url()
- Dash$get_relative_path()
- Dash$strip_relative_path()
- Dash$index_string()
- Dash$interpolate_index()
- Dash$title()
- Dash$run_server()
- Dash$clone()

Method `new()`**:** Create and configure a Dash application.

Usage:

```
Dash$new(
  server = fiery::Fire$new(),
  assets_folder = "assets",
  assets_url_path = "/assets",
  eager_loading = FALSE,
  assets_ignore = "",
  serve_locally = TRUE,
  meta_tags = NULL,
  url_base_pathname = "/",
  routes_pathname_prefix = NULL,
  requests_pathname_prefix = NULL,
  external_scripts = NULL,
  external_stylesheets = NULL,
  compress = TRUE,
  suppress_callback_exceptions = FALSE,
  show_undo_redo = FALSE,
  update_title = "Updating..."
)
```

Arguments:

`server` fiery::Fire object. The web server used to power the application.

`assets_folder` Character. A path, relative to the current working directory, for extra files to be used in the browser. All .js and .css files will be loaded immediately unless excluded by `assets_ignore`, and other files such as images will be served if requested. Default is `assets`.

`assets_url_path` Character. Specify the URL path for asset serving. Default is `assets`.

`eager_loading` Logical. Controls whether asynchronous resources are prefetched (if `TRUE`) or loaded on-demand (if `FALSE`).

`assets_ignore` Character. A regular expression, to match assets to omit from immediate loading. Ignored files will still be served if specifically requested. You cannot use this to prevent access to sensitive files.

`serve_locally` Logical. Whether to serve HTML dependencies locally or remotely (via URL).

`meta_tags` List of lists. HTML <meta> tags to be added to the index page. Each list element should have the attributes and values for one tag, eg: `list(name = 'description',content = 'My App')`.

`url_base_pathname` Character. A local URL prefix to use app-wide. Default is `/`. Both `requests_pathname_prefix` and `routes_pathname_prefix` default to `url_base_pathname`. Environment variable is `DASH_URL_BASE_PATHNAME`.

`routes_pathname_prefix` Character. A prefix applied to the backend routes. Environment variable is `DASH_ROUTES_PATHNAME_PREFIX`.

`requests_pathname_prefix` Character. A prefix applied to request endpoints made by Dash's front-end. Environment variable is `DASH_REQUESTS_PATHNAME_PREFIX`.

`external_scripts` List. An optional list of valid URLs from which to serve JavaScript source for rendered pages. Each entry can be a string (the URL) or a named list with `src` (the URL) and optionally other <script> tag attributes such as `integrity` and `crossorigin`.

`external_stylesheets` List. An optional list of valid URLs from which to serve CSS for rendered pages. Each entry can be a string (the URL) or a list with `href` (the URL) and optionally other <link> tag attributes such as `rel`, `integrity` and `crossorigin`.

`compress` Logical. Whether to try to compress files and data served by Fiery. By default, `brotli` is attempted first, then `gzip`, then the `deflate` algorithm, before falling back to `identity`.

`suppress_callback_exceptions` Logical. Whether to relay warnings about possible layout mis-specifications when registering a callback.

`show_undo_redo` Logical. Set to `TRUE` to enable undo and redo buttons for stepping through the history of the app state.

`update_title` Character. Defaults to `Updating...`; configures the document.title (the text that appears in a browser tab) text when a callback is being run. Set to NULL or '' if you don't want the document.title to change or if you want to control the document.title through a separate component or clientside callback.

Method `server_route()`**:** Connect a URL to a custom server route

Usage:

```
Dash$server_route(path = NULL, handler = NULL, methods = "get")
```

Arguments:

`path` Character. Represents a URL path comprised of strings, parameters (strings prefixed with :), and wildcards (*), separated by /. Wildcards can be used to match any path element, rather than restricting (as by default) to a single path element. For example, it is possible to catch requests to multiple subpaths using a wildcard. For more information, see Route.

`handler` Function. Adds a handler function to the specified method and path. For more information, see Route.

`methods` Character. A string indicating the request method (in lower case, e.g. 'get', 'put', etc.), as used by `reqres`. The default is `get`. For more information, see Route.

Details: `fiery`, the underlying web service framework upon which Dash for R is based, supports custom routing through plugins. While convenient, the plugin API providing this functionality is different from that provided by Flask, as used by Dash for Python. This method wraps the pluggable routing of `routr` routes in a manner that should feel slightly more idiomatic to Dash users.

Querying User-Defined Routes::
It is possible to retrieve the list of user-defined routes by invoking the `get_data` method. For example, if your Dash application object is app, use `app$server$get_data("user-routes")`. If you wish to erase all user-defined routes without instantiating a new Dash application object, one option is to clear the routes manually: `app$server$set_data("user-routes",list())`.

Examples:

```
library(dash)
app <- Dash$new()

# A handler to redirect requests with `307` status code (temporary redirects);
# for permanent redirects (`301`), see the `redirect` method described below
#
# A simple single path-to-path redirect
app$server_route('/getting-started', function(request, response, keys, ...) {
  response$status <- 307L
  response$set_header('Location', '/layout')
  TRUE
})

# Example of a redirect with a wildcard for subpaths
app$server_route('/getting-started/*', function(request, response, keys, ...) {
  response$status <- 307L
  response$set_header('Location', '/layout')
  TRUE
})

# Example of a parameterized redirect with wildcard for subpaths
app$server_route('/accounts/:user_id/*', function(request, response, keys, ...) {
  response$status <- 307L
  response$set_header('Location', paste0('/users/', keys$user_id))
  TRUE
})
```

Method `redirect()`**:** Redirect a Dash application URL path

Usage:

```
Dash$redirect(old_path = NULL, new_path = NULL, methods = "get")
```

Arguments:

`old_path` Character. Represents the URL path to redirect, comprised of strings, parameters (strings prefixed with :), and wildcards (*), separated by /. Wildcards can be used to match

any path element, rather than restricting (as by default) to a single path element. For example, it is possible to catch requests to multiple subpaths using a wildcard. For more information, see Route.

`new_path` Character or function. Same as `old_path`, but represents the new path which the client should load instead. If a function is provided instead of a string, it should have `keys` within its formals.

`methods` Character. A string indicating the request method (in lower case, e.g. 'get', 'put', etc.), as used by `reqres`. The default is `get`. For more information, see Route.

Details: This is a convenience method to simplify adding redirects for your Dash application which automatically return a `301` HTTP status code and direct the client to load an alternate URL.

Examples:

```
library(dash)
app <- Dash$new()

# example of a simple single path-to-path redirect
app$redirect("/getting-started", "/layout")

# example of a redirect using wildcards
app$redirect("/getting-started/*", "/layout/*")

# example of a parameterized redirect using a function for new_path,
# which requires passing in keys to take advantage of subpaths within
# old_path that are preceded by a colon (e.g. :user_id):
app$redirect("/accounts/:user_id/*", function(keys) paste0("/users/", keys$user_id))
```

Method `layout_get()`**:** Retrieves the Dash application layout.

Usage:

```
Dash$layout_get(render = TRUE)
```

Arguments:

`render` Logical. If the layout is a function, should the function be executed to return the layout? If `FALSE`, the function is returned as-is.

Details: If render is `TRUE`, and the layout is a function, the result of the function (rather than the function itself) is returned.

Returns: List or function, depending on the value of `render` (see above). When returning an object of class `dash_component`, the default `print` method for this class will display the corresponding pretty-printed JSON representation of the object to the console.

Method `layout()`**:** Set the Dash application layout (i.e., specify its user interface).

Usage:

```
Dash$layout(value)
```

Arguments:

`value` An object of the `dash_component` class, which provides a component or collection of components, specified either as a Dash component or a function that returns a Dash component.

Details: `value` should be either a collection of Dash components (e.g., dccSlider, htmlDiv, etc) or a function which returns a collection of components. The collection of components must be nested, such that any additional components contained within `value` are passed solely as `children` of the top-level component. In all cases, `value` must be a member of the `dash_component` class.

Method `react_version_set()`**:** Update the version of React in the list of dependencies served by dash-renderer to the client.

Usage:

```
Dash$react_version_set(version)
```

Arguments:

`version` Character. The version number of React to use.

Method `callback()`**:** Define a Dash callback.

Usage:

```
Dash$callback(output, params, func)
```

Arguments:

`output` Named list. The `output` argument provides the component `id` and `property` which will be updated by the callback; a callback can target one or more outputs (i.e. multiple outputs).

`params` Unnamed list; provides input and state statements, each with its own defined `id` and `property`. For pattern-matching callbacks, the `id` field of a component is written in JSON-like syntax and provides fields that are arbitrary keys which describe the targets of the callback. See selectors for more details.

`func` Function; must return output provided input or state arguments. `func` may be any valid R function, or a character string containing valid JavaScript, or a call to clientsideFunction, including `namespace` and `function_name` arguments for a locally served JavaScript function.

Details: Describes a server or clientside callback relating the values of one or more `output` items to one or more `input` items which will trigger the callback when they change, and optionally `state` items which provide additional information but do not trigger the callback directly.
For detailed examples of how to use pattern-matching callbacks, see the entry for selectors or visit our interactive online documentation at https://dash.plotly.com/r/.
The `output` argument defines which layout component property should receive the results (via the output object). The events that trigger the callback are then described by the input (and/or state) object(s) (which should reference layout components), which become argument values for R callback handlers defined in `func`.
Here `func` may either be an anonymous R function, a JavaScript function provided as a character string, or a call to `clientsideFunction()`, which describes a locally served JavaScript function instead. The latter two methods define a "clientside callback", which updates components without passing data to and from the Dash backend. The latter may offer improved performance relative to callbacks written purely in R.

Method `callback_context()`**:** Request and return the calling context of a Dash callback.

Usage:

```
Dash$callback_context()
```

Details: The callback_context method permits retrieving the inputs which triggered the firing of a given callback, and allows introspection of the input/state values given their names. It is only available from within a callback; attempting to use this method outside of a callback will result in a warning.
The callback_context method returns a list containing three elements: states, triggered, inputs. The first and last of these correspond to the values of states and inputs for the current invocation of the callback, and triggered provides a list of changed properties.

Returns: List comprising elements states, triggered, inputs.

Method callback_context.record_timing(): Records timing information for a server resource.

Usage:

```
Dash$callback_context.record_timing(name, duration = NULL, description = NULL)
```

Arguments:

name Character. The name of the resource.

duration Numeric. The time in seconds to report. Internally, this is rounded to the nearest millisecond.

description Character. A description of the resource.

Details: The callback_context.record_timing method permits retrieving the duration required to execute a given callback. It may only be called from within a callback; a warning will be thrown and the method will otherwise return NULL if invoked outside of a callback.

Method get_asset_url(): Return a URL for a Dash asset.

Usage:

```
Dash$get_asset_url(asset_path, prefix = self$config$requests_pathname_prefix)
```

Arguments:

asset_path Character. Specifies asset filename whose URL should be returned.

prefix Character. Specifies pathname prefix; default is to use requests_pathname_prefix.

Details: The get_asset_url method permits retrieval of an asset's URL given its filename. For example, app$get_asset_url('style.css') should return /assets/style.css when assets_folder = 'assets'. By default, the prefix is the value of requests_pathname_prefix, but this is configurable via the prefix parameter. Note: this method will present a warning and return NULL if the Dash app was not loaded via source() if the DASH_APP_PATH environment variable is undefined.

Returns: Character. A string representing the URL to the asset.

Method get_relative_path(): Return relative asset paths for Dash assets.

Usage:

```
Dash$get_relative_path(
  path,
  requests_pathname_prefix = self$config$requests_pathname_prefix
)
```

Arguments:

path Character. A path string prefixed with a leading / which directs at a path or asset directory.

`requests_pathname_prefix` Character. The pathname prefix for the application when deployed. Defaults to the environment variable set by the server, or `""` if run locally.

Details: The `get_relative_path` method simplifies the handling of URLs and pathnames for apps running locally and on a deployment server such as Dash Enterprise. It handles the prefix for requesting assets similar to the `get_asset_url` method, but can also be used for URL handling in components such as `dccLink` or `dccLocation`. For example, `app$get_relative_url("/page/")` would return /app/page/ for an app running on a deployment server. The path must be prefixed with a `/`.

Returns: Character. A string describing a relative path to a Dash app's asset given a `path` and `requests_pathname_prefix`.

Method `strip_relative_path()`**:** Return a Dash asset path without its prefix.

Usage:

```
Dash$strip_relative_path(
  path,
  requests_pathname_prefix = self$config$requests_pathname_prefix
)
```

Arguments:

`path` Character. A path string prefixed with a leading `/` which directs at a path or asset directory.

`requests_pathname_prefix` Character. The pathname prefix for the app on a deployed application. Defaults to the environment variable set by the server, or `""` if run locally.

Details: The `strip_relative_path` method simplifies the handling of URLs and pathnames for apps running locally and on a deployment server such as Dash Enterprise. It acts almost opposite to the `get_relative_path` method, by taking a relative path as an input, and returning the `path` stripped of the `requests_pathname_prefix`, and any leading or trailing `/`. For example, a path string /app/homepage/, would be returned as `homepage`. This is particularly useful for `dccLocation` URL routing.

Method `index_string()`**:** Specify a custom index string for a Dash application.

Usage:

```
Dash$index_string(string)
```

Arguments:

`string` Character; the index string template, with interpolation keys included.

Details: The `index_string` method allows the specification of a custom index by changing the default `HTML` template that is generated by the Dash UI. #' Meta tags, CSS, and JavaScript are some examples of features that can be modified. This method will present a warning if your HTML template is missing any necessary elements and return an error if a valid index is not defined. The following interpolation keys are currently supported:

{%metas%} Optional - The registered meta tags.

{%favicon%} Optional - A favicon link tag if found in assets.

{%css%} Optional - Link tags to CSS resources.

{%config%} Required - Configuration details generated by Dash for the renderer.

{%app_entry%} Required - The container where Dash React components are rendered.

{%scripts%} Required - Script tags for collected dependencies.

Example of a basic HTML index string: `"<!DOCTYPE html>`

```
<html>
 <head>
  {%meta_tags%}
     <title>{{
     {%favicon%}
     {%css_tags%}
 </head>
  <body>
    {%app_entry%}
    <footer>
     {%config%}
     {%scripts%}
    </footer>
  </body>
</html>"
```

Method `interpolate_index()`**:** Modify index template variables for a Dash application.

Usage:

```
Dash$interpolate_index(template_index = private$template_index[[1]], ...)
```

Arguments:

`template_index` Character. A formatted string with the HTML index string. Defaults to the initial template.

`...` Named list. The unnamed arguments can be passed as individual named lists corresponding to the components of the Dash HTML index. These include the same argument as those found in the `index_string()` template.

Details: With the `interpolate_index` method, one can pass a custom index with template string variables that are already evaluated. Directly passing arguments to the `template_index` has the effect of assigning them to variables present in the template. This is similar to the `index_string` method but offers the ability to change the default components of the Dash index as seen in the example below.

Examples:

```
library(dash)
app <- Dash$new()

sample_template <- "<!DOCTYPE html>
<html>
<head>
{%meta_tags%}
<title>Index Template Test</title>
{%favicon%}
{%css_tags%}
</head>
<body>
{%app_entry%}
<footer>
```

```
{%config%}
{%scripts%}
</footer>
</body>
</html>"

# this is the default configuration, but custom configurations
# are possible -- the structure of the "config" argument is
# a list, in which each element is a JSON key/value pair, when
# reformatted as JSON from the list:
#  e.g. {"routes_pathname_prefix":"/", "ui":false}
config <- sprintf("<script id='_dash-config' type='application/json'> %s </script>",
                  jsonlite::toJSON(app$config, auto_unbox=TRUE))

app$interpolate_index(
  sample_template,
  metas = "<meta_charset='UTF-8'/>",
 app_entry = "<div id='react-entry-point'><div class='_dash-loading'>Loading...</div></div>",
  config = config,
  scripts = "")
```

Method `title()`**:** Set the title of the Dash app

Usage:

```
Dash$title(string = "Dash")
```

Arguments:

`string` Character. A string representation of the name of the Dash application.

Details: If no title is supplied, Dash for R will use 'Dash'.

Method `run_server()`**:** Start the Fiery HTTP server and run a Dash application.

Usage:

```
Dash$run_server(
  host = Sys.getenv("HOST", "127.0.0.1"),
  port = Sys.getenv("PORT", 8050),
  block = TRUE,
  showcase = FALSE,
  use_viewer = FALSE,
  dev_tools_prune_errors = TRUE,
  debug = Sys.getenv("DASH_DEBUG"),
  dev_tools_ui = Sys.getenv("DASH_UI"),
  dev_tools_props_check = Sys.getenv("DASH_PROPS_CHECK"),
  dev_tools_hot_reload = Sys.getenv("DASH_HOT_RELOAD"),
  dev_tools_hot_reload_interval = Sys.getenv("DASH_HOT_RELOAD_INTERVAL"),
 dev_tools_hot_reload_watch_interval = Sys.getenv("DASH_HOT_RELOAD_WATCH_INTERVAL)"),
  dev_tools_hot_reload_max_retry = Sys.getenv("DASH_HOT_RELOAD_MAX_RETRY"),
  dev_tools_silence_routes_logging = NULL,
  ...
)
```

Arguments:

`host` Character. A string specifying a valid IPv4 address for the Fiery server, or 0.0.0.0 to listen on all addresses. Default is 127.0.0.1 Environment variable: `HOST`.

`port` Integer. Specifies the port number on which the server should listen (default is `8050`). Environment variable: `PORT`.

`block` Logical. Start the server while blocking console input? Default is `TRUE`.

`showcase` Logical. Load the Dash application into the default web browser when server starts? Default is `FALSE`.

`use_viewer` Logical. Load the Dash application into RStudio's viewer pane? Requires that `host` is either 127.0.0.1 or `localhost`, and that Dash application is started within RStudio; if `use_viewer = TRUE` and these conditions are not satisfied, the user is warned and the app opens in the default browser instead. Default is `FALSE`.

`dev_tools_prune_errors` Logical. Reduce tracebacks such that only lines relevant to user code remain, stripping out Fiery and Dash references? Only available with debugging. `TRUE` by default, set to `FALSE` to see the complete traceback. Environment variable: `DASH_PRUNE_ERRORS`.

`debug` Logical. Enable/disable all the Dash developer tools (and the within-browser user interface for the callback graph visualizer and stack traces) unless overridden by the arguments or environment variables. Default is `FALSE` when called via `run_server`. For more information, please visit `https://dash.plotly.com/r/devtools`. Environment variable: `DASH_DEBUG`.

`dev_tools_ui` Logical. Show Dash's developer tools UI? Default is `TRUE` if `debug == TRUE`, `FALSE` otherwise. Environment variable: `DASH_UI`.

`dev_tools_props_check` Logical. Validate the types and values of Dash component properties? Default is `TRUE` if `debug == TRUE`, `FALSE` otherwise. Environment variable: `DASH_PROPS_CHECK`.

`dev_tools_hot_reload` Logical. Activate hot reloading when app, assets, and component files change? Default is `TRUE` if `debug == TRUE`, `FALSE` otherwise. Requires that the Dash application is loaded using `source()`, so that `srcref` attributes are available for executed code. Environment variable: `DASH_HOT_RELOAD`.

`dev_tools_hot_reload_interval` Numeric. Interval in seconds for the client to request the reload hash. Default is `3`. Environment variable: `DASH_HOT_RELOAD_INTERVAL`.

`dev_tools_hot_reload_watch_interval` Numeric. Interval in seconds for the server to check asset and component folders for changes. Default `0.5`. Environment variable: `DASH_HOT_RELOAD_WATCH_INTERVAL`.

`dev_tools_hot_reload_max_retry` Integer. Maximum number of failed reload hash requests before failing and displaying a pop up. Default `0.5`. Environment variable: `DASH_HOT_RELOAD_MAX_RETRY`.

`dev_tools_silence_routes_logging` Logical. Replace Fiery's default logger with `dashLogger` instead (will remove all routes logging)? Enabled with debugging by default because hot reload hash checks generate a lot of requests.

`...` Additional arguments to pass to the `start` handler; see the fiery documentation for relevant examples.

Details: Starts the Fiery server in local mode and launches the Dash application. If a parameter can be set by an environment variable, that is listed too. Values provided here take precedence over environment variables. . If provided, `host`/`port` set the `host`/`port` fields of the underlying fiery::Fire web server. The `block`/`showcase`/`...` arguments are passed along to the `ignite()` method of the fiery::Fire server.

Examples:

```
if (interactive() ) {
  library(dash)

  app <- Dash$new()
  app$layout(htmlDiv(
    list(
      dccInput(id = "inputID", value = "initial value", type = "text"),
      htmlDiv(id = "outputID")
    )
   )
  )

  app$callback(output = list(id="outputID", property="children"),
               params = list(input(id="inputID", property="value"),
                        state(id="inputID", property="type")),
    function(x, y)
      sprintf("You've entered: '%s' into a '%s' input control", x, y)
  )

  app$run_server(showcase = TRUE)
}
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Dash$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `Dash$server_route`
## ------------------------------------------------

library(dash)
app <- Dash$new()

# A handler to redirect requests with `307` status code (temporary redirects);
# for permanent redirects (`301`), see the `redirect` method described below
#
# A simple single path-to-path redirect
app$server_route('/getting-started', function(request, response, keys, ...) {
  response$status <- 307L
  response$set_header('Location', '/layout')
  TRUE
})

# Example of a redirect with a wildcard for subpaths
app$server_route('/getting-started/*', function(request, response, keys, ...) {
```

```
  response$status <- 307L
  response$set_header('Location', '/layout')
  TRUE
})

# Example of a parameterized redirect with wildcard for subpaths
app$server_route('/accounts/:user_id/*', function(request, response, keys, ...) {
  response$status <- 307L
  response$set_header('Location', paste0('/users/', keys$user_id))
  TRUE
})

## ------------------------------------------------
## Method `Dash$redirect`
## ------------------------------------------------

library(dash)
app <- Dash$new()

# example of a simple single path-to-path redirect
app$redirect("/getting-started", "/layout")

# example of a redirect using wildcards
app$redirect("/getting-started/*", "/layout/*")

# example of a parameterized redirect using a function for new_path,
# which requires passing in keys to take advantage of subpaths within
# old_path that are preceded by a colon (e.g. :user_id):
app$redirect("/accounts/:user_id/*", function(keys) paste0("/users/", keys$user_id))

## ------------------------------------------------
## Method `Dash$interpolate_index`
## ------------------------------------------------

library(dash)
app <- Dash$new()

sample_template <- "<!DOCTYPE html>
<html>
<head>
{%meta_tags%}
<title>Index Template Test</title>
{%favicon%}
{%css_tags%}
</head>
<body>
{%app_entry%}
<footer>
{%config%}
{%scripts%}
</footer>
</body>
</html>"
```

```
# this is the default configuration, but custom configurations
# are possible -- the structure of the "config" argument is
# a list, in which each element is a JSON key/value pair, when
# reformatted as JSON from the list:
#  e.g. {"routes_pathname_prefix":"/", "ui":false}
config <- sprintf("<script id='_dash-config' type='application/json'> %s </script>",
                  jsonlite::toJSON(app$config, auto_unbox=TRUE))

app$interpolate_index(
  sample_template,
  metas = "<meta_charset='UTF-8'/>",
 app_entry = "<div id='react-entry-point'><div class='_dash-loading'>Loading...</div></div>",
  config = config,
  scripts = "")

## ------------------------------------------------
## Method `Dash$run_server`
## ------------------------------------------------

if (interactive() ) {
  library(dash)

  app <- Dash$new()
  app$layout(htmlDiv(
    list(
      dccInput(id = "inputID", value = "initial value", type = "text"),
      htmlDiv(id = "outputID")
    )
   )
  )

  app$callback(output = list(id="outputID", property="children"),
               params = list(input(id="inputID", property="value"),
                         state(id="inputID", property="type")),
    function(x, y)
      sprintf("You've entered: '%s' into a '%s' input control", x, y)
  )

  app$run_server(showcase = TRUE)
}
```

---

dashDataTable *DataTable component*

Description

Dash DataTable is an interactive table component designed for viewing, editing, and exploring large datasets. DataTable is rendered with standard, semantic HTML <table/> markup, which makes it accessible, responsive, and easy to style. This component was written from scratch in React.js

specifically for the Dash community. Its API was designed to be ergonomic and its behavior is completely customizable through its properties.

Usage

```
dashDataTable(id=NULL, data=NULL, columns=NULL, active_cell=NULL,
include_headers_on_copy_paste=NULL, locale_format=NULL,
markdown_options=NULL, css=NULL, data_previous=NULL,
data_timestamp=NULL, editable=NULL, end_cell=NULL,
export_columns=NULL, export_format=NULL,
export_headers=NULL, fill_width=NULL, hidden_columns=NULL,
is_focused=NULL, merge_duplicate_headers=NULL,
fixed_columns=NULL, fixed_rows=NULL, column_selectable=NULL,
row_deletable=NULL, cell_selectable=NULL,
row_selectable=NULL, selected_cells=NULL,
selected_rows=NULL, selected_columns=NULL,
selected_row_ids=NULL, start_cell=NULL,
style_as_list_view=NULL, page_action=NULL,
page_current=NULL, page_count=NULL, page_size=NULL,
dropdown=NULL, dropdown_conditional=NULL,
dropdown_data=NULL, tooltip=NULL, tooltip_conditional=NULL,
tooltip_data=NULL, tooltip_header=NULL, tooltip_delay=NULL,
tooltip_duration=NULL, filter_query=NULL,
filter_action=NULL, filter_options=NULL, sort_action=NULL,
sort_mode=NULL, sort_by=NULL, sort_as_null=NULL,
style_table=NULL, style_cell=NULL, style_data=NULL,
style_filter=NULL, style_header=NULL,
style_cell_conditional=NULL, style_data_conditional=NULL,
style_filter_conditional=NULL,
style_header_conditional=NULL, virtualization=NULL,
derived_filter_query_structure=NULL,
derived_viewport_data=NULL, derived_viewport_indices=NULL,
derived_viewport_row_ids=NULL,
derived_viewport_selected_columns=NULL,
derived_viewport_selected_rows=NULL,
derived_viewport_selected_row_ids=NULL,
derived_virtual_data=NULL, derived_virtual_indices=NULL,
derived_virtual_row_ids=NULL,
derived_virtual_selected_rows=NULL,
derived_virtual_selected_row_ids=NULL, loading_state=NULL,
persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

id	Character. The ID of the table.
data	List of list with named elements and values of type character | numeric | logicals. The contents of the table. The keys of each item in data should match the column IDs. Each item can also have an 'id' key, whose value is its row ID. If

there is a column with ID='id' this will display the row ID, otherwise it is just used to reference the row for selections, filtering, etc. Example: ['column-1': 4.5, 'column-2': 'montreal', 'column-3': 'canada', 'column-1': 8, 'column-2': 'boston', 'column-3': 'america']

`columns` List of lists containing elements 'clearable', 'deletable', 'editable', 'filter_options', 'hideable', 'renamable', 'selectable', 'format', 'id', 'name', 'presentation', 'on_change', 'sort_as_null', 'validation', 'type'. those elements have the following types: - clearable (a value equal to: 'first', 'last' | logical | list of logicals; optional): if true, the user can clear the column by clicking on the 'clear' action button on the column. if there are multiple header rows, true will display the action button on each row. if 'last', the 'clear' action button will only appear on the last header row. if 'first' it will only appear on the first header row. these are respectively shortcut equivalents to '[false, ..., false, true]' and '[true, false, ..., false]'. if there are merged, multi-header columns then you can choose which column header row to display the 'clear' action button in by supplying an array of booleans. for example, '[true, false]' will display the 'clear' action button on the first row, but not the second row. if the 'clear' action button appears on a merged column, then clicking on that button will clear *all* of the merged columns associated with it. unlike 'column.deletable', this action does not remove the column(s) from the table. it only removed the associated entries from 'data'. - deletable (a value equal to: 'first', 'last' | logical | list of logicals; optional): if true, the user can remove the column by clicking on the 'delete' action button on the column. if there are multiple header rows, true will display the action button on each row. if 'last', the 'delete' action button will only appear on the last header row. if 'first' it will only appear on the first header row. these are respectively shortcut equivalents to '[false, ..., false, true]' and '[true, false, ..., false]'. if there are merged, multi-header columns then you can choose which column header row to display the 'delete' action button in by supplying an array of booleans. for example, '[true, false]' will display the 'delete' action button on the first row, but not the second row. if the 'delete' action button appears on a merged column, then clicking on that button will remove *all* of the merged columns associated with it. - editable (logical; optional): there are two 'editable' flags in the table. this is the column-level editable flag and there is also the table-level 'editable' flag. these flags determine whether the contents of the table are editable or not. if the column-level 'editable' flag is set it overrides the table-level 'editable' flag for that column. - filter_options (optional): there are two 'filter_options' props in the table. this is the column-level filter_options prop and there is also the table-level 'filter_options' prop. these props determine whether the applicable filter relational operators will default to 'sensitive' or 'insensitive' comparison. if the column-level 'filter_options' prop is set it overrides the table-level 'filter_options' prop for that column.. filter_options has the following type: lists containing elements 'case'. those elements have the following types: - case (a value equal to: 'sensitive', 'insensitive'; optional) - hideable (a value equal to: 'first', 'last' | logical | list of logicals; optional): if true, the user can hide the column by clicking on the 'hide' action button on the column. if there are multiple header rows, true will display the action button on each row. if 'last', the 'hide' action button will only appear on the last header row. if 'first' it will only appear on the first header row. these are respectively shortcut equivalents to '[false,

..., false, true]‘ and ‘[true, false, ..., false]‘. if there are merged, multi-header columns then you can choose which column header row to display the ‘hide‘ action button in by supplying an array of booleans. for example, ‘[true, false]‘ will display the ‘hide‘ action button on the first row, but not the second row. if the ‘hide‘ action button appears on a merged column, then clicking on that button will hide *all* of the merged columns associated with it. - renamable (a value equal to: ’first’, ’last’ | logical | list of logicals; optional): if true, the user can rename the column by clicking on the ‘rename‘ action button on the column. if there are multiple header rows, true will display the action button on each row. if ‘last‘, the ‘rename‘ action button will only appear on the last header row. if ‘first‘ it will only appear on the first header row. these are respectively shortcut equivalents to ‘[false, ..., false, true]‘ and ‘[true, false, ..., false]‘. if there are merged, multi-header columns then you can choose which column header row to display the ‘rename‘ action button in by supplying an array of booleans. for example, ‘[true, false]‘ will display the ‘rename‘ action button on the first row, but not the second row. if the ‘rename‘ action button appears on a merged column, then clicking on that button will rename *all* of the merged columns associated with it. - selectable (a value equal to: ’first’, ’last’ | logical | list of logicals; optional): if true, the user can select the column by clicking on the checkbox or radio button in the column. if there are multiple header rows, true will display the input on each row. if ‘last‘, the input will only appear on the last header row. if ‘first‘ it will only appear on the first header row. these are respectively shortcut equivalents to ‘[false, ..., false, true]‘ and ‘[true, false, ..., false]‘. if there are merged, multi-header columns then you can choose which column header row to display the input in by supplying an array of booleans. for example, ‘[true, false]‘ will display the ‘selectable‘ input on the first row, but now on the second row. if the ‘selectable‘ input appears on a merged columns, then clicking on that input will select *all* of the merged columns associated with it. the table-level prop ‘column_selectable‘ is used to determine the type of column selection to use. - format (optional): the formatting applied to the column’s data. this prop is derived from the [d3-format](https://github.com/d3/d3-format) library specification. apart from being structured slightly differently (under a single prop), the usage is the same. see also dash_table.formattemplate. it contains helper functions for typical number formats.. format has the following type: lists containing elements ’locale’, ’nully’, ’prefix’, ’specifier’. those elements have the following types: - locale (optional): represents localization specific formatting information. when left unspecified, will use the default value provided by d3-format.. locale has the following type: lists containing elements ’symbol’, ’decimal’, ’group’, ’grouping’, ’numerals’, ’percent’, ’separate_4digits’. those elements have the following types: - symbol (list of characters; optional): (default: [’$’, ”]). a list of two strings representing the prefix and suffix symbols. typically used for currency, and implemented using d3’s currency format, but you can use this for other symbols such as measurement units - decimal (character; optional): (default: ’.’). the string used for the decimal separator - group (character; optional): (default: ’,’). the string used for the groups separator - grouping (list of numerics; optional): (default: [3]). a list of integers representing the grouping pattern. the default is 3 for thousands. - numerals (list of characters; optional): a list of ten strings used as replacements for numbers

0-9 - percent (character; optional): (default: '%'). the string used for the percentage symbol - separate_4digits (logical; optional): (default: true). separates integers with 4-digits or less - nully (logical | numeric | character | named list | unnamed list; optional): a value that will be used in place of the nully value during formatting. if the value type matches the column type, it will be formatted normally. - prefix (numeric; optional): a number representing the si unit to use during formatting. see 'dash_table.format.prefix' enumeration for the list of valid values - specifier (character; optional): (default: ''). represents the d3 rules to apply when formatting the number. - id (character; required): the 'id' of the column. the column 'id' is used to match cells in data with particular columns. the 'id' is not visible in the table. - name (character | list of characters; required): the 'name' of the column, as it appears in the column header. if 'name' is a list of strings, then the columns will render with multiple headers rows. - presentation (a value equal to: 'input', 'dropdown', 'markdown'; optional): the 'presentation' to use to display data. markdown can be used on columns with type 'text'. see 'dropdown' for more info. defaults to 'input' for ['datetime', 'numeric', 'text', 'any']. - on_change (optional): the 'on_change' behavior of the column for user-initiated modifications.. on_change has the following type: lists containing elements 'action', 'failure'. those elements have the following types: - action (a value equal to: 'coerce', 'none', 'validate'; optional): (default 'coerce'): 'none': do not validate data; 'coerce': check if the data corresponds to the destination type and attempts to coerce it into the destination type if not; 'validate': check if the data corresponds to the destination type (no coercion). - failure (a value equal to: 'accept', 'default', 'reject'; optional): (default 'reject'): what to do with the value if the action fails. 'accept': use the invalid value; 'default': replace the provided value with 'validation.default'; 'reject': do not modify the existing value. - sort_as_null (list of character | numeric | logicals; optional): an array of string, number and boolean values that are treated as 'null' (i.e. ignored and always displayed last) when sorting. this value overrides the table-level 'sort_as_null'. - validation (optional): the 'validation' options for user input processing that can accept, reject or apply a default value.. validation has the following type: lists containing elements 'allow_null', 'default', 'allow_yy'. those elements have the following types: - allow_null (logical; optional): allow the use of nully values. (undefined, null, nan) (default: false) - default (logical | numeric | character | named list | unnamed list; optional): the default value to apply with on_change.failure = 'default'. (default: none) - allow_yy (logical; optional): this is for 'datetime' columns only. allow 2-digit years (default: false). if true, we interpret years as ranging from now-70 to now+29 - in 2019 this is 1949 to 2048 but in 2020 it will be different. if used with 'action: 'coerce'', will convert user input to a 4-digit year. - type (a value equal to: 'any', 'numeric', 'text', 'datetime'; optional): the data-type provides support for per column typing and allows for data validation and coercion. 'numeric': represents both floats and ints. 'text': represents a string. 'datetime': a string representing a date or date-time, in the form: 'yyyy-mm-dd hh:mm:ss.ssssss' or some truncation thereof. years must have 4 digits, unless you use 'validation.allow_yy: true'. also accepts 't' or 'T' between date and time, and allows timezone info at the end. to convert these strings to python 'datetime' objects, use 'dateutil.parser.isoparse'. in r use 'parse_iso_8601' from the 'parse-

date‘ library. warning: these parsers do not work with 2-digit years, if you use ‘validation.allow_yy: true‘ and do not coerce to 4-digit years. and parsers that do work with 2-digit years may make a different guess about the century than we make on the front end. 'any': represents any type of data. defaults to 'any' if undefined.s. Columns describes various aspects about each individual column. ‘name‘ and ‘id‘ are the only required parameters.

`active_cell` Lists containing elements 'row', 'column', 'row_id', 'column_id'. those elements have the following types: - row (numeric; optional) - column (numeric; optional) - row_id (character | numeric; optional) - column_id (character; optional). The row and column indices and IDs of the currently active cell. ‘row_id‘ is only returned if the data rows have an ‘id‘ key.

`include_headers_on_copy_paste`
Logical. If true, headers are included when copying from the table to different tabs and elsewhere. Note that headers are ignored when copying from the table onto itself and between two tables within the same tab.

`locale_format` Lists containing elements 'symbol', 'decimal', 'group', 'grouping', 'numerals', 'percent', 'separate_4digits'. those elements have the following types: - symbol (list of characters; optional): (default: ['$', '']). a list of two strings representing the prefix and suffix symbols. typically used for currency, and implemented using d3's currency format, but you can use this for other symbols such as measurement units. - decimal (character; optional): (default: '.'). the string used for the decimal separator. - group (character; optional): (default: ','). the string used for the groups separator. - grouping (list of numerics; optional): (default: [3]). a list of integers representing the grouping pattern. - numerals (list of characters; optional): a list of ten strings used as replacements for numbers 0-9. - percent (character; optional): (default: '%'). the string used for the percentage symbol. - separate_4digits (logical; optional): (default: true). separate integers with 4-digits or less.. The localization specific formatting information applied to all columns in the table. This prop is derived from the [d3.formatLocale](https://github.com/d3/d3-format#formatLocale) data structure specification. When left unspecified, each individual nested prop will default to a pre-determined value.

`markdown_options`
Lists containing elements 'link_target', 'html'. those elements have the following types: - link_target (character | a value equal to: '_blank', '_parent', '_self', '_top'; optional): (default: '_blank'). the link's behavior (_blank opens the link in a new tab, _parent opens the link in the parent frame, _self opens the link in the current tab, and _top opens the link in the top frame) or a string - html (logical; optional): (default: false) if true, html may be used in markdown cells be careful enabling html if the content being rendered can come from an untrusted user, as this may create an xss vulnerability.. The ‘markdown_options‘ property allows customization of the markdown cells behavior.

`css` List of lists containing elements 'selector', 'rule'. those elements have the following types: - selector (character; required) - rule (character; required)s. The ‘css‘ property is a way to embed CSS selectors and rules onto the page. We recommend starting with the ‘style_*‘ properties before using this ‘css‘ property. Example: ["selector": ".dash-spreadsheet", "rule": 'font-family: "monospace"']

data_previous List of named lists. The previous state of ‘data‘. ‘data_previous‘ has the same structure as ‘data‘ and it will be updated whenever ‘data‘ changes, either through a callback or by editing the table. This is a read-only property: setting this property will not have any impact on the table.

data_timestamp Numeric. The unix timestamp when the data was last edited. Use this property with other timestamp properties (such as ‘n_clicks_timestamp‘ in ‘dash_html_components‘) to determine which property has changed within a callback.

editable Logical. If True, then the data in all of the cells is editable. When ‘editable‘ is True, particular columns can be made uneditable by setting ‘editable‘ to ‘False‘ inside the ‘columns‘ property. If False, then the data in all of the cells is uneditable. When ‘editable‘ is False, particular columns can be made editable by setting ‘editable‘ to ‘True‘ inside the ‘columns‘ property.

end_cell Lists containing elements ’row’, ’column’, ’row_id’, ’column_id’. those elements have the following types: - row (numeric; optional) - column (numeric; optional) - row_id (character | numeric; optional) - column_id (character; optional). When selecting multiple cells (via clicking on a cell and then shift-clicking on another cell), ‘end_cell‘ represents the row / column coordinates and IDs of the cell in one of the corners of the region. ‘start_cell‘ represents the coordinates of the other corner.

export_columns A value equal to: ’all’, ’visible’. Denotes the columns that will be used in the export data file. If ‘all‘, all columns will be used (visible + hidden). If ‘visible‘, only the visible columns will be used. Defaults to ‘visible‘.

export_format A value equal to: ’csv’, ’xlsx’, ’none’. Denotes the type of the export data file, Defaults to ‘’none’‘

export_headers A value equal to: ’none’, ’ids’, ’names’, ’display’. Denotes the format of the headers in the export data file. If ‘’none’‘, there will be no header. If ‘’display’‘, then the header of the data file will be be how it is currently displayed. Note that ‘’display’‘ is only supported for ‘’xlsx’‘ export_format and will behave like ‘’names’‘ for ‘’csv’‘ export format. If ‘’ids’‘ or ‘’names’‘, then the headers of data file will be the column id or the column names, respectively

fill_width Logical. ‘fill_width‘ toggles between a set of CSS for two common behaviors: True: The table container’s width will grow to fill the available space; False: The table container’s width will equal the width of its content.

hidden_columns List of characters. List of columns ids of the columns that are currently hidden. See the associated nested prop ‘columns.hideable‘.

is_focused Logical. If True, then the ‘active_cell‘ is in a focused state.

merge_duplicate_headers
Logical. If True, then column headers that have neighbors with duplicate names will be merged into a single cell. This will be applied for single column headers and multi-column headers.

fixed_columns Lists containing elements ’data’, ’headers’. those elements have the following types: - data (a value equal to: 0; optional): example ‘’headers’:false, ’data’:0‘ no columns are fixed (the default) - headers (a value equal to: false; optional) | lists containing elements ’data’, ’headers’. those elements have the following types: - data (numeric; optional): example ‘’headers’:true, ’data’:1‘ one column

is fixed. - headers (a value equal to: true; required). ‘fixed_columns‘ will "fix" the set of columns so that they remain visible when scrolling horizontally across the unfixed columns. ‘fixed_columns‘ fixes columns from left-to-right. If ‘headers‘ is False, no columns are fixed. If ‘headers‘ is True, all operation columns (see ‘row_deletable‘ and ‘row_selectable‘) are fixed. Additional data columns can be fixed by assigning a number to ‘data‘.

Note that fixing columns introduces some changes to the underlying markup of the table and may impact the way that your columns are rendered or sized. View the documentation examples to learn more.

`fixed_rows` Lists containing elements ’data’, ’headers’. those elements have the following types: - data (a value equal to: 0; optional): example ‘’headers’:false, ’data’:0‘ no rows are fixed (the default) - headers (a value equal to: false; optional) | lists containing elements ’data’, ’headers’. those elements have the following types: - data (numeric; optional): example ‘’headers’:true, ’data’:1‘ one row is fixed. - headers (a value equal to: true; required). ‘fixed_rows‘ will "fix" the set of rows so that they remain visible when scrolling vertically down the table. ‘fixed_rows‘ fixes rows from top-to-bottom, starting from the headers. If ‘headers‘ is False, no rows are fixed. If ‘headers‘ is True, all header and filter rows (see ‘filter_action‘) are fixed. Additional data rows can be fixed by assigning a number to ‘data‘. Note that fixing rows introduces some changes to the underlying markup of the table and may impact the way that your columns are rendered or sized. View the documentation examples to learn more.

`column_selectable`

A value equal to: ’single’, ’multi’, false. If ‘single‘, then the user can select a single column or group of merged columns via the radio button that will appear in the header rows. If ‘multi‘, then the user can select multiple columns or groups of merged columns via the checkbox that will appear in the header rows. If false, then the user will not be able to select columns and no input will appear in the header rows. When a column is selected, its id will be contained in ‘selected_columns‘ and ‘derived_viewport_selected_columns‘.

`row_deletable` Logical. If True, then a ‘x‘ will appear next to each ‘row‘ and the user can delete the row.

`cell_selectable`

Logical. If True (default), then it is possible to click and navigate table cells.

`row_selectable` A value equal to: ’single’, ’multi’, false. If ‘single‘, then the user can select a single row via a radio button that will appear next to each row. If ‘multi‘, then the user can select multiple rows via a checkbox that will appear next to each row. If false, then the user will not be able to select rows and no additional UI elements will appear. When a row is selected, its index will be contained in ‘selected_rows‘.

`selected_cells` List of lists containing elements ’row’, ’column’, ’row_id’, ’column_id’. those elements have the following types: - row (numeric; optional) - column (numeric; optional) - row_id (character | numeric; optional) - column_id (character; optional)s. ‘selected_cells‘ represents the set of cells that are selected, as an array of objects, each item similar to ‘active_cell‘. Multiple cells can be selected by holding down shift and clicking on a different cell or holding down shift and navigating with the arrow keys.

selected_rows List of numerics. ‘selected_rows‘ contains the indices of rows that are selected via the UI elements that appear when ‘row_selectable‘ is ‘’single’‘ or ‘’multi’‘.

selected_columns
List of characters. ‘selected_columns‘ contains the ids of columns that are selected via the UI elements that appear when ‘column_selectable‘ is ‘’single’ or ’multi’‘.

selected_row_ids
List of character | numerics. ‘selected_row_ids‘ contains the ids of rows that are selected via the UI elements that appear when ‘row_selectable‘ is ‘’single’‘ or ‘’multi’‘.

start_cell Lists containing elements ’row’, ’column’, ’row_id’, ’column_id’. those elements have the following types: - row (numeric; optional) - column (numeric; optional) - row_id (character | numeric; optional) - column_id (character; optional). When selecting multiple cells (via clicking on a cell and then shift-clicking on another cell), ‘start_cell‘ represents the [row, column] coordinates of the cell in one of the corners of the region. ‘end_cell‘ represents the coordinates of the other corner.

style_as_list_view
Logical. If True, then the table will be styled like a list view and not have borders between the columns.

page_action A value equal to: ’custom’, ’native’, ’none’. ‘page_action‘ refers to a mode of the table where not all of the rows are displayed at once: only a subset are displayed (a "page") and the next subset of rows can viewed by clicking "Next" or "Previous" buttons at the bottom of the page. Pagination is used to improve performance: instead of rendering all of the rows at once (which can be expensive), we only display a subset of them. With pagination, we can either page through data that exists in the table (e.g. page through ‘10,000‘ rows in ‘data‘ ‘100‘ rows at a time) or we can update the data on-the-fly with callbacks when the user clicks on the "Previous" or "Next" buttons. These modes can be toggled with this ‘page_action‘ parameter: ‘’native’‘: all data is passed to the table up-front, paging logic is handled by the table; ‘’custom’‘: data is passed to the table one page at a time, paging logic is handled via callbacks; ‘’none’‘: disables paging, render all of the data at once.

page_current Numeric. ‘page_current‘ represents which page the user is on. Use this property to index through data in your callbacks with backend paging.

page_count Numeric. ‘page_count‘ represents the number of the pages in the paginated table. This is really only useful when performing backend pagination, since the front end is able to use the full size of the table to calculate the number of pages.

page_size Numeric. ‘page_size‘ represents the number of rows that will be displayed on a particular page when ‘page_action‘ is ‘’custom’‘ or ‘’native’‘

dropdown List with named elements and values of type lists containing elements ’clearable’, ’options’. those elements have the following types: - clearable (logical; optional) - options (required): . options has the following type: list of lists containing elements ’label’, ’value’. those elements have the following types: - label (character; required) - value (numeric | character | logical; required)s. ‘dropdown‘ specifies dropdown options for different columns. Each entry refers

to the column ID. The ‘clearable‘ property defines whether the value can be deleted. The ‘options‘ property refers to the ‘options‘ of the dropdown.

`dropdown_conditional`

List of lists containing elements 'clearable', 'if', 'options'. those elements have the following types: - clearable (logical; optional) - if (optional): . if has the following type: lists containing elements 'column_id', 'filter_query'. those elements have the following types: - column_id (character; optional) - filter_query (character; optional) - options (required): . options has the following type: list of lists containing elements 'label', 'value'. those elements have the following types: - label (character; required) - value (numeric | character | logical; required)ss. ‘dropdown_conditional‘ specifies dropdown options in various columns and cells. This property allows you to specify different dropdowns depending on certain conditions. For example, you may render different "city" dropdowns in a row depending on the current value in the "state" column.

`dropdown_data` List of list with named elements and values of type lists containing elements 'clearable', 'options'. those elements have the following types: - clearable (logical; optional) - options (required): . options has the following type: list of lists containing elements 'label', 'value'. those elements have the following types: - label (character; required) - value (numeric | character | logical; required)ss. ‘dropdown_data‘ specifies dropdown options on a row-by-row, column-by-column basis. Each item in the array corresponds to the corresponding dropdowns for the ‘data‘ item at the same index. Each entry in the item refers to the Column ID.

`tooltip` List with named elements and values of type character | lists containing elements 'delay', 'duration', 'type', 'use_with', 'value'. those elements have the following types: - delay (numeric; optional): represents the delay in milliseconds before the tooltip is shown when hovering a cell. this overrides the table's ‘tooltip_delay‘ property. if set to ‘none‘, the tooltip will be shown immediately. - duration (numeric; optional): represents the duration in milliseconds during which the tooltip is shown when hovering a cell. this overrides the table's ‘tooltip_duration‘ property. if set to ‘none‘, the tooltip will not disappear. - type (a value equal to: 'text', 'markdown'; optional): refers to the type of tooltip syntax used for the tooltip generation. can either be ‘markdown‘ or ‘text‘. defaults to ‘text‘. - use_with (a value equal to: 'both', 'data', 'header'; optional): refers to whether the tooltip will be shown only on data or headers. can be ‘both‘, ‘data‘, ‘header‘. defaults to ‘both‘. - value (character; required): refers to the syntax-based content of the tooltip. this value is required. alternatively, the value of the property can also be a plain string. the ‘text‘ syntax will be used in that case.. ‘tooltip‘ is the column based tooltip configuration applied to all rows. The key is the column id and the value is a tooltip configuration. Example: i: 'value': i, 'use_with: 'both' for i in df.columns

`tooltip_conditional`

List of lists containing elements 'delay', 'duration', 'if', 'type', 'value'. those elements have the following types: - delay (numeric; optional): the ‘delay‘ represents the delay in milliseconds before the tooltip is shown when hovering a cell. this overrides the table's ‘tooltip_delay‘ property. if set to ‘none‘, the tooltip will be shown immediately. - duration (numeric; optional): the ‘duration‘ represents the duration in milliseconds during which the tooltip is shown

when hovering a cell. this overrides the table's ‘tooltip_duration‘ property. if set to ‘none‘, the tooltip will not disappear. - if (required): the ‘if‘ refers to the condition that needs to be fulfilled in order for the associated tooltip configuration to be used. if multiple conditions are defined, all conditions must be met for the tooltip to be used by a cell.. if has the following type: lists containing elements 'column_id', 'filter_query', 'row_index'. those elements have the following types: - column_id (character; optional): ‘column_id‘ refers to the column id that must be matched. - filter_query (character; optional): ‘filter_query‘ refers to the query that must evaluate to true. - row_index (numeric | a value equal to: 'odd', 'even'; optional): ‘row_index‘ refers to the index of the row in the source ‘data‘. - type (a value equal to: 'text', 'markdown'; optional): the ‘type‘ refers to the type of tooltip syntax used for the tooltip generation. can either be ‘markdown‘ or ‘text‘. defaults to ‘text‘. - value (character; required): the ‘value‘ refers to the syntax-based content of the tooltip. this value is required.s. ‘tooltip_conditional‘ represents the tooltip shown for different columns and cells. This property allows you to specify different tooltips depending on certain conditions. For example, you may have different tooltips in the same column based on the value of a certain data property. Priority is from first to last defined conditional tooltip in the list. Higher priority (more specific) conditional tooltips should be put at the beginning of the list.

`tooltip_data` List of list with named elements and values of type character | lists containing elements 'delay', 'duration', 'type', 'value'. those elements have the following types: - delay (numeric; optional): the ‘delay‘ represents the delay in milliseconds before the tooltip is shown when hovering a cell. this overrides the table's ‘tooltip_delay‘ property. if set to ‘none‘, the tooltip will be shown immediately. - duration (numeric; optional): the ‘duration‘ represents the duration in milliseconds during which the tooltip is shown when hovering a cell. this overrides the table's ‘tooltip_duration‘ property. if set to ‘none‘, the tooltip will not disappear. alternatively, the value of the property can also be a plain string. the ‘text‘ syntax will be used in that case. - type (a value equal to: 'text', 'markdown'; optional): for each tooltip configuration, the ‘type‘ refers to the type of tooltip syntax used for the tooltip generation. can either be ‘markdown‘ or ‘text‘. defaults to ‘text‘. - value (character; required): the ‘value‘ refers to the syntax-based content of the tooltip. this value is required.s. ‘tooltip_data‘ represents the tooltip shown for different columns and cells. A list of dicts for which each key is a column id and the value is a tooltip configuration.

`tooltip_header` List with named elements and values of type character | lists containing elements 'delay', 'duration', 'type', 'value'. those elements have the following types: - delay (numeric; optional): the ‘delay‘ represents the delay in milliseconds before the tooltip is shown when hovering a cell. this overrides the table's ‘tooltip_delay‘ property. if set to ‘none‘, the tooltip will be shown immediately. - duration (numeric; optional): the ‘duration‘ represents the duration in milliseconds during which the tooltip is shown when hovering a cell. this overrides the table's ‘tooltip_duration‘ property. if set to ‘none‘, the tooltip will not disappear. alternatively, the value of the property can also be a plain string. the ‘text‘ syntax will be used in that case. - type (a value equal to: 'text', 'markdown'; optional): for each tooltip configuration, the ‘type‘ refers to the type of tooltip syntax used for the tooltip generation. can either be ‘markdown‘ or ‘text‘. defaults to ‘text‘.

- value (character; required): the ‘value‘ refers to the syntax-based content of the tooltip. this value is required. | list of a value equal to: null | character | lists containing elements ’delay’, ’duration’, ’type’, ’value’. those elements have the following types: - delay (numeric; optional) - duration (numeric; optional) - type (a value equal to: ’text’, ’markdown’; optional) - value (character; required)s. ‘tooltip_header‘ represents the tooltip shown for each header column and optionally each header row. Example to show long column names in a tooltip: i: i for i in df.columns. Example to show different column names in a tooltip: ’Rep’: ’Republican’, ’Dem’: ’Democrat’. If the table has multiple rows of headers, then use a list as the value of the ‘tooltip_header‘ items.

`tooltip_delay` Numeric. ‘tooltip_delay‘ represents the table-wide delay in milliseconds before the tooltip is shown when hovering a cell. If set to ‘None‘, the tooltip will be shown immediately. Defaults to 350.

`tooltip_duration`

Numeric. ‘tooltip_duration‘ represents the table-wide duration in milliseconds during which the tooltip will be displayed when hovering a cell. If set to ‘None‘, the tooltip will not disappear. Defaults to 2000.

`filter_query` Character. If ‘filter_action‘ is enabled, then the current filtering string is represented in this ‘filter_query‘ property.

`filter_action` A value equal to: ’custom’, ’native’, ’none’ | lists containing elements ’type’, ’operator’. those elements have the following types: - type (a value equal to: ’custom’, ’native’; required) - operator (a value equal to: ’and’, ’or’; optional). The ‘filter_action‘ property controls the behavior of the ‘filtering‘ UI. If ‘’none’‘, then the filtering UI is not displayed. If ‘’native’‘, then the filtering UI is displayed and the filtering logic is handled by the table. That is, it is performed on the data that exists in the ‘data‘ property. If ‘’custom’‘, then the filtering UI is displayed but it is the responsibility of the developer to program the filtering through a callback (where ‘filter_query‘ or ‘derived_filter_query_structure‘ would be the input and ‘data‘ would be the output).

`filter_options` Lists containing elements ’case’. those elements have the following types: - case (a value equal to: ’sensitive’, ’insensitive’; optional). There are two ‘filter_options‘ props in the table. This is the table-level filter_options prop and there is also the column-level ‘filter_options‘ prop. These props determine whether the applicable filter relational operators will default to ‘sensitive‘ or ‘insensitive‘ comparison. If the column-level ‘filter_options‘ prop is set it overrides the table-level ‘filter_options‘ prop for that column.

`sort_action` A value equal to: ’custom’, ’native’, ’none’. The ‘sort_action‘ property enables data to be sorted on a per-column basis. If ‘’none’‘, then the sorting UI is not displayed. If ‘’native’‘, then the sorting UI is displayed and the sorting logic is handled by the table. That is, it is performed on the data that exists in the ‘data‘ property. If ‘’custom’‘, the the sorting UI is displayed but it is the responsibility of the developer to program the sorting through a callback (where ‘sort_by‘ would be the input and ‘data‘ would be the output). Clicking on the sort arrows will update the ‘sort_by‘ property.

`sort_mode` A value equal to: ’single’, ’multi’. Sorting can be performed across multiple columns (e.g. sort by country, sort within each country, sort by year) or by a single column. NOTE - With multi-column sort, it’s currently not possible

to determine the order in which the columns were sorted through the UI. See https://github.com/plotly/dash-table/issues/170

`sort_by` List of lists containing elements 'column_id', 'direction'. those elements have the following types: - column_id (character; required) - direction (a value equal to: 'asc', 'desc'; required)s. 'sort_by' describes the current state of the sorting UI. That is, if the user clicked on the sort arrow of a column, then this property will be updated with the column ID and the direction ('asc' or 'desc') of the sort. For multi-column sorting, this will be a list of sorting parameters, in the order in which they were clicked.

`sort_as_null` List of character | numeric | logicals. An array of string, number and boolean values that are treated as 'None' (i.e. ignored and always displayed last) when sorting. This value will be used by columns without 'sort_as_null'. Defaults to '[]'.

`style_table` Named list. CSS styles to be applied to the outer 'table' container. This is commonly used for setting properties like the width or the height of the table.

`style_cell` Named list. CSS styles to be applied to each individual cell of the table. This includes the header cells, the 'data' cells, and the filter cells.

`style_data` Named list. CSS styles to be applied to each individual data cell. That is, unlike 'style_cell', it excludes the header and filter cells.

`style_filter` Named list. CSS styles to be applied to the filter cells. Note that this may change in the future as we build out a more complex filtering UI.

`style_header` Named list. CSS styles to be applied to each individual header cell. That is, unlike 'style_cell', it excludes the 'data' and filter cells.

`style_cell_conditional`
List of lists containing elements 'if'. those elements have the following types: - if (optional): . if has the following type: lists containing elements 'column_id', 'column_type'. those elements have the following types: - column_id (character | list of characters; optional) - column_type (a value equal to: 'any', 'numeric', 'text', 'datetime'; optional)s. Conditional CSS styles for the cells. This can be used to apply styles to cells on a per-column basis.

`style_data_conditional`
List of lists containing elements 'if'. those elements have the following types: - if (optional): . if has the following type: lists containing elements 'column_id', 'column_type', 'filter_query', 'state', 'row_index', 'column_editable'. those elements have the following types: - column_id (character | list of characters; optional) - column_type (a value equal to: 'any', 'numeric', 'text', 'datetime'; optional) - filter_query (character; optional) - state (a value equal to: 'active', 'selected'; optional) - row_index (numeric | a value equal to: 'odd', 'even' | list of numerics; optional) - column_editable (logical; optional)s. Conditional CSS styles for the data cells. This can be used to apply styles to data cells on a per-column basis.

`style_filter_conditional`
List of lists containing elements 'if'. those elements have the following types: - if (optional): . if has the following type: lists containing elements 'column_id', 'column_type', 'column_editable'. those elements have the following types: -

column_id (character | list of characters; optional) - column_type (a value equal to: 'any', 'numeric', 'text', 'datetime'; optional) - column_editable (logical; optional)s. Conditional CSS styles for the filter cells. This can be used to apply styles to filter cells on a per-column basis.

`style_header_conditional`
: List of lists containing elements 'if'. those elements have the following types: - if (optional): . if has the following type: lists containing elements 'column_id', 'column_type', 'header_index', 'column_editable'. those elements have the following types: - column_id (character | list of characters; optional) - column_type (a value equal to: 'any', 'numeric', 'text', 'datetime'; optional) - header_index (numeric | list of numerics | a value equal to: 'odd', 'even'; optional) - column_editable (logical; optional)s. Conditional CSS styles for the header cells. This can be used to apply styles to header cells on a per-column basis.

`virtualization`
: Logical. This property tells the table to use virtualization when rendering. Assumptions are that: the width of the columns is fixed; the height of the rows is always the same; and runtime styling changes will not affect width and height vs. first rendering

`derived_filter_query_structure`
: Named list. This property represents the current structure of 'filter_query' as a tree structure. Each node of the query structure has: type (string; required): 'open-block', 'logical-operator', 'relational-operator', 'unary-operator', or 'expression'; subType (string; optional): 'open-block': '()', 'logical-operator': '&&', '||', 'relational-operator': '=', '>=', '>', '<=', '<', '!=', 'contains', 'unary-operator': '!', 'is bool', 'is even', 'is nil', 'is num', 'is object', 'is odd', 'is prime', 'is str', 'expression': 'value', 'field'; value (any): 'expression, value': passed value, 'expression, field': the field/prop name. block (nested query structure; optional). left (nested query structure; optional). right (nested query structure; optional). If the query is invalid or empty, the 'derived_filter_query_structure' will be 'None'.

`derived_viewport_data`
: List of named lists. This property represents the current state of 'data' on the current page. This property will be updated on paging, sorting, and filtering.

`derived_viewport_indices`
: List of numerics. 'derived_viewport_indices' indicates the order in which the original rows appear after being filtered, sorted, and/or paged. 'derived_viewport_indices' contains indices for the current page, while 'derived_virtual_indices' contains indices across all pages.

`derived_viewport_row_ids`
: List of character | numerics. 'derived_viewport_row_ids' lists row IDs in the order they appear after being filtered, sorted, and/or paged. 'derived_viewport_row_ids' contains IDs for the current page, while 'derived_virtual_row_ids' contains IDs across all pages.

`derived_viewport_selected_columns`
: List of characters. 'derived_viewport_selected_columns' contains the ids of the 'selected_columns' that are not currently hidden.

`derived_viewport_selected_rows`
: List of numerics. 'derived_viewport_selected_rows' represents the indices of the 'selected_rows' from the perspective of the 'derived_viewport_indices'.

derived_viewport_selected_row_ids
: List of character | numerics. ‘derived_viewport_selected_row_ids‘ represents the IDs of the ‘selected_rows‘ on the currently visible page.

derived_virtual_data
: List of named lists. This property represents the visible state of ‘data‘ across all pages after the front-end sorting and filtering as been applied.

derived_virtual_indices
: List of numerics. ‘derived_virtual_indices‘ indicates the order in which the original rows appear after being filtered and sorted. ‘derived_viewport_indices‘ contains indices for the current page, while ‘derived_virtual_indices‘ contains indices across all pages.

derived_virtual_row_ids
: List of character | numerics. ‘derived_virtual_row_ids‘ indicates the row IDs in the order in which they appear after being filtered and sorted. ‘derived_viewport_row_ids‘ contains IDs for the current page, while ‘derived_virtual_row_ids‘ contains IDs across all pages.

derived_virtual_selected_rows
: List of numerics. ‘derived_virtual_selected_rows‘ represents the indices of the ‘selected_rows‘ from the perspective of the ‘derived_virtual_indices‘.

derived_virtual_selected_row_ids
: List of character | numerics. ‘derived_virtual_selected_row_ids‘ represents the IDs of the ‘selected_rows‘ as they appear after filtering and sorting, across all pages.

loading_state
: Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence
: Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn’t changed from its previous value, any ‘persisted_props‘ that the user has changed while using the app will keep those changes, as long as the new prop value also matches what was given originally. Used in conjunction with ‘persistence_type‘ and ‘persisted_props‘.

persisted_props
: List of a value equal to: ’columns.name’, ’data’, ’filter_query’, ’hidden_columns’, ’selected_columns’, ’selected_rows’, ’sort_by’s. Properties whose user interactions will persist after refreshing the component or the page.

persistence_type
: A value equal to: ’local’, ’session’, ’memory’. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# For comprehensive documentation of this package's features,
# please consult https://dashr.plot.ly/datatable
#
# A package vignette is currently in development and will
# provide many of the same examples currently available online
# in an offline-friendly format.

# The following if statement is not required to run this
# example locally, but was added at the request of CRAN
# maintainers.
if (interactive() && require(dash)) {
  library(dash)

  app <- Dash$new()

  # We can easily restrict the number of rows to display at
  # once by using style_table:
  app$layout(
    dashDataTable(
      id = "table",
      columns = lapply(colnames(iris),
                       function(colName){
                         list(
                           id = colName,
                           name = colName
                         )
                       }),
      style_table = list(
        maxHeight = "250px",
        overflowY = "scroll"
      ),
      data = df_to_list(iris)
    )
  )

  app$run_server()

  app <- Dash$new()

  # We can also make rows and columns selectable/deletable
  # by setting a few additional attributes:
  app$layout(
    dashDataTable(
      id = "table",
      columns = lapply(colnames(iris),
                       function(colName){
                         list(
                           id = colName,
                           name = colName,
                           deletable = TRUE
                         )
```

```
                    }),
      style_table = list(
        maxHeight = "250px",
        overflowY = "scroll"
      ),
      data = df_to_list(iris),
      editable = TRUE,
      filter_action = "native",
      sort_action = "native",
      sort_mode = "multi",
      column_selectable = "single",
      row_selectable = "multi",
      row_deletable = TRUE
    )
  )

  app$run_server()
}
```

---

dash_app *Create a Dash application*

Description

This is a convenience function that returns a Dash R6 object. For advanced usage, you can use the object as an R6 object directly instead of the functions provided by the {dash} package.

Usage

```
dash_app(
  title = NULL,
  update_title = "Updating...",
  assets_folder = "assets",
  assets_url_path = "/assets",
  assets_ignore = NULL,
  eager_loading = FALSE,
  serve_locally = TRUE,
  pathname_url_base = "/",
  pathname_routes_prefix = NULL,
  pathname_requests_prefix = NULL,
  compress = TRUE,
  suppress_callback_exceptions = FALSE,
  show_undo_redo = FALSE
)
```

Arguments

title	*(character)* The browser window title.

update_title *(character)* The browser window title while a callback is being processed. Set to NULL or "" if you don't want Dash to automatically update the window title.

assets_folder *(character)* Path (relative to the current working directory) containing extra files to be served by the browser. All files with ".js" or ".css" extensions will automatically be included on the page, unless excluded with assets_ignore. Any other files, such as images, will only be served if explicitly requested.

assets_url_path *(character)* URL path for serving assets. For example, a value of "www" means that any request path that begins with "/www" will be mapped to the assets_folder. If your assets are hosted online, you can provide a CDN URL, such as "http://your-assets-website".

assets_ignore *(character)* Regular expression for ".js" and ".css" files that should not be automatically included. Ignored files will still be served if explicitly requested. Note that you cannot use this to prevent access to sensitive files since ignored files are accessible by users.

eager_loading *(logical)* Whether asynchronous resources are prefetched (TRUE) or loaded on-demand (FALSE).

serve_locally *(logical)* Whether to serve HTML dependencies locally or remotely (via URL).

pathname_url_base *(character)* Local URL prefix to use app-wide.

pathname_routes_prefix *(character)* Prefix applied to the backend routes. Defaults to pathname_url_base.

pathname_requests_prefix *(character)* Prefix applied to request endpoints made by Dash's front-end. Defaults to pathname_url_base.

compress *(logical)* Whether to try to compress files and data. If TRUE, then brotli compression is attempted first, then gzip, then the deflate algorithm, before falling back to identity.

suppress_callback_exceptions *(logical)* Whether to relay warnings about possible layout mis-specifications when registering a callback.

show_undo_redo *(logical)* If TRUE, the app will have undo and redo buttons for stepping through the history of the app state.

See Also

run_app()

---

dbcAccordion *Accordion component*

Description

A self contained Accordion component. Build up the children using the AccordionItem component.

Usage

```
dbcAccordion(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, flush=NULL, active_item=NULL,
start_collapsed=NULL, loading_state=NULL, persistence=NULL,
persisted_props=NULL, persistence_type=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
flush	Logical. Renders accordion edge-to-edge with its parent container
active_item	Character. The item_id of the currently active item. If item_id has not been specified for the active item, this will default to item-i, where i is the index (starting from 0) of the item.
start_collapsed	Logical. Set to True for all items to be collapsed initially.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
persistence	Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn’t changed from its previous value, a ‘value‘ that the user has changed while using the app will keep that change, as long as the new ‘value‘ also matches what was given originally. Used in conjunction with ‘persistence_type‘.
persisted_props	List of a value equal to: ’active_item’s. Properties whose user interactions will persist after refreshing the component or the page. Since only ‘value‘ is allowed this prop can normally be ignored.
persistence_type	A value equal to: ’local’, ’session’, ’memory’. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcAccordionItem` *AccordionItem component*

Description

A component to build up the children of the accordion.

Usage

```
dbcAccordionItem(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, title=NULL, item_id=NULL, loading_state=NULL)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`style`	Named list. Defines CSS styles which will override styles previously set.
`class_name`	Character. Often used with CSS to style elements with common properties.
`className`	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
`title`	Character. The title on display in the collapsed accordion item.
`item_id`	Character. Optional identifier for item used for determining which item is visible if not specified, and AccordionItem is being used inside Accordion component, the itemId will be set to "item-i" where i is (zero indexed) position of item in list items pased to Accordion component.
`loading_state`	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcAlert *Alert component*

Description

Alert allows you to create contextual feedback messages on user actions. Control the visibility using callbacks with the ‘is_open‘ prop, or set it to auto-dismiss with the ‘duration‘ prop.

Usage

```
dbcAlert(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, color=NULL, is_open=NULL,
fade=NULL, dismissable=NULL, duration=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component.
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
color	Character. Alert color, options: primary, secondary, success, info, warning, danger, link or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name) Default: secondary.
is_open	Logical. Whether alert is open. Default: True.
fade	Logical. If True, a fade animation will be applied when ‘is_open‘ is toggled. If False the Alert will simply appear and disappear.
dismissable	Logical. If true, add a close button that allows Alert to be dismissed.
duration	Numeric. Duration in milliseconds after which the Alert dismisses itself.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcBadge *Badge component*

Description

Badges can be used to add counts or labels to other components.

Usage

```
dbcBadge(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, color=NULL, text_color=NULL,
pill=NULL, href=NULL, tag=NULL, loading_state=NULL,
external_link=NULL, n_clicks=NULL, n_clicks_timestamp=NULL,
target=NULL, title=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component.
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
color	Character. Badge color, options: primary, secondary, success, info, warning, danger, link or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name) Default: secondary.
text_color	Character. Badge color, options: primary, secondary, success, info, warning, danger, link or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name) Default: secondary.
pill	Logical. Make badge "pill" shaped (rounded ends, like a pill). Default: False.
href	Character. Attach link to badge.
tag	Character. HTML tag to use for the Badge. Default: span.

loading_state Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

external_link Logical. If true, the browser will treat this as an external link, forcing a page refresh at the new location. If false, this just changes the location without triggering a page refresh. Use this if you are observing dcc.Location, for instance. Defaults to true for absolute URLs and false otherwise.

n_clicks Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

target Character. Target attribute to pass on to the link. Only applies to external links.

title Character. Sets the title attribute of the underlying HTML button.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcBreadcrumb *Breadcrumb component*

Description

Use breadcrumbs to create a navigation breadcrumb in your app.

Usage

```
dbcBreadcrumb(id=NULL, items=NULL, style=NULL, item_style=NULL,
class_name=NULL, className=NULL, item_class_name=NULL,
itemClassName=NULL, key=NULL, tag=NULL, loading_state=NULL)
```

Arguments

id Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

items List of lists containing elements 'label', 'href', 'active', 'external_link', 'target', 'title'. those elements have the following types: - label (character; optional): label to display inside the breadcrumbs. - href (character; optional): url of the resource to link to - active (logical; optional): apply 'active' style to this component. - external_link (logical; optional): if true, the browser will treat this as an external link, forcing a page refresh at the new location. if false, this just

changes the location without triggering a page refresh. use this if you are observing dcc.location, for instance. defaults to true for absolute urls and false otherwise. - target (character; optional): target attribute to pass on to the link. only applies to external links. - title (character; optional): title attribute for the inner a elements. The details of the items to render inside of this component.

`style` Named list. Defines CSS styles which will override styles previously set.

`item_style` Named list. Defines inline CSS styles that will be added to each item in the breadcrumbs.

`class_name` Character. Often used with CSS to style elements with common properties.

`className` Character. **DEPRECATED** - Use class_name instead.

Often used with CSS to style elements with common properties.

`item_class_name`

Character. Class name to apply to each item.

`itemClassName` Character. **DEPRECATED** - use item_class_name instead.

Class name ot apply to each item.

`key` Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

`tag` Named list. HTML tag to use for the outer breadcrumb component. Default: "nav".

`loading_state` Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcButton` *Button component*

Description

A component for creating Bootstrap buttons with different style options. The Button component can act as a HTML button, link (<a>) or can be used like a dash_core_components style 'Link' for navigating between pages of a Dash app. Use the 'n_clicks' prop to trigger callbacks when the button has been clicked.

Usage

```
dbcButton(children=NULL, id=NULL, class_name=NULL, className=NULL,
style=NULL, key=NULL, href=NULL, external_link=NULL,
n_clicks=NULL, n_clicks_timestamp=NULL, active=NULL,
color=NULL, disabled=NULL, size=NULL, title=NULL,
outline=NULL, loading_state=NULL, target=NULL, type=NULL,
download=NULL, name=NULL, value=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component.
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
style	Named list. Defines CSS styles which will override styles previously set.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
href	Character. Pass a URL (relative or absolute) to make the menu entry a link.
external_link	Logical. If true, the browser will treat this as an external link, forcing a page refresh at the new location. If false, this just changes the location without triggering a page refresh. Use this if you are observing dcc.Location, for instance. Defaults to true for absolute URLs and false otherwise.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. Use of *_timestamp props has been deprecated in Dash in favour of dash.callback_context. See "How do I determine which Input has changed?" in the Dash FAQs https://dash.plot.ly/faqs. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
active	Logical. Whether button is in active state. Default: False.
color	Character. Button color, options: primary, secondary, success, info, warning, danger, link. Default: secondary.
disabled	Logical. Disable button (make unclickable). Default: False.
size	Character. Button size, options: ’lg’, ’md’, ’sm’.
title	Character. Sets the title attribute of the underlying HTML button.
outline	Logical. Set outline button style, which removes background images and colors for a lightweight style.

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
target	Character. Target attribute to pass on to link if using Button as an external link.
type	A value equal to: 'button', 'reset', 'submit'. The default behavior of the button. Possible values are: "button", "reset", "submit". If left unspecified the default depends on usage: for buttons associated with a form (e.g. a dbc.Button inside a dbc.Form) the default is "submit". Otherwise the default is "button".
download	Character. Indicates that the hyperlink is to be used for downloading a resource.
name	Character. The name of the button, submitted as a pair with the button's value as part of the form data.
value	Character. Defines the value associated with the button's name when it's submitted with the form data. This value is passed to the server in params when the form is submitted.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcButtonGroup *ButtonGroup component*

Description

A component for creating groups of buttons. Can be used with 'Button' or 'DropdownMenu'.

Usage

```
dbcButtonGroup(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, vertical=NULL, size=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
vertical	Logical. Group buttons vertically.
size	Character. Size of button group, options: 'sm', 'md', 'lg'.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcCard *Card component*

Description

Component for creating Bootstrap cards. Use in conjunction with CardBody, CardImg, CardLink, CardHeader and CardFooter. Can also be used in conjunction with CardColumns, CardDeck, CardGroup for different layout options.

Usage

```
dbcCard(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, color=NULL, body=NULL,
outline=NULL, inverse=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

color	Character. Card color, options: primary, secondary, success, info, warning, danger, light, dark or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name). Default is light.
body	Logical. Apply the ‘card-body‘ class to the card, so that there is no need to also include a CardBody component in the children of this Card. Default: False
outline	Logical. Apply color styling to just the border of the card.
inverse	Logical. Invert text colours for use with a darker background.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcCardBody` *CardBody component*

Description

Wrap the content of your ‘Card‘ in ‘CardBody‘ to apply padding and other styles.

Usage

```
dbcCardBody(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, tag=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
tag	Character. HTML tag to use for the card body, default: div

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcCardFooter *CardFooter component*

Description

Use the CardFooter component to add a footer to any card.

Usage

```
dbcCardFooter(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, tag=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
tag	Character. HTML tag to use for the card footer, default: div
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcCardGroup *CardGroup component*

Description

Use CardGroup to render cards as a single, attached element of columns with equal width and height.

Usage

```
dbcCardGroup(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, tag=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
tag	Character. HTML tag to use for the card group, default: div
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcCardHeader *CardHeader component*

Description

Use the CardHeader component to add a header to any card.

Usage

```
dbcCardHeader(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, tag=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
tag	Character. HTML tag to use for the card header, default: div
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcCardImg *CardImg component*

Description

Use CardImg to add images to your cards.

Usage

```
dbcCardImg(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, tag=NULL, top=NULL, bottom=NULL,
src=NULL, alt=NULL, title=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
tag	Character. HTML tag to use for the card body, default: div
top	Logical. Set to True if image is at top of card. This will apply the card-img-top class which rounds the top corners to match the corners of the card.
bottom	Logical. Set to True if image is at bottom of card. This will apply the card-img-bottom class which rounds the bottom corners to match the corners of the card.
src	Character. The URI of the embeddable content.
alt	Character. Alternative text in case an image can’t be displayed.
title	Character. Text to be displayed as a tooltip when hovering
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcCardImgOverlay *CardImgOverlay component*

Description

Use CardImgOverlay to turn an image into the background of your card and add text on top of it.

Usage

```
dbcCardImgOverlay(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, tag=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
tag	Character. HTML tag to use for the card image overlay, default: div
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcCardLink *CardLink component*

Description

Use card link to add consistently styled links to your cards. Links can be used like buttons, external links, or internal Dash style links.

Usage

```
dbcCardLink(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, href=NULL, external_link=NULL,
n_clicks=NULL, n_clicks_timestamp=NULL, loading_state=NULL,
target=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
href	Character. URL of the resource to link to
external_link	Logical. If true, the browser will treat this as an external link, forcing a page refresh at the new location. If false, this just changes the location without triggering a page refresh. Use this if you are observing dcc.Location, for instance. Defaults to true for absolute URLs and false otherwise.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
target	Character. Target attribute to pass on to the link. Only applies to external links.

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcCarousel` *Carousel component*

Description

Component for creating Bootstrap carousel. This component is a slideshow for cycling through a series of content.

Usage

```
dbcCarousel(id=NULL, style=NULL, class_name=NULL, className=NULL,
items=NULL, active_index=NULL, controls=NULL,
indicators=NULL, ride=NULL, slide=NULL, variant=NULL,
interval=NULL, loading_state=NULL)
```

Arguments

`id`	Character. The ID of the component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`style`	Named list. Defines CSS styles of the carousel container. Will override styles previously set.
`class_name`	Character. Defines the className of the carousel container. Often used with CSS to style elements with common properties.
`className`	Character. **DEPRECATED** Use ‘class_name‘ instead. efines the className of the carousel container. Often used with CSS to style elements with common properties.
`items`	List of lists containing elements ’key’, ’src’, ’alt’, ’img_class_name’, ’imgclassname’, ’img_style’, ’header’, ’caption’, ’caption_class_name’, ’captionclassname’. those elements have the following types: - key (character; optional): a unique identifier for the slide, used to improve performance by react.js while rendering components see https://reactjs.org/docs/lists-and-keys.html for more info. - src (character; optional): the url of the image - alt (character; optional): the alternate text for an image, if the image cannot be displayed - img_class_name (character; optional): the classname for the image. the default is ’d-block w-100’ - imgclassname (character; optional): **deprecated** use ‘img_class_name‘ instead. the classname for the image. the default is ’d-block w-100’ - img_style (named list; optional): the style for the image - header (character; optional): the header of the text on the slide. it is displayed in a <h5> element - caption (character; optional): the caption of the item. the text is displayed in a <p> element - caption_class_name (character; optional): the class name for the header and

	caption container - captionclassname (character; optional): **deprecated** use ‘caption_class_name‘ instead. the class name for the header and caption containers. The items to display on the slides in the carousel
active_index	Numeric. The current visible slide number
controls	Logical. Show the Carousel previous and next arrows for changing the current slide
indicators	Logical. Show a set of slide position indicators
ride	A value equal to: ’carousel’. Autoplays the carousel after the user manually cycles the first item. If "carousel", autoplays the carousel on load.
slide	Logical. controls whether the slide animation on the Carousel works or not
variant	A value equal to: ’dark’. Add ‘variant="dark"‘ to the Carousel for darker controls, indicators, and captions.
interval	Numeric. the interval at which the carousel automatically cycles (default: 5000) If set to None, carousel will not Autoplay (i.e. will not automatically cycle).
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcCheckbox *Checkbox component*

Description

Checklist is a component that encapsulates several checkboxes. The values and labels of the checklist is specified in the ‘options‘ property and the checked items are specified with the ‘value‘ property. Each checkbox is rendered as an input / label pair. ‘Checklist‘ must be given an ‘id‘ to work properly.

Usage

```
dbcCheckbox(id=NULL, class_name=NULL, className=NULL, style=NULL,
input_style=NULL, inputStyle=NULL, input_class_name=NULL,
inputClassName=NULL, label=NULL, label_id=NULL,
label_style=NULL, labelStyle=NULL, label_class_name=NULL,
labelClassName=NULL, name=NULL, disabled=NULL, value=NULL,
loading_state=NULL, persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
class_name	Character. The class of the container (div)
className	Character. **DEPRECATED** Use ‘class_name‘ instead. The class of the container (div)
style	Named list. The style of the container (div)
input_style	Named list. The style of the <input> checkbox element.
inputStyle	Named list. **DEPRECATED** Use ‘input_style‘ instead. The style of the <input> checkbox element.
input_class_name	Character. The class of the <input> checkbox element
inputClassName	Character. **DEPRECATED** Use ‘input_class_name‘ instead. The class of the <input> checkbox element
label	Character. The label of the <input> element
label_id	Character. The id of the label
label_style	Named list. Inline style arguments to apply to the <label> element for each item.
labelStyle	Named list. **DEPRECATED** Use ‘label_style‘ instead. Inline style arguments to apply to the <label> element for each item.
label_class_name	Character. CSS classes to apply to the <label> element for each item.
labelClassName	Character. **DEPRECATED** Use ‘label_class_name‘ instead. CSS classes to apply to the <label> element for each item.
name	Character. The name of the control, which is submitted with the form data.
disabled	Logical. Disable the Checkbox.
value	Logical. The value of the input.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
persistence	Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn’t changed from its previous value, a ‘value‘ that the user has changed while using the app will keep that change, as long as the new ‘value‘ also matches what was given originally. Used in conjunction with ‘persistence_type‘.
persisted_props	List of a value equal to: ’value’s. Properties whose user interactions will persist after refreshing the component or the page. Since only ‘value‘ is allowed this prop can normally be ignored.

persistence_type
: A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcChecklist *Checklist component*

Description

Checklist is a component that encapsulates several checkboxes. The values and labels of the checklist is specified in the 'options' property and the checked items are specified with the 'value' property. Each checkbox is rendered as an input / label pair. 'Checklist' must be given an 'id' to work properly.

Usage

```
dbcChecklist(id=NULL, options=NULL, value=NULL, class_name=NULL,
className=NULL, style=NULL, key=NULL, input_style=NULL,
inputStyle=NULL, input_checked_style=NULL,
inputCheckedStyle=NULL, input_class_name=NULL,
inputClassName=NULL, input_checked_class_name=NULL,
inputCheckedClassName=NULL, label_style=NULL,
labelStyle=NULL, label_checked_style=NULL,
labelCheckedStyle=NULL, label_class_name=NULL,
labelClassName=NULL, label_checked_class_name=NULL,
labelCheckedClassName=NULL, inline=NULL, switch=NULL,
loading_state=NULL, persistence=NULL, persisted_props=NULL,
persistence_type=NULL, name=NULL)
```

Arguments

id
: Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

options
: List of lists containing elements 'label', 'value', 'disabled', 'input_id', 'label_id'. those elements have the following types: - label (character | numeric; required): the checkbox's label - value (character | numeric; required): the value of the checkbox. this value corresponds to the items specified in the 'value' property. - disabled (logical; optional): if true, this checkbox is disabled and can't be clicked on. - input_id (character; optional): id for this option's input, can be used to attach tooltips or apply css styles - label_id (character; optional): id for this option's label, can be used to attach tooltips or apply css styless. An array of options

`value` List of character | numerics. The currently selected value

`class_name` Character. The class of the container (div)

`className` Character. **DEPRECATED** Use ‘class_name‘ instead.
The class of the container (div)

`style` Named list. The style of the container (div)

`key` Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

`input_style` Named list. The style of the <input> checkbox element.

`inputStyle` Named list. **DEPRECATED** Use ‘input_style‘ instead.
The style of the <input> checkbox element.

`input_checked_style`
Named list. Additional inline style arguments to apply to <input> elements on checked items.

`inputCheckedStyle`
Named list. **DEPRECATED** Use ‘input_checked_style‘ instead.
Additional inline style arguments to apply to <input> elements on checked items.

`input_class_name`
Character. The class of the <input> checkbox element

`inputClassName` Character. **DEPRECATED** Use ‘input_class_name‘ instead.
The class of the <input> checkbox element

`input_checked_class_name`
Character. Additional CSS classes to apply to the <input> element when the corresponding checkbox is checked.

`inputCheckedClassName`
Character. **DEPRECATED** Use ‘input_checked_class_name‘ instead.
Additional CSS classes to apply to the <input> element when the corresponding checkbox is checked.

`label_style` Named list. Inline style arguments to apply to the <label> element for each item.

`labelStyle` Named list. **DEPRECATED** Use ‘label_style‘ instead.
Inline style arguments to apply to the <label> element for each item.

`label_checked_style`
Named list. Additional inline style arguments to apply to <label> elements on checked items.

`labelCheckedStyle`
Named list. **DEPRECATED** Use ‘label_checked_style‘ instead.
Additional inline style arguments to apply to <label> elements on checked items.

`label_class_name`
Character. CSS classes to apply to the <label> element for each item.

`labelClassName` Character. **DEPRECATED** Use ‘label_class_name‘ instead.
CSS classes to apply to the <label> element for each item.

`label_checked_class_name`
Character. Additional CSS classes to apply to the <label> element when the corresponding checkbox is checked.

labelCheckedClassName
: Character. **DEPRECATED** Use ‘label_checked_class_name‘ instead.
Additional CSS classes to apply to the <label> element when the corresponding checkbox is checked.

inline
: Logical. Arrange Checklist inline

switch
: Logical. Set to True to render toggle-like switches instead of checkboxes.

loading_state
: Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence
: Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn’t changed from its previous value, a ‘value‘ that the user has changed while using the app will keep that change, as long as the new ‘value‘ also matches what was given originally. Used in conjunction with ‘persistence_type‘.

persisted_props
: List of a value equal to: ’value’s. Properties whose user interactions will persist after refreshing the component or the page. Since only ‘value‘ is allowed this prop can normally be ignored.

persistence_type
: A value equal to: ’local’, ’session’, ’memory’. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

name
: Character. The name of the control, which is submitted with the form data.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcCol *Col component*

Description

Component for creating Bootstrap columns to control the layout of your page. Use the width argument to specify width, or use the breakpoint arguments (xs, sm, md, lg, xl) to control the width of the columns on different screen sizes to achieve a responsive layout.

Usage

```
dbcCol(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, width=NULL, xs=NULL, sm=NULL,
md=NULL, lg=NULL, xl=NULL, xxl=NULL, align=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
width	Specify the width of the column. Behind the scenes this sets behaviour at the xs breakpoint, and will be overriden if xs is specified. Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys ’offset’, ’order’, ’size’. See the documentation for more details.
xs	Specify column behaviour on an extra small screen. Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys ’offset’, ’order’, ’size’. See the documentation for more details.
sm	Specify column behaviour on a small screen. Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys ’offset’, ’order’, ’size’. See the documentation for more details.
md	Specify column behaviour on a medium screen. Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys ’offset’, ’order’, ’size’. See the documentation for more details.
lg	Specify column behaviour on a large screen. Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys ’offset’, ’order’, ’size’. See the documentation for more details.
xl	Specify column behaviour on an extra large screen. Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys ’offset’, ’order’, ’size’. See the documentation for more details.
xxl	Specify column behaviour on an extra extra large screen. Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys ’offset’, ’order’, ’size’. See the documentation for more details.
align	A value equal to: ’start’, ’center’, ’end’, ’stretch’, ’baseline’. Set vertical alignment of this column’s content in the parent row. Options are ’start’, ’center’, ’end’, ’stretch’, ’baseline’.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcCollapse *Collapse component*

Description

Hide or show content with a vertical collapsing animation. Visibility of the children is controlled by the ‘is_open‘ prop which can be targetted by callbacks.

Usage

```
dbcCollapse(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, is_open=NULL, navbar=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component.
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
is_open	Logical. Whether collapse is currently open.
navbar	Logical. Set to True when using a collapse inside a navbar.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcContainer *Container component*

Description

Containers provide a means to center and horizontally pad your site's contents.

Usage

```
dbcContainer(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, fluid=NULL, tag=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
fluid	Logical | character. If True the container-fluid class will be applied, and the Container will expand to fill available space. A non-fluid container resizes responsively to a fixed width at the different breakpoints. You can also set the fluid property to one of the Bootstrap breakpoints: "sm", "md", "lg", "xl" or "xxl", so that the container fluidly expands to fill the screen until the specified breakpoint, then resize responsively at higher breakpoints.
tag	Character. HTML tag to apply the container class to, default: div
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcDropdownMenu *DropdownMenu component*

Description

DropdownMenu creates an overlay useful for grouping together links and other content to organise navigation or other interactive elements.

Usage

```
dbcDropdownMenu(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, label=NULL, direction=NULL,
align_end=NULL, right=NULL, in_navbar=NULL, addon_type=NULL,
disabled=NULL, nav=NULL, caret=NULL, color=NULL,
menu_variant=NULL, toggle_style=NULL,
toggle_class_name=NULL, toggleClassName=NULL, size=NULL,
loading_state=NULL, group=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component.
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
label	Character. Label for the DropdownMenu toggle.
direction	A value equal to: ’down’, ’start’, ’end’, ’up’, ’left’, ’right’, ’end’. Direction in which to expand the DropdownMenu. Default: ’down’. ‘left‘ and ‘right‘ have been deprecated, and ‘start‘ and ‘end‘ should be used instead.
align_end	Logical. Align the DropdownMenu along the right side of its parent. Default: False.
right	Logical. **DEPRECATED** Use ‘align_end‘ instead. Align the DropdownMenu along the right side of its parent. Default: False.
in_navbar	Logical. Set this to True if the DropdownMenu is inside a navbar. Default: False.
addon_type	Logical | a value equal to: ’prepend’, ’append’. Set this to ’prepend’ or ’append’ if the DropdownMenu is being used in an input group.

disabled	Logical. Disable the dropdown.
nav	Logical. Set this to True if the DropdownMenu is inside a nav for styling consistent with other nav items. Default: False.
caret	Logical. Add a caret to the DropdownMenu toggle. Default: True.
color	Character. Set the color of the DropdownMenu toggle. Available options are: 'primary', 'secondary', 'success', 'warning', 'danger', 'info', 'link' or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name) Default: 'secondary'
menu_variant	A value equal to: 'light', 'dark'. Set 'menu_variant="dark"' to create a dark-mode drop down instead
toggle_style	Named list. Defines CSS styles which will override styles previously set. The styles set here apply to the DropdownMenu toggle.
toggle_class_name	Character. Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the DropdownMenu toggle.
toggleClassName	Character. **DEPRECATED** Use 'toggle_class_name' instead. Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the DropdownMenu toggle.
size	A value equal to: 'sm', 'md', 'lg'. Size of the DropdownMenu. 'sm' corresponds to small, 'md' to medium and 'lg' to large.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
group	Logical. Set group to True if the DropdownMenu is inside a ButtonGroup.

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcDropdownMenuItem` *DropdownMenuItem component*

Description

Use DropdownMenuItem to build up the content of a DropdownMenu.

Usage

```
dbcDropdownMenuItem(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, active=NULL, disabled=NULL,
divider=NULL, header=NULL, href=NULL, toggle=NULL,
external_link=NULL, n_clicks=NULL, n_clicks_timestamp=NULL,
loading_state=NULL, target=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component.
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
active	Logical. Style this item as 'active'.
disabled	Logical. Style this item as 'disabled'.
divider	Logical. Set to True if this entry is a divider. Typically, it will have no children.
header	Logical. Set to True if this is a header, rather than a conventional menu item.
href	Character. Pass a URL (relative or absolute) to make the menu entry a link.
toggle	Logical. Whether to toggle the DropdownMenu on click. Default: True.
external_link	Logical. If true, the browser will treat this as an external link, forcing a page refresh at the new location. If false, this just changes the location without triggering a page refresh. Use this if you are observing dcc.Location, for instance. Defaults to true for absolute URLs and false otherwise.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
target	Character. Target attribute to pass on to the link. Only applies to external links.

Value

named list of JSON elements corresponding to React.js properties and their values

dbcFade *Fade component*

Description

Hide or show content with a fading animation. Visibility of the children is controlled by the 'is_open' prop which can be targetted by callbacks.

Usage

```
dbcFade(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, is_in=NULL, timeout=NULL,
appear=NULL, enter=NULL, exit=NULL, tag=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
is_in	Logical. Controls whether the children of the Fade component are currently visible or not.
timeout	Numeric | lists containing elements 'enter', 'exit'. those elements have the following types: - enter (numeric; optional) - exit (numeric; optional). The duration of the transition, in milliseconds. You may specify a single timeout for all transitions like: 'timeout=500' or individually like: timeout='enter': 300, 'exit': 500
appear	Logical. Show fade-in animation on initial page load. Default: True.
enter	Logical. Enable or disable enter transitions. Default: True.
exit	Logical. Enable or disable exit transitions. Default: True.
tag	Character. HTML tag to use for the fade component. Default: div.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcForm` *Form component*

Description

The Form component can be used to organise collections of input components and apply consistent styling.

Usage

```
dbcForm(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, action=NULL, method=NULL,
n_submit=NULL, n_submit_timestamp=NULL,
prevent_default_on_submit=NULL, loading_state=NULL)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`style`	Named list. Defines CSS styles which will override styles previously set.
`class_name`	Character. Often used with CSS to style elements with common properties.
`className`	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`action`	Character. The URI of a program that processes the information submitted via the form.
`method`	A value equal to: ’get’, ’post’. Defines which HTTP method to use when submitting the form. Can be GET (default) or POST.
`n_submit`	Numeric. Number of times the ‘Enter‘ key was pressed while the input had focus.
`n_submit_timestamp`	Numeric. Last time that ‘Enter‘ was pressed.
`prevent_default_on_submit`	Logical. The form calls preventDefault on submit events. If you want form data to be posted to the endpoint specified by ‘action‘ on submit events, set prevent_default_on_submit to False. Defaults to True.

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcFormFeedback` *FormFeedback component*

Description

The FormFeedback component can be used to provide feedback on input values in a form. Add the form feedback to your layout and set the 'valid' or 'invalid' props of the associated input to toggle visibility.

Usage

```
dbcFormFeedback(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, type=NULL, tooltip=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
type	Character. Either 'valid' or 'invalid'.
tooltip	Logical. Use styled tooltips to display validation feedback.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcFormFloating` *FormFloating component*

Description

A component for adding float labels to form controls in forms.

Usage

```
dbcFormFloating(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, html_for=NULL, loading_state=NULL)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`style`	Named list. Defines CSS styles which will override styles previously set.
`class_name`	Character. Often used with CSS to style elements with common properties.
`className`	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`html_for`	Character. Set the ‘for‘ attribute of the label to bind it to a particular element
`loading_state`	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcFormText *FormText component*

Description

Add explanatory text below your input components.

Usage

```
dbcFormText(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, color=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
color	Character. Text color, options: primary, secondary, success, warning, danger, info, muted, light, dark, body, white, black-50, white-50 or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name).
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcIcons *dbcIcons*

Description

A list of contextually colored icon styles that can be added to Dash Bootstrap Components.

Usage

```
dbcIcons
```

Format

An object of class `list` of length 2.

dbcInput *Input component*

Description

A basic HTML input control for entering text, numbers, or passwords, with Bootstrap styles automatically applied. This component is much like its counterpart in dash_core_components, but with a few additions such as the 'valid' and 'invalid' props for providing user feedback. Note that checkbox and radio types are supported through the Checklist and RadioItems component. Dates, times, and file uploads are supported through separate components in other libraries.

Usage

```
dbcInput(id=NULL, style=NULL, class_name=NULL, className=NULL,
key=NULL, type=NULL, value=NULL, disabled=NULL,
autocomplete=NULL, autoComplete=NULL, autofocus=NULL,
autoFocus=NULL, inputmode=NULL, inputMode=NULL, list=NULL,
max=NULL, maxlength=NULL, maxLength=NULL, min=NULL,
minlength=NULL, minLength=NULL, step=NULL, html_size=NULL,
size=NULL, valid=NULL, invalid=NULL, required=NULL,
plaintext=NULL, placeholder=NULL, name=NULL, pattern=NULL,
n_submit=NULL, n_submit_timestamp=NULL, n_blur=NULL,
n_blur_timestamp=NULL, debounce=NULL, loading_state=NULL,
persistence=NULL, persisted_props=NULL,
persistence_type=NULL, tabindex=NULL, tabIndex=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
type	A value equal to: "text", ’number’, ’password’, ’email’, ’range’, ’search’, ’tel’, ’url’, ’hidden’. The type of control to render
value	Character | numeric. The value of the Input
disabled	Logical. Set to True to disable the Input.
autocomplete	Character. This attribute indicates whether the value of the control can be automatically completed by the browser.
autoComplete	Character. **DEPRECATED** Use ‘autocomplete‘ instead. This attribute indicates whether the value of the control can be automatically completed by the browser.
autofocus	A value equal to: ’autofocus’, ’autofocus’, ’autofocus’ | logical. The element should be automatically focused after the page loaded. autoFocus is an HTML boolean attribute - it is enabled by a boolean or ’autoFocus’. Alternative capitalizations ‘autofocus‘ & ‘AUTOFOCUS‘ are also acccepted.
autoFocus	A value equal to: ’autofocus’, ’autofocus’, ’autofocus’ | logical. **DEPRECATED** Use ‘autofocus‘ instead. The element should be automatically focused after the page loaded. autoFocus is an HTML boolean attribute - it is enabled by a boolean or ’autoFocus’. Alternative capitalizations ‘autofocus‘ & ‘AUTOFOCUS‘ are also acccepted.
inputmode	A value equal to: "verbatim", "latin", "latin-name", "latin-prose", "full-width-latin", "kana", "katakana", "numeric", "tel", "email", "url". Provides a hint to the browser as to the type of data that might be entered by the user while editing the element or its contents.
inputMode	A value equal to: "verbatim", "latin", "latin-name", "latin-prose", "full-width-latin", "kana", "katakana", "numeric", "tel", "email", "url". **DEPRECATED** Use ‘inputmode‘ instead. Provides a hint to the browser as to the type of data that might be entered by the user while editing the element or its contents.
list	Character. Identifies a list of pre-defined options to suggest to the user. The value must be the id of a <datalist> element in the same document. The browser displays only options that are valid values for this input element. This attribute is ignored when the type attribute’s value is hidden, checkbox, radio, file, or a button type.

`max` Character | numeric. The maximum (numeric or date-time) value for this item, which must not be less than its minimum (min attribute) value.

`maxlength` Character | numeric. If the value of the type attribute is text, email, search, password, tel, or url, this attribute specifies the maximum number of characters (in UTF-16 code units) that the user can enter. For other control types, it is ignored. It can exceed the value of the size attribute. If it is not specified, the user can enter an unlimited number of characters. Specifying a negative number results in the default behavior (i.e. the user can enter an unlimited number of characters). The constraint is evaluated only when the value of the attribute has been changed.

`maxLength` Character | numeric. **DEPRECATED** Use ‘maxlength‘ instead.

If the value of the type attribute is text, email, search, password, tel, or url, this attribute specifies the maximum number of characters (in UTF-16 code units) that the user can enter. For other control types, it is ignored. It can exceed the value of the size attribute. If it is not specified, the user can enter an unlimited number of characters. Specifying a negative number results in the default behavior (i.e. the user can enter an unlimited number of characters). The constraint is evaluated only when the value of the attribute has been changed.

`min` Character | numeric. The minimum (numeric or date-time) value for this item, which must not be greater than its maximum (max attribute) value.

`minlength` Character | numeric. If the value of the type attribute is text, email, search, password, tel, or url, this attribute specifies the minimum number of characters (in Unicode code points) that the user can enter. For other control types, it is ignored.

`minLength` Character | numeric. **DEPRECATED** Use ‘minlength‘ instead.

If the value of the type attribute is text, email, search, password, tel, or url, this attribute specifies the minimum number of characters (in Unicode code points) that the user can enter. For other control types, it is ignored.

`step` Character | numeric. Works with the min and max attributes to limit the increments at which a numeric or date-time value can be set. It can be the string any or a positive floating point number. If this attribute is not set to any, the control accepts only values at multiples of the step value greater than the minimum.

`html_size` Character. The initial size of the control. This value is in pixels unless the value of the type attribute is text or password, in which case it is an integer number of characters. This attribute applies only when the type attribute is set to text, search, tel, url, email, or password, otherwise it is ignored. In addition, the size must be greater than zero. If you do not specify a size, a default value of 20 is used.

`size` Character. Set the size of the Input. Options: ’sm’ (small), ’md’ (medium) or ’lg’ (large). Default is ’md’.

`valid` Logical. Apply valid style to the Input for feedback purposes. This will cause any FormFeedback in the enclosing div with valid=True to display.

`invalid` Logical. Apply invalid style to the Input for feedback purposes. This will cause any FormFeedback in the enclosing div with valid=False to display.

required — A value equal to: 'required', 'required' | logical. This attribute specifies that the user must fill in a value before submitting a form. It cannot be used when the type attribute is hidden, image, or a button type (submit, reset, or button). The :optional and :required CSS pseudo-classes will be applied to the field as appropriate. required is an HTML boolean attribute - it is enabled by a boolean or 'required'. Alternative capitalizations 'REQUIRED' are also acccepted.

plaintext — Logical. Set to true for a readonly input styled as plain text with the default form field styling removed and the correct margins and padding preserved.

placeholder — Character | numeric. A hint to the user of what can be entered in the control . The placeholder text must not contain carriage returns or line-feeds. Note: Do not use the placeholder attribute instead of a <label> element, their purposes are different. The <label> attribute describes the role of the form element (i.e. it indicates what kind of information is expected), and the placeholder attribute is a hint about the format that the content should take. There are cases in which the placeholder attribute is never displayed to the user, so the form must be understandable without it.

name — Character. The name of the control, which is submitted with the form data.

pattern — Character. A regular expression that the control's value is checked against. The pattern must match the entire value, not just some subset. Use the title attribute to describe the pattern to help the user. This attribute applies when the value of the type attribute is text, search, tel, url, email, or password, otherwise it is ignored. The regular expression language is the same as JavaScript RegExp algorithm, with the 'u' parameter that makes it treat the pattern as a sequence of unicode code points. The pattern is not surrounded by forward slashes.

n_submit — Numeric. Number of times the 'Enter' key was pressed while the input had focus.

n_submit_timestamp — Numeric. Last time that 'Enter' was pressed.

n_blur — Numeric. Number of times the input lost focus.

n_blur_timestamp — Numeric. Last time the input lost focus.

debounce — Logical. If true, changes to input will be sent back to the Dash server only when the enter key is pressed or when the component loses focus. If it's false, it will sent the value back on every change.

loading_state — Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence — Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If 'persisted' is truthy and hasn't changed from its previous value, a 'value' that the user has changed while using the app will keep that change, as long as the new 'value' also matches what was given originally. Used in conjunction with 'persistence_type'.

persisted_props
: List of a value equal to: 'value's. Properties whose user interactions will persist after refreshing the component or the page. Since only 'value' is allowed this prop can normally be ignored.

persistence_type
: A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

tabindex
: Character. Overrides the browser's default tab order and follows the one specified instead.

tabIndex
: Character. **DEPRECATED** Use 'tabindex' instead.
Overrides the browser's default tab order and follows the one specified instead.

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcInputGroup` *InputGroup component*

Description

A component for grouping together inputs and buttons, dropdowns or text.

Usage

```
dbcInputGroup(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, size=NULL, loading_state=NULL)
```

Arguments

children
: A list of or a singular dash component, string or number. The children of this component.

id
: Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

style
: Named list. Defines CSS styles which will override styles previously set.

class_name
: Character. Often used with CSS to style elements with common properties.

className
: Character. **DEPRECATED** Use 'class_name' instead.
Often used with CSS to style elements with common properties.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

size
: Character. Set the size of the Input. Options: 'sm' (small), 'md' (medium) or 'lg' (large). Default is 'md'.

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcInputGroupText` *InputGroupText component*

Description

Use for wrapping text in InputGroups.

Usage

```
dbcInputGroupText(children=NULL, id=NULL, style=NULL, key=NULL,
class_name=NULL, className=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component.
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcLabel *Label component*

Description

A component for adding labels to inputs in forms with added sizing controls.

Usage

```
dbcLabel(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, hidden=NULL, size=NULL,
html_for=NULL, check=NULL, width=NULL, xs=NULL, sm=NULL,
md=NULL, lg=NULL, xl=NULL, align=NULL, color=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
hidden	Logical. Hide label from UI, but allow it to be discovered by screen-readers.
size	Character. Set size of label. Options ’sm’, ’md’ (default) or ’lg’.
html_for	Character. Set the ‘for‘ attribute of the label to bind it to a particular element
check	Logical. Set to True when using to label a Checkbox or RadioButton.
width	Specify width of label for use in grid layouts. Accepts the same values as the Col component.
xs	Specify label width on extra small screen Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys ’offset’, ’order’, ’size’. See the documentation for more details.
sm	Specify label width on a small screen Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys ’offset’, ’order’, ’size’. See the documentation for more details.
md	Specify label width on a medium screen Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys ’offset’, ’order’, ’size’. See the documentation for more details.

lg	Specify label width on a large screen Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys 'offset', 'order', 'size'. See the documentation for more details.
xl	Specify label width on an extra large screen Valid arguments are boolean, an integer in the range 1-12 inclusive, or a dictionary with keys 'offset', 'order', 'size'. See the documentation for more details.
align	A value equal to: 'start', 'center', 'end'. Set vertical alignment of the label, options: 'start', 'center', 'end', default: 'center'
color	Character. Text color, options: primary, secondary, success, warning, danger, info, muted, light, dark, body, white, black-50, white-50 or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name).
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcListGroup` *ListGroup component*

Description

Bootstrap list groups are a flexible way to display a series of content. Use in conjunction with 'ListGroupItem', 'ListGroupItemHeading' and 'ListGroupItemText'.

Usage

```
dbcListGroup(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, tag=NULL, flush=NULL,
loading_state=NULL, horizontal=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.

`className`	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`tag`	Character. HTML tag to use for the list, default: ul
`flush`	Logical. When True the ‘list-group-flush‘ class is applied which removes some borders and rounded corners from the list group in order that they can be rendered edge-to-edge in the parent container (e.g. a Card).
`loading_state`	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
`horizontal`	Logical | character. Set to True for a horizontal ListGroup, or supply one of the breakpoints as a string for a ListGroup that is horizontal at that breakpoint and up. Note that horizontal ListGroups cannot be combined with flush ListGroups, so if flush is True then horizontal has no effect.

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcListGroupItem` *ListGroupItem component*

Description

Create a single item in a ‘ListGroup‘.

Usage

```
dbcListGroupItem(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, tag=NULL, active=NULL,
disabled=NULL, color=NULL, action=NULL, href=NULL,
external_link=NULL, n_clicks=NULL, n_clicks_timestamp=NULL,
loading_state=NULL, target=NULL)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
tag	Character. HTML tag to use for the listgroupitem, default: li
active	Logical. Apply active style to item
disabled	Logical. Apply disabled style to item
color	Character. Item color, options: primary, secondary, success, info, warning, danger, or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name) default: secondary
action	Logical. Apply list-group-item-action class for hover animation etc.
href	Character. Pass a URL (relative or absolute) to make the list group item a link.
external_link	Logical. If true, the browser will treat this as an external link, forcing a page refresh at the new location. If false, this just changes the location without triggering a page refresh. Use this if you are observing dcc.Location, for instance. Defaults to true for absolute URLs and false otherwise.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
target	Character. Target attribute to pass on to the link. Only applies to external links.

Value

named list of JSON elements corresponding to React.js properties and their values

dbcModal *Modal component*

Description

Create a toggleable dialog using the Modal component. Toggle the visibility with the ‘is_open‘ prop.

Usage

```
dbcModal(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, tag=NULL, is_open=NULL, centered=NULL,
scrollable=NULL, autofocus=NULL, autoFocus=NULL, size=NULL,
role=NULL, labelledby=NULL, labelledBy=NULL, keyboard=NULL,
backdrop=NULL, modal_class_name=NULL, modalClassName=NULL,
backdrop_class_name=NULL, backdropClassName=NULL,
content_class_name=NULL, contentClassName=NULL, fade=NULL,
fullscreen=NULL, zindex=NULL, zIndex=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
tag	Character. HTML tag to use for the Modal, default: div
is_open	Logical. Whether modal is currently open.
centered	Logical. If true, vertically center modal on page.
scrollable	Logical. It true, scroll the modal body rather than the entire modal when it is too long to all fit on the screen.
autofocus	Logical. Puts the focus on the modal when initialized.
autoFocus	Logical. **DEPRECATED** Use ‘autofocus‘ instead Puts the focus on the modal when initialized.
size	Character. Set the size of the modal. Options sm, lg, xl for small, large or extra large sized modals, or leave undefined for default size.
role	Character. The ARIA role attribute.
labelledby	Character. The ARIA labelledby attribute
labelledBy	Character. **DEPRECATED** Use ‘labelledby‘ instead The ARIA labelledby attribute

keyboard Logical. Close the modal when escape key is pressed.

backdrop Logical | a value equal to: 'static'. Includes a modal-backdrop element. Alternatively, specify 'static' for a backdrop which doesn't close the modal on click.

modal_class_name
Character. CSS class to apply to the modal.

modalClassName Character. **DEPRECATED** Use ‘modal_class_name‘ instead
CSS class to apply to the modal.

backdrop_class_name
Character. CSS class to apply to the backdrop.

backdropClassName
Character. **DEPRECATED** Use ‘backdrop_class_name‘ instead
CSS class to apply to the backdrop.

content_class_name
Character. CSS class to apply to the modal content.

contentClassName
Character. **DEPRECATED** Use ‘content_class_name‘ instead
CSS class to apply to the modal content.

fade Logical. Set to false for a modal that simply appears rather than fades into view.

fullscreen A value equal to: proptypes.bool, proptypes.oneof(['sm-down', 'md-down', 'lg-down', 'xl-down', 'xxl-down']). Renders a fullscreen modal. Specifying a breakpoint will render the modal as fullscreen below the breakpoint size.

zindex Numeric | character. Set the z-index of the modal. Default 1050.

zIndex Numeric | character. **DEPRECATED** Use ‘zindex‘ instead
Set the z-index of the modal. Default 1050.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcModalBody *ModalBody component*

Description

Use this component to add consistent padding to the body (main content) of your Modals.

Usage

```
dbcModalBody(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, tag=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
tag	Character. HTML tag to use for the ModalBody, default: div
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcModalFooter *ModalFooter component*

Description

Add a footer to any modal.

Usage

```
dbcModalFooter(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, tag=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.

tag	Character. HTML tag to use for the ModalFooter, default: div
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcModalHeader *ModalHeader component*

Description

Add a header to any modal.

Usage

```
dbcModalHeader(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, close_button=NULL, tag=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
close_button	Logical. Add a close button to the header that can be used to close the modal.
tag	Character. HTML tag to use for the ModalHeader, default: div
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcModalTitle *ModalTitle component*

Description

Add a title to any modal. Should be used as a child of the ModalHeader.

Usage

```
dbcModalTitle(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, tag=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
tag	Character. HTML tag to use for the ModalTitle, default: div
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcNav *Nav component*

Description

Nav can be used to group together a collection of navigation links.

Usage

```
dbcNav(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, pills=NULL, card=NULL, fill=NULL,
justified=NULL, vertical=NULL, horizontal=NULL, navbar=NULL,
navbar_scroll=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
pills	Logical. Apply pill styling to nav items. Active items will be indicated by a pill.
card	Logical. Set to True when using Nav with pills styling inside a CardHeader.
fill	Logical. Expand the nav items to fill available horizontal space.
justified	Logical. Expand the nav items to fill available horizontal space, making sure every nav item has the same width.
vertical	Logical | character. Stack NavItems vertically. Set to True for a vertical Nav on all screen sizes, or pass one of the Bootstrap breakpoints (’xs’, ’sm’, ’md’, ’lg’, ’xl’) for a Nav which is vertical at that breakpoint and above, and horizontal on smaller screens.
horizontal	A value equal to: ’start’, ’center’, ’end’, ’between’, ’around’. Specify the horizontal alignment of the NavItems. Options are ’start’, ’center’, or ’end’.
navbar	Logical. Set to True if using Nav in Navbar component. This applies the ‘navbar-nav‘ class to the Nav which uses more lightweight styles to match the parent Navbar better.
navbar_scroll	Logical. Enable vertical scrolling within the toggleable contents of a collapsed Navbar.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcNavbar *Navbar component*

Description

The Navbar component can be used to make fully customisable navbars.

Usage

```
dbcNavbar(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, light=NULL, dark=NULL, fixed=NULL,
sticky=NULL, color=NULL, role=NULL, tag=NULL, expand=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
light	Logical. Applies the ‘navbar-light‘ class to the Navbar, causing text in the children of the Navbar to use dark colors for contrast / visibility.
dark	Logical. Applies the ‘navbar-dark‘ class to the Navbar, causing text in the children of the Navbar to use light colors for contrast / visibility.
fixed	Character. Fix the navbar’s position at the top or bottom of the page, options: top, bottom
sticky	A value equal to: ’top’. Position the navbar at the top of the viewport, but only after scrolling past it. A convenience prop for the sticky-top positioning class. Not supported in <= IE11 and other older browsers With ‘sticky‘, the navbar remains in the viewport when you scroll. By contrast, with ‘fixed‘, the navbar will remain at the top or bottom of the page. sticky=’top’
color	Character. Sets the color of the Navbar. Main options are primary, light and dark, default light. You can also choose one of the other contextual classes provided by Bootstrap (secondary, success, warning, danger, info, white) or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name)

role	Character. The ARIA role attribute.
tag	Character. HTML tag to use for the Navbar, default 'nav'
expand	Logical | character. Specify screen size at which to expand the menu bar, e.g. sm, md, lg etc.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcNavbarBrand *NavbarBrand component*

Description

Call out attention to a brand name or site title within a navbar.

Usage

```
dbcNavbarBrand(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, external_link=NULL, href=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
external_link	Logical. If true, the browser will treat this as an external link, forcing a page refresh at the new location. If false, this just changes the location without triggering a page refresh. Use this if you are observing dcc.Location, for instance. Defaults to true for absolute URLs and false otherwise.

href	Character. URL of the linked resource
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcNavbarSimple *NavbarSimple component*

Description

A self-contained navbar ready for use. If you need more customisability try 'Navbar' instead.

Usage

```
dbcNavbarSimple(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, brand=NULL, brand_href=NULL,
brand_style=NULL, brand_external_link=NULL, fluid=NULL,
links_left=NULL, light=NULL, dark=NULL, fixed=NULL,
sticky=NULL, color=NULL, expand=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
brand	Character. Brand text, to go top left of the navbar.
brand_href	Character. Link to attach to brand.
brand_style	Named list. CSS style options for brand.

brand_external_link
: Logical. If true, the browser will treat the brand link as external, forcing a page refresh at the new location. If false, this just changes the location without triggering a page refresh. Use this if you are observing dcc.Location, for instance. Defaults to true for absolute URLs and false otherwise.

fluid
: Logical. The contents of the Navbar are wrapped in a container, use fluid=True to make this container fluid, so that in particular, the contents of the navbar fill the available horizontal space.

links_left
: Logical. Align the navlinks in the navbar to the left. Default: False.

light
: Logical. Applies the 'navbar-light' class to the NavbarSimple, causing text in the children of the Navbar to use dark colors for contrast / visibility.

dark
: Logical. Applies the 'navbar-dark' class to the NavbarSimple, causing text in the children of the Navbar to use light colors for contrast / visibility.

fixed
: Character. Fix the navbar's position at the top or bottom of the page, options: top, bottom

sticky
: Character. Stick the navbar to the top or the bottom of the viewport, options: top, bottom

  With 'sticky', the navbar remains in the viewport when you scroll. By contrast, with 'fixed', the navbar will remain at the top or bottom of the page.

color
: Character. Sets the color of the NavbarSimple. Main options are primary, light and dark, default light.

  You can also choose one of the other contextual classes provided by Bootstrap (secondary, success, warning, danger, info, white) or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name)

expand
: Logical | character. Specify breakpoint at which to expand the menu bar. Options are: 'xs', 'sm', 'md', 'lg', or 'xl'. Below this breakpoint the navbar will collapse and navitems will be placed in a togglable collapse element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcNavbarToggler *NavbarToggler component*

Description

Use this component to create a navbar toggle to show navlinks when the navbar collapses on smaller screens.

Usage

```
dbcNavbarToggler(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, type=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
type	Character. Toggle type, default: button.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcNavItem *NavItem component*

Description

Create a single item in a ‘Nav‘.

Usage

```
dbcNavItem(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcNavLink *NavLink component*

Description

Add a link to a ‘Nav‘. Can be used as a child of ‘NavItem‘ or of ‘Nav‘ directly.

Usage

```
dbcNavLink(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, href=NULL, active=NULL,
disabled=NULL, external_link=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, loading_state=NULL, target=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
href	Character. The URL of the linked resource.
active	Logical	a value equal to: ’partial’, ’exact’. Apply ’active’ style to this component. Set to "exact" to automatically toggle active status when the current pathname matches href, or to "partial" to automatically toggle on a partial match. Assumes that href is a relative url such as /link rather than an absolute such as https://example.com/link For example - dbc.NavLink(..., href="/my-page", active="exact") will be active on "/my-page" but not "/my-page/other" or "/random" - dbc.NavLink(..., href="/my-page", active="partial") will be active on "/my-page" and "/my-page/other" but not "/random"
disabled	Logical. Disable the link
external_link	Logical. If true, the browser will treat this as an external link, forcing a page refresh at the new location. If false, this just changes the location without triggering a page refresh. Use this if you are observing dcc.Location, for instance. Defaults to true for absolute URLs and false otherwise.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
target	Character. Target attribute to pass on to the link. Only applies to external links.

Value

named list of JSON elements corresponding to React.js properties and their values

dbcOffcanvas *Offcanvas component*

Description

Create a toggleable hidden sidebar using the Offcanvas component. Toggle the visibility with the ‘is_open‘ prop.

Usage

```
dbcOffcanvas(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, labelledby=NULL, labelledBy=NULL,
backdrop=NULL, backdrop_class_name=NULL,
backdropClassName=NULL, keyboard=NULL, is_open=NULL,
placement=NULL, scrollable=NULL, autofocus=NULL,
autoFocus=NULL, title=NULL, close_button=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** - Use class_name instead. Often used with CSS to style elements with common properties.
labelledby	Character. The ARIA labelledby attribute
labelledBy	Character. **DEPRECATED** Use ‘labelledby‘ instead The ARIA labelledby attribute
backdrop	Logical | a value equal to: ’static’. Includes an offcanvas-backdrop element. Alternatively, specify ’static’ for a backdrop which doesn’t close the modal on click.
backdrop_class_name	Character. CSS class to apply to the backdrop.
backdropClassName	Character. **DEPRECATED** - Use backdrop_class_name instead. CSS class to apply to the backdrop.
keyboard	Logical. Close the offcanvas when escape key is pressed.
is_open	Logical. Whether offcanvas is currently open.
placement	A value equal to: ’start’, ’end’, ’top’, ’bottom’. Which side of the viewport the offcanvas will appear from.

scrollable	Logical. Allow body scrolling while offcanvas is open.
autofocus	Logical. Puts the focus on the offcanvas when initialized.
autoFocus	Logical. **DEPRECATED** Use ‘autofocus‘ instead Puts the focus on the modal when initialized.
title	Character. The header title
close_button	Logical. Specify whether the Component should contain a close button in the header
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcPagination *Pagination component*

Description

The container for presentational components for building a pagination UI. Individual pages should be added as children using the ‘PaginationItem‘ component.

Usage

```
dbcPagination(id=NULL, class_name=NULL, className=NULL, style=NULL,
size=NULL, min_value=NULL, max_value=NULL, step=NULL,
active_page=NULL, fully_expanded=NULL, previous_next=NULL,
first_last=NULL, loading_state=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** - Use class_name instead. Often used with CSS to style elements with common properties.
style	Named list. Defines CSS styles which will override styles previously set.
size	A value equal to: ’sm’, ’lg’. Set the size of all page items in the pagination.
min_value	Numeric. Minimum (leftmost) value to appear in the pagination.

max_value	Numeric. Maximum (rightmost) value to appear in the pagination. Must be defined. If the ‘min_value‘ and ‘step‘ together cannot reach this value, then the next stepped value is used as the maximum.
step	Numeric. Page increment step.
active_page	Numeric. The currently active page
fully_expanded	Logical. When True, this will display all numbers between ‘min_value‘ and ‘max_value‘.
previous_next	Logical. When True, this will display a previous and next icon before and after the individual page numbers.
first_last	Logical. When True, this will display a first and last icon at the beginning and end of the component.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcPopover *Popover component*

Description

Popover creates a toggleable overlay that can be used to provide additional information or content to users without having to load a new page or open a new window. Use the ‘PopoverHeader‘ and ‘PopoverBody‘ components to control the layout of the children.

Usage

```
dbcPopover(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, placement=NULL, target=NULL,
trigger=NULL, is_open=NULL, hide_arrow=NULL,
inner_class_name=NULL, innerClassName=NULL, delay=NULL,
offset=NULL, flip=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
placement	A value equal to: ’auto’, ’auto-start’, ’auto-end’, ’top’, ’top-start’, ’top-end’, ’right’, ’right-start’, ’right-end’, ’bottom’, ’bottom-start’, ’bottom-end’, ’left’, ’left-start’, ’left-end’. Specify popover placement.
target	Character | named list. ID of the component to attach the popover to.
trigger	Character. Space separated list of triggers (e.g. "click hover focus legacy"). These specify ways in which the target component can toggle the popover. If not specified you must toggle the popover yourself using callbacks. Options are: - "click": toggles the popover when the target is clicked. - "hover": toggles the popover when the target is hovered over with the cursor. - "focus": toggles the popover when the target receives focus - "legacy": toggles the popover when the target is clicked, but will also dismiss the popover when the user clicks outside of the popover.
is_open	Logical. Whether the Popover is open or not.
hide_arrow	Logical. Hide popover arrow.
inner_class_name	Character. CSS class to apply to the popover.
innerClassName	Character. **DEPRECATED** Use ‘inner_class_name‘ instead CSS class to apply to the popover.
delay	Lists containing elements ’show’, ’hide’. those elements have the following types: - show (numeric; optional) - hide (numeric; optional) | numeric. Optionally override show/hide delays
offset	Character | numeric. Offset of the popover relative to its target
flip	Logical. Whether to flip the direction of the popover if too close to the container edge, default True.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcPopoverBody *PopoverBody component*

Description

Componnet for wrapping the body (i.e. main content) of a 'Popover'.

Usage

```
dbcPopoverBody(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, tag=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
tag	Character. HTML tag to use for the PopoverBody, default: div
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcPopoverHeader *PopoverHeader component*

Description

Creates a header for use inside the ‘Popover‘ component.

Usage

```
dbcPopoverHeader(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, tag=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
tag	Character. HTML tag to use for the PopoverHeader, default: h3
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dbcProgress *Progress component*

Description

A component for creating progress bars just with CSS. Control the current progress with a callback and the ‘value‘ prop.

Usage

```
dbcProgress(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, bar=NULL, min=NULL, max=NULL,
value=NULL, label=NULL, hide_label=NULL, animated=NULL,
striped=NULL, color=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component. Use this to nest progress bars.
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
bar	Logical. Set to True when nesting Progress inside another Progress component to create a multi-progress bar.
min	Numeric. Upper limit for value, default: 100
max	Numeric. Upper limit for value, default: 100
value	Character | numeric. Specify progress, value from min to max inclusive.
label	Character. Adds a label to the progress bar.
hide_label	Logical. Set to True to hide the label.
animated	Logical. Animate the bar, must have striped set to True to work.
striped	Logical. Use striped progress bar
color	Character. Set color of the progress bar, options: primary, secondary, success, warning, danger, info or any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name).

loading_state Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcRadioButton *RadioButton component*

Description

Checklist is a component that encapsulates several checkboxes. The values and labels of the checklist is specified in the ‘options‘ property and the checked items are specified with the ‘value‘ property. Each checkbox is rendered as an input / label pair. ‘Checklist‘ must be given an ‘id‘ to work properly.

Usage

```
dbcRadioButton(id=NULL, class_name=NULL, className=NULL, style=NULL,
input_style=NULL, inputStyle=NULL, input_class_name=NULL,
inputClassName=NULL, label=NULL, label_id=NULL,
label_style=NULL, labelStyle=NULL, label_class_name=NULL,
labelClassName=NULL, name=NULL, value=NULL, disabled=NULL,
loading_state=NULL, persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
class_name	Character. The class of the container (div)
className	Character. **DEPRECATED** Use ‘class_name‘ instead. The class of the container (div)
style	Named list. The style of the container (div)
input_style	Named list. The style of the <input> checkbox element.
inputStyle	Named list. **DEPRECATED** Use ‘input_style‘ instead. The style of the <input> checkbox element.
input_class_name	Character. The class of the <input> checkbox element

`inputClassName` Character. **DEPRECATED** Use ‘input_class_name‘ instead.
The class of the <input> checkbox element

`label` Character. The label of the <input> element

`label_id` Character. The id of the label

`label_style` Named list. Inline style arguments to apply to the <label> element for each item.

`labelStyle` Named list. **DEPRECATED** Use ‘label_style‘ instead.
Inline style arguments to apply to the <label> element for each item.

`label_class_name`
Character. CSS classes to apply to the <label> element for each item.

`labelClassName` Character. **DEPRECATED** Use ‘label_class_name‘ instead.
CSS classes to apply to the <label> element for each item.

`name` Character. The name of the control, which is submitted with the form data.

`value` Logical. The value of the input.

`disabled` Logical. Disable the RadioButton.

`loading_state` Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

`persistence` Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn’t changed from its previous value, a ‘value‘ that the user has changed while using the app will keep that change, as long as the new ‘value‘ also matches what was given originally. Used in conjunction with ‘persistence_type‘.

`persisted_props`
List of a value equal to: ’value’s. Properties whose user interactions will persist after refreshing the component or the page. Since only ‘value‘ is allowed this prop can normally be ignored.

`persistence_type`
A value equal to: ’local’, ’session’, ’memory’. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

dbcRadioItems *RadioItems component*

Description

RadioItems is a component that encapsulates several radio item inputs. The values and labels of the RadioItems is specified in the ‘options‘ property and the seleced item is specified with the ‘value‘ property. Each radio item is rendered as an input and associated label which are siblings of each other.

Usage

```
dbcRadioItems(id=NULL, key=NULL, options=NULL, value=NULL, style=NULL,
class_name=NULL, className=NULL, input_style=NULL,
inputStyle=NULL, input_checked_style=NULL,
inputCheckedStyle=NULL, input_class_name=NULL,
inputClassName=NULL, input_checked_class_name=NULL,
inputCheckedClassName=NULL, label_style=NULL,
labelStyle=NULL, label_checked_style=NULL,
labelCheckedStyle=NULL, label_class_name=NULL,
labelClassName=NULL, label_checked_class_name=NULL,
labelCheckedClassName=NULL, inline=NULL, switch=NULL,
loading_state=NULL, persistence=NULL, persisted_props=NULL,
persistence_type=NULL, name=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
options	List of lists containing elements 'label', 'value', 'disabled', 'input_id', 'label_id'. those elements have the following types: - label (character | numeric; required): the radio item's label - value (character | numeric; required): the value of the radio item. this value corresponds to the items specified in the ‘value‘ property. - disabled (logical; optional): if true, this radio item is disabled and can't be clicked on. - input_id (character; optional): id for this option's input, can be used to attach tooltips or apply css styles - label_id (character; optional): id for this option's label, can be used to attach tooltips or apply css styless. An array of options
value	Character | numeric. The currently selected value
style	Named list. The style of the container (div)
class_name	Character. The class of the container (div)

`className` Character. **DEPRECATED** Use ‘class_name‘ instead.
The class of the container (div)

`input_style` Named list. The style of the <input> radio element

`inputStyle` Named list. **DEPRECATED** Use ‘input_style‘ instead.
The style of the <input> radio element

`input_checked_style`
Named list. Additional inline style arguments to apply to <input> elements on checked items.

`inputCheckedStyle`
Named list. **DEPRECATED** Use ‘input_checked_style‘ instead.
Additional inline style arguments to apply to <input> elements on checked items.

`input_class_name`
Character. The class of the <input> radio element

`inputClassName` Character. **DEPRECATED** Use ‘input_class_name‘ instead.
The class of the <input> radio element

`input_checked_class_name`
Character. Additional CSS classes to apply to the <input> element when the corresponding radio is checked.

`inputCheckedClassName`
Character. **DEPRECATED** Use ‘input_checked_class_name‘ instead.
Additional CSS classes to apply to the <input> element when the corresponding radio is checked.

`label_style` Named list. Inline style arguments to apply to the <label> element for each item.

`labelStyle` Named list. **DEPRECATED** Use ‘label_style‘ instead.
Inline style arguments to apply to the <label> element for each item.

`label_checked_style`
Named list. Additional inline style arguments to apply to <label> elements on checked items.

`labelCheckedStyle`
Named list. **DEPRECATED** Use ‘label_checked_style‘ instead.
Additional inline style arguments to apply to <label> elements on checked items.

`label_class_name`
Character. CSS classes to apply to the <label> element for each item.

`labelClassName` Character. **DEPRECATED** Use ‘label_class_name‘ instead.
CSS classes to apply to the <label> element for each item.

`label_checked_class_name`
Character. Additional CSS classes to apply to the <label> element when the corresponding radio is checked.

`labelCheckedClassName`
Character. **DEPRECATED** Use ‘label_checked_class_name‘ instead.
Additional CSS classes to apply to the <label> element when the corresponding radio is checked.

`inline` Logical. Arrange RadioItems inline

switch Logical. Set to True to render toggle-like switches instead of radios.

loading_state Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If 'persisted' is truthy and hasn't changed from its previous value, a 'value' that the user has changed while using the app will keep that change, as long as the new 'value' also matches what was given originally. Used in conjunction with 'persistence_type'.

persisted_props List of a value equal to: 'value's. Properties whose user interactions will persist after refreshing the component or the page. Since only 'value' is allowed this prop can normally be ignored.

persistence_type A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

name Character. The name of the control, which is submitted with the form data.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcRow *Row component*

Description

Row is one of the core layout components in Bootstrap. Build up your layout as a series of rows of columns. Row has arguments for controlling the vertical and horizontal alignment of its children, as well as the spacing between columns.

Usage

```
dbcRow(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, align=NULL, justify=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
align	A value equal to: ’start’, ’center’, ’end’, ’stretch’, ’baseline’. Set vertical alignment of columns in this row. Options are ’start’, ’center’, ’end’, ’stretch’ and ’baseline’.
justify	A value equal to: ’start’, ’center’, ’end’, ’around’, ’between’. Set horizontal spacing and alignment of columns in this row. Options are ’start’, ’center’, ’end’, ’around’ and ’between’.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcSelect *Select component*

Description

Create a HTML select element with Bootstrap styles. Specify options as a list of dictionaries with keys label, value and disabled.

Usage

```
dbcSelect(id=NULL, style=NULL, class_name=NULL, className=NULL,
key=NULL, placeholder=NULL, value=NULL, options=NULL,
disabled=NULL, required=NULL, valid=NULL, invalid=NULL,
size=NULL, html_size=NULL, persistence=NULL,
persisted_props=NULL, persistence_type=NULL, name=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
placeholder	Character. Placeholder text to display before a selection is made.
value	Character | numeric. The value of the currently selected option.
options	List of lists containing elements ’label’, ’value’, ’disabled’, ’title’. those elements have the following types: - label (character | numeric; required): the options’s label - value (character; required): the value of the option. this value corresponds to the items specified in the ‘value‘ property. - disabled (logical; optional): if true, this checkbox is disabled and can’t be clicked on. - title (character; optional): the html ’title’ attribute for the option. allows for information on hover. for more information on this attribute, see https://developer.mozilla.org/en-us/docs/web/html/global_attributes/titles. An array of options for the select
disabled	Logical. Set to True to disable the Select.
required	A value equal to: ’required’, ’required’ | logical. This attribute specifies that the user must fill in a value before submitting a form. It cannot be used when the type attribute is hidden, image, or a button type (submit, reset, or button). The :optional and :required CSS pseudo-classes will be applied to the field as appropriate. required is an HTML boolean attribute - it is enabled by a boolean or ’required’. Alternative capitalizations ‘REQUIRED‘ are also acccepted.
valid	Logical. Apply valid style to the Input for feedback purposes. This will cause any FormFeedback in the enclosing div with valid=True to display.
invalid	Logical. Apply invalid style to the Input for feedback purposes. This will cause any FormFeedback in the enclosing div with valid=False to display.
size	Character. Set the size of the Input. Options: ’sm’ (small), ’md’ (medium) or ’lg’ (large). Default is ’md’.
html_size	Character. This represents the number of rows in the select that should be visible at one time. It will result in the Select being rendered as a scrolling list box rather than a dropdown.
persistence	Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn’t changed from its previous value, a ‘value‘ that the user has changed while using the app will keep that change, as long as the new ‘value‘ also matches what was given originally. Used in conjunction with ‘persistence_type‘.

persisted_props
: List of a value equal to: 'value's. Properties whose user interactions will persist after refreshing the component or the page. Since only 'value' is allowed this prop can normally be ignored.

persistence_type
: A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

name
: Character. The name of the control, which is submitted with the form data.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcSpinner *Spinner component*

Description

Render Bootstrap style loading spinners using only CSS. This component can be used standalone to render a loading spinner, or it can be used like 'dash_core_components.Loading' by giving it children. In the latter case the chosen spinner will display while the children are loading.

Usage

```
dbcSpinner(children=NULL, id=NULL, fullscreen_style=NULL,
spinner_style=NULL, fullscreen_class_name=NULL,
fullscreenClassName=NULL, spinner_class_name=NULL,
spinnerClassName=NULL, color=NULL, type=NULL, size=NULL,
fullscreen=NULL, delay_hide=NULL, delay_show=NULL,
show_initially=NULL)
```

Arguments

children
: A list of or a singular dash component, string or number. The children of this component.

id
: Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

fullscreen_style
: Named list. Defines CSS styles for the container when fullscreen=True.

spinner_style
: Named list. Inline CSS styles to apply to the spinner.

fullscreen_class_name
: Character. Often used with CSS to style elements with common properties.

fullscreenClassName
: Character. **DEPRECATED** - use ‘fullscreen_class_name‘ instead.
Often used with CSS to style elements with common properties.

spinner_class_name
: Character. CSS class names to apply to the spinner.

spinnerClassName
: Character. **DEPRECATED** - use ‘spinner_class_name‘ instead.
CSS class names to apply to the spinner.

color
: Character. Sets the color of the Spinner. Main options are Bootstrap contextual colors: primary, secondary, success, info, warning, danger, light, dark, body, muted, white-50, black-50. You can also specify any valid CSS color of your choice (e.g. a hex code, a decimal code or a CSS color name)
If not specified will default to text colour.

type
: Character. The type of spinner. Options ’border’ and ’grow’. Default ’border’.

size
: Character. The spinner size. Options are ’sm’, and ’md’.

fullscreen
: Logical. Boolean that determines if the loading spinner will be displayed full-screen or not.

delay_hide
: Numeric. When using the spinner as a loading spinner, add a time delay (in ms) to the spinner being removed to prevent flickering.

delay_show
: Numeric. When using the spinner as a loading spinner, add a time delay (in ms) to the spinner being shown after the loading_state is set to true.

show_initially
: Logical. Whether the Spinner should show on app start-up before the loading state has been determined. Default True.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcSwitch *Switch component*

Description

Checklist is a component that encapsulates several checkboxes. The values and labels of the checklist is specified in the ‘options‘ property and the checked items are specified with the ‘value‘ property. Each checkbox is rendered as an input / label pair. ‘Checklist‘ must be given an ‘id‘ to work properly.

Usage

```
dbcSwitch(id=NULL, class_name=NULL, className=NULL, style=NULL,
input_style=NULL, inputStyle=NULL, input_class_name=NULL,
inputClassName=NULL, label=NULL, label_id=NULL,
label_style=NULL, labelStyle=NULL, label_class_name=NULL,
labelClassName=NULL, name=NULL, value=NULL, disabled=NULL,
loading_state=NULL, persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
class_name	Character. The class of the container (div)
className	Character. **DEPRECATED** Use ‘class_name‘ instead. The class of the container (div)
style	Named list. The style of the container (div)
input_style	Named list. The style of the <input> checkbox element.
inputStyle	Named list. **DEPRECATED** Use ‘input_style‘ instead. The style of the <input> checkbox element.
input_class_name	Character. The class of the <input> checkbox element
inputClassName	Character. **DEPRECATED** Use ‘input_class_name‘ instead. The class of the <input> checkbox element
label	Character. The label of the <input> element
label_id	Character. The id of the label
label_style	Named list. Inline style arguments to apply to the <label> element for each item.
labelStyle	Named list. **DEPRECATED** Use ‘label_style‘ instead. Inline style arguments to apply to the <label> element for each item.
label_class_name	Character. CSS classes to apply to the <label> element for each item.
labelClassName	Character. **DEPRECATED** Use ‘label_class_name‘ instead. CSS classes to apply to the <label> element for each item.
name	Character. The name of the control, which is submitted with the form data.
value	Logical. The value of the input.
disabled	Logical. Disable the Switch.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
persistence	Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn’t changed from its previous value, a ‘value‘ that the user has changed while using the app will keep that change, as long as the new ‘value‘ also matches what was given originally. Used in conjunction with ‘persistence_type‘.
persisted_props	List of a value equal to: ’value’s. Properties whose user interactions will persist after refreshing the component or the page. Since only ‘value‘ is allowed this prop can normally be ignored.

persistence_type
: A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcTab *Tab component*

Description

Create a single tab. Should be used as a component of Tabs.

Usage

```
dbcTab(children=NULL, id=NULL, style=NULL, tab_style=NULL,
active_tab_style=NULL, label_style=NULL,
active_label_style=NULL, class_name=NULL, className=NULL,
tab_class_name=NULL, tabClassName=NULL,
active_tab_class_name=NULL, activeTabClassName=NULL,
label_class_name=NULL, labelClassName=NULL,
active_label_class_name=NULL, activeLabelClassName=NULL,
key=NULL, label=NULL, tab_id=NULL, disabled=NULL,
loading_state=NULL)
```

Arguments

children
: A list of or a singular dash component, string or number. The children of this component

id
: Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

style
: Named list. Defines CSS styles which will override styles previously set. The styles set here apply to the content of the Tab

tab_style
: Named list. Defines CSS styles which will override styles previously set. The styles set here apply to the NavItem in the tab.

active_tab_style
: Named list. Defines CSS styles which will override styles previously set. The styles set here apply to the NavItem in the tab when it is active.

label_style
: Named list. Defines CSS styles which will override styles previously set. The styles set here apply to the NavLink in the tab.

active_label_style
: Named list. Defines CSS styles which will override styles previously set. The styles set here apply to the NavLink in the tab when it is active

`class_name` Character. Often used with CSS to style elements with common properties.

`className` Character. **DEPRECATED** Use ‘class_name‘ instead.
Often used with CSS to style elements with common properties.

`tab_class_name` Character. Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the NavItem in the tab.

`tabClassName` Character. **DEPRECATED** Use ‘tab_class_name‘ instead
Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the NavItem in the tab.

`active_tab_class_name`
Character. Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the NavItem in the tab when it is active.

`activeTabClassName`
Character. **DEPRECATED** Use ‘active_tab_class_name‘ instead
Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the NavItem in the tab when it is active.

`label_class_name`
Character. Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the NavLink in the tab.

`labelClassName` Character. **DEPRECATED** Use ‘label_class_name‘ instead
Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the NavLink in the tab.

`active_label_class_name`
Character. Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the NavLink in the tab when it is active.

`activeLabelClassName`
Character. **DEPRECATED** Use ‘active_label_class_name‘ instead
Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the NavLink in the tab when it is active.

`key` Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

`label` Character. The tab’s label, displayed in the tab itself.

`tab_id` Character. Optional identifier for tab used for determining which tab is visible if not specified, and Tab is being used inside Tabs component, the tabId will be set to "tab-i" where i is (zero indexed) position of tab in list tabs pased to Tabs component.

`disabled` Logical. Determines if tab is disabled or not - defaults to false

`loading_state` Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcTable` *Table component*

Description

A component for applying Bootstrap styles to HTML tables. Use this as a drop-in replacement for ‘html.Table‘, or generate a table from a Pandas DataFrame using ‘dbc.Table.from_dataframe‘.

Usage

```
dbcTable(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, size=NULL, bordered=NULL,
borderless=NULL, striped=NULL, color=NULL, dark=NULL,
hover=NULL, responsive=NULL, loading_state=NULL)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`style`	Named list. Defines CSS styles which will override styles previously set.
`class_name`	Character. Often used with CSS to style elements with common properties.
`className`	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`size`	Character. Specify table size, options: ’sm’, ’md’, ’lg’
`bordered`	Logical. Apply the ‘table-bordered‘ class which adds borders on all sides of the table and cells.
`borderless`	Logical. Apply the ‘table-borderless‘ class which removes all borders from the table and cells.
`striped`	Logical. Apply the ‘table-striped‘ class which applies ’zebra striping’ to rows in the table body.
`color`	Character. Table color, options: primary, secondary, success, info, warning, danger, dark, light. Default: secondary.
`dark`	Logical. **DEPRECATED** - Use color="dark" instead. Apply the ‘table-dark‘ class for dark cell backgrounds and light text.

hover	Logical. Apply the ‘table-hover‘ class which enables a hover state on table rows within the table body.
responsive	Logical | character. Set to True or one of the breakpoints ’sm’, ’md’, ’lg’, ’xl’ to make table scroll horizontally at lower breakpoints.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcTabs *Tabs component*

Description

Create Bootstrap styled tabs. Use the ‘active_tab‘ property to set, or get get the currently active tab in a callback.

Usage

```
dbcTabs(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, active_tab=NULL,
loading_state=NULL, persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
style	Named list. Defines CSS styles which will override styles previously set.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

active_tab Character. The tab_id of the currently active tab. If tab_id has not been specified for the active tab, this will default to tab-i, where i is the index (starting from 0) of the tab.

loading_state Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If 'persisted' is truthy and hasn't changed from its previous value, a 'value' that the user has changed while using the app will keep that change, as long as the new 'value' also matches what was given originally. Used in conjunction with 'persistence_type'.

persisted_props List of a value equal to: 'active_tab's. Properties whose user interactions will persist after refreshing the component or the page. Since only 'value' is allowed this prop can normally be ignored.

persistence_type A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dbcTextarea *Textarea component*

Description

A basic HTML textarea for entering multiline text based on the corresponding component in dash-core-components

Usage

```
dbcTextarea(id=NULL, key=NULL, value=NULL, autofocus=NULL,
autoFocus=NULL, cols=NULL, disabled=NULL, form=NULL,
maxlength=NULL, maxLength=NULL, minlength=NULL,
minLength=NULL, name=NULL, placeholder=NULL, readonly=NULL,
readOnly=NULL, required=NULL, rows=NULL, wrap=NULL,
accesskey=NULL, accessKey=NULL, class_name=NULL,
className=NULL, contenteditable=NULL, contentEditable=NULL,
```

```
contextmenu=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellcheck=NULL,
spellCheck=NULL, style=NULL, tabindex=NULL, tabIndex=NULL,
title=NULL, size=NULL, valid=NULL, invalid=NULL,
n_blur=NULL, n_blur_timestamp=NULL, n_submit=NULL,
n_submit_timestamp=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, debounce=NULL, loading_state=NULL,
persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
value	Character. The value of the textarea
autofocus	Character. The element should be automatically focused after the page loaded.
autoFocus	Character. **DEPRECATED** Use ‘autofocus‘ instead The element should be automatically focused after the page loaded.
cols	Character | numeric. Defines the number of columns in a textarea.
disabled	Character | logical. Indicates whether the user can interact with the element.
form	Character. Indicates the form that is the owner of the element.
maxlength	Character | numeric. Defines the maximum number of characters allowed in the element.
maxLength	Character | numeric. **DEPRECATED** Use ‘maxlength‘ instead Defines the maximum number of characters allowed in the element.
minlength	Character | numeric. Defines the minimum number of characters allowed in the element.
minLength	Character | numeric. **DEPRECATED** Use ‘minlength‘ instead Defines the minimum number of characters allowed in the element.
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
placeholder	Character. Provides a hint to the user of what can be entered in the field.
readonly	Logical | a value equal to: ’readonly’, ’readonly’, ’readonly’. Indicates whether the element can be edited.
readOnly	Logical | a value equal to: ’readonly’, ’readonly’, ’readonly’. **DEPRECATED** Use ‘readonly‘ instead Indicates whether the element can be edited.
required	A value equal to: ’required’, ’required’ | logical. This attribute specifies that the user must fill in a value before submitting a form. It cannot be used when the type attribute is hidden, image, or a button type (submit, reset, or button).

	The :optional and :required CSS pseudo-classes will be applied to the field as appropriate. required is an HTML boolean attribute - it is enabled by a boolean or 'required'. Alternative capitalizations 'REQUIRED' are also acccepted.
rows	Character | numeric. Defines the number of rows in a text area.
wrap	Character. Indicates whether the text should be wrapped.
accesskey	Character. Defines a keyboard shortcut to activate or add focus to the element.
accessKey	Character. **DEPRECATED** Use 'accesskey' instead Defines a keyboard shortcut to activate or add focus to the element.
class_name	Character. Often used with CSS to style elements with common properties.
className	Character. **DEPRECATED** Use 'class_name' instead. Often used with CSS to style elements with common properties.
contenteditable	Character | numeric. Indicates whether the element's content is editable.
contentEditable	Character | numeric. **DEPRECATED** Use 'contenteditable' instead Indicates whether the element's content is editable.
contextmenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
contextMenu	Character. **DEPRECATED** Use 'contextmenu' instead Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	A value equal to: 'true', 'false' | logical. Defines whether the element can be dragged.
hidden	Character. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellcheck	A value equal to: 'true', 'false' | logical. Indicates whether spell checking is allowed for the element.
spellCheck	A value equal to: 'true', 'false' | logical. **DEPRECATED** Use 'spellcheck' instead Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabindex	Character | numeric. Overrides the browser's default tab order and follows the one specified instead.
tabIndex	Character | numeric. **DEPRECATED** Use 'tabindex' instead Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
size	Character. Set the size of the Textarea, valid options are 'sm', 'md', or 'lg'
valid	Logical. Apply valid style to the Textarea for feedback purposes. This will cause any FormFeedback in the enclosing div with valid=True to display.

`invalid` Logical. Apply invalid style to the Textarea for feedback purposes. This will cause any FormFeedback in the enclosing div with valid=False to display.

`n_blur` Numeric. Number of times the input lost focus.

`n_blur_timestamp`
Numeric. Last time the input lost focus.

`n_submit` Numeric. Number of times the ‘Enter‘ key was pressed while the textarea had focus.

`n_submit_timestamp`
Numeric. Last time that ‘Enter‘ was pressed.

`n_clicks` Numeric. An integer that represents the number of times that this element has been clicked on.

`n_clicks_timestamp`
Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

`debounce` Logical. If true, changes to input will be sent back to the Dash server only on enter or when losing focus. If it’s false, it will sent the value back on every change.

`loading_state` Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

`persistence` Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn’t changed from its previous value, a ‘value‘ that the user has changed while using the app will keep that change, as long as the new ‘value‘ also matches what was given originally. Used in conjunction with ‘persistence_type‘.

`persisted_props`
List of a value equal to: ’value’s. Properties whose user interactions will persist after refreshing the component or the page. Since only ‘value‘ is allowed this prop can normally be ignored.

`persistence_type`
A value equal to: ’local’, ’session’, ’memory’. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

dbcThemes *dbcThemes*

Description

Externally hosted themes that can be passed to the Dash app with `app %>% add_stylesheet()`.

Usage

```
dbcThemes
```

Format

An object of class `list` of length 27.

dbcToast *Toast component*

Description

Toasts can be used to push messages and notifactions to users. Control visibility of the toast with the ‘is_open‘ prop, or use ‘duration‘ to set a timer for auto-dismissal.

Usage

```
dbcToast(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, header_style=NULL, header_class_name=NULL,
headerClassName=NULL, body_style=NULL, body_class_name=NULL,
bodyClassName=NULL, tag=NULL, is_open=NULL, key=NULL,
header=NULL, dismissable=NULL, duration=NULL,
n_dismiss=NULL, n_dismiss_timestamp=NULL, icon=NULL,
color=NULL, loading_state=NULL)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`style`	Named list. Defines CSS styles which will override styles previously set.
`class_name`	Character. Often used with CSS to style elements with common properties.
`className`	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.

header_style Named list. Defines CSS styles which will override styles previously set. The styles set here apply to the header of the toast.

header_class_name
Character. Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the header of the toast.

headerClassName
Character. **DEPRECATED** - use ‘header_class_name‘ instead
Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the header of the toast.

body_style Named list. Defines CSS styles which will override styles previously set. The styles set here apply to the body of the toast.

body_class_name
Character. Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the body of the toast.

bodyClassName Character. **DEPRECATED** - use ‘body_class_name‘ instead.
Often used with CSS to style elements with common properties. The classes specified with this prop will be applied to the body of the toast.

tag Character. HTML tag to use for the Toast, default: div

is_open Logical. Whether Toast is currently open.

key Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

header Character. Text to populate the header with

dismissable Logical. Set to True to add a dismiss button to the header which will close the toast on click

duration Numeric. Duration in milliseconds after which the Alert dismisses itself.

n_dismiss Numeric. An integer that represents the number of times that the dismiss button has been clicked on.

n_dismiss_timestamp
Numeric. Use of *_timestamp props has been deprecated in Dash in favour of dash.callback_context. See "How do I determine which Input has changed?" in the Dash FAQs https://dash.plot.ly/faqs.
An integer that represents the time (in ms since 1970) at which n_dismiss changed. This can be used to tell which button was changed most recently.

icon Character. Add a contextually coloured icon to the header of the toast. Options are: "primary", "secondary", "success", "warning", "danger", "info", "light" or "dark".

color Character. Toast color, options: primary, secondary, success, info, warning, danger, light, dark. Default: secondary.

loading_state Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dbcTooltip` *Tooltip component*

Description

A component for adding tooltips to any element, no callbacks required! Simply add the Tooltip to you layout, and give it a target (id of a component to which the tooltip should be attached)

Usage

```
dbcTooltip(children=NULL, id=NULL, style=NULL, class_name=NULL,
className=NULL, key=NULL, target=NULL, placement=NULL,
flip=NULL, delay=NULL, loading_state=NULL)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`style`	Named list. Defines CSS styles which will override styles previously set.
`class_name`	Character. Often used with CSS to style elements with common properties.
`className`	Character. **DEPRECATED** Use ‘class_name‘ instead. Often used with CSS to style elements with common properties.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`target`	Character | named list. The id of the element to attach the tooltip to
`placement`	A value equal to: ’auto’, ’auto-start’, ’auto-end’, ’top’, ’top-start’, ’top-end’, ’right’, ’right-start’, ’right-end’, ’bottom’, ’bottom-start’, ’bottom-end’, ’left’, ’left-start’, ’left-end’. How to place the tooltip.
`flip`	Logical. Whether to flip the direction of the popover if too close to the container edge, default True.
`delay`	Lists containing elements ’show’, ’hide’. those elements have the following types: - show (numeric; optional) - hide (numeric; optional). Control the delay of hide and show events.
`loading_state`	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dccChecklist` *Checklist component*

Description

Checklist is a component that encapsulates several checkboxes. The values and labels of the checklist are specified in the ‘options‘ property and the checked items are specified with the ‘value‘ property. Each checkbox is rendered as an input with a surrounding label.

Usage

```
dccChecklist(id=NULL, options=NULL, value=NULL, className=NULL,
style=NULL, inputStyle=NULL, inputClassName=NULL,
labelStyle=NULL, labelClassName=NULL, loading_state=NULL,
persistence=NULL, persisted_props=NULL,
persistence_type=NULL, inline=NULL)
```

Arguments

`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`options`	List of character | numeric | logicals | named list | list of lists containing elements 'label', 'value', 'disabled', 'title'. those elements have the following types: - label (character | numeric | logical; required): the option's label - value (character | numeric | logical; required): the value of the option. this value corresponds to the items specified in the ‘value‘ property. - disabled (logical; optional): if true, this option is disabled and cannot be selected. - title (character; optional): the html 'title' attribute for the option. allows for information on hover. for more information on this attribute, see https://developer.mozilla.org/en-us/docs/web/html/global_attributes/titles. An array of options
`value`	List of character | numeric | logicals. The currently selected value
`className`	Character. The class of the container (div)
`style`	Named list. The style of the container (div)
`inputStyle`	Named list. The style of the <input> checkbox element
`inputClassName`	Character. The class of the <input> checkbox element
`labelStyle`	Named list. The style of the <label> that wraps the checkbox input and the option's label
`labelClassName`	Character. The class of the <label> that wraps the checkbox input and the option's label

loading_state Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If 'persisted' is truthy and hasn't changed from its previous value, a 'value' that the user has changed while using the app will keep that change, as long as the new 'value' also matches what was given originally. Used in conjunction with 'persistence_type'.

persisted_props List of a value equal to: 'value's. Properties whose user interactions will persist after refreshing the component or the page. Since only 'value' is allowed this prop can normally be ignored.

persistence_type A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

inline Logical. Indicates whether labelStyle should be inline or not True: Automatically set 'display': 'inline-block' to labelStyle False: No additional styles are passed into labelStyle.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    dccChecklist(
      id = "checklist-input",
      options=list(
        list("label" = "New York City", "value" = "NYC"),
        list("label" = "Montreal", "value" = "MTL"),
        list("label" = "San Francisco", "value" = "SF")
       ),
      value=list("MTL", "SF")
     )
  )

  app$run_server()
}
```

dccClipboard *Clipboard component*

Description

The Clipboard component copies text to the clipboard

Usage

```
dccClipboard(id=NULL, target_id=NULL, content=NULL, n_clicks=NULL,
title=NULL, style=NULL, className=NULL, loading_state=NULL)
```

Arguments

id	Character. The ID used to identify this component.
target_id	Character | named list. The id of target component containing text to copy to the clipboard. The inner text of the ‘children‘ prop will be copied to the clipboard. If none, then the text from the ‘value‘ prop will be copied.
content	Character. The text to be copied to the clipboard if the ‘target_id‘ is None.
n_clicks	Numeric. The number of times copy button was clicked
title	Character. The text shown as a tooltip when hovering over the copy icon.
style	Named list. The icon’s styles
className	Character. The class name of the icon element
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

dccConfirmDialog *ConfirmDialog component*

Description

ConfirmDialog is used to display the browser’s native "confirm" modal, with an optional message and two buttons ("OK" and "Cancel"). This ConfirmDialog can be used in conjunction with buttons when the user is performing an action that should require an extra step of verification.

Usage

```
dccConfirmDialog(id=NULL, message=NULL, submit_n_clicks=NULL,
submit_n_clicks_timestamp=NULL, cancel_n_clicks=NULL,
cancel_n_clicks_timestamp=NULL, displayed=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
message	Character. Message to show in the popup.
submit_n_clicks	Numeric. Number of times the submit button was clicked
submit_n_clicks_timestamp	Numeric. Last time the submit button was clicked.
cancel_n_clicks	Numeric. Number of times the popup was canceled.
cancel_n_clicks_timestamp	Numeric. Last time the cancel button was clicked.
displayed	Logical. Set to true to send the ConfirmDialog.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(
      htmlDiv(
        list(
          dccConfirmDialog(
            id='confirm',
            message='Danger danger! Are you sure you want to continue?'),
          dccDropdown(
        options=lapply(list('Safe', 'Danger!!'),function(x){list('label'= x, 'value'= x)}),
            id='dropdown'
          ),
          htmlDiv(id='output-confirm1')
        )
      )
    )

    app$callback(
      output = list(id = 'confirm', property = 'displayed'),
      params=list(input(id = 'dropdown', property = 'value')),
      function(value){
```

```
          if(value == 'Danger!!'){
            return(TRUE)}
          else{
            return(FALSE)}
        })

  app$run_server()
}
```

---

dccConfirmDialogProvider

ConfirmDialogProvider component

Description

A wrapper component that will display a confirmation dialog when its child component has been clicked on. For example: "‘ dcc.ConfirmDialogProvider(html.Button('click me', id='btn'), message='Danger - Are you sure you want to continue.' id='confirm') "‘

Usage

```
dccConfirmDialogProvider(children=NULL, id=NULL, message=NULL, submit_n_clicks=NULL,
submit_n_clicks_timestamp=NULL, cancel_n_clicks=NULL,
cancel_n_clicks_timestamp=NULL, displayed=NULL,
loading_state=NULL)
```

Arguments

children	Logical	numeric	character	named list	unnamed list. The children to hijack clicks from and display the popup.
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
message	Character. Message to show in the popup.
submit_n_clicks	Numeric. Number of times the submit was clicked
submit_n_clicks_timestamp	Numeric. Last time the submit button was clicked.
cancel_n_clicks	Numeric. Number of times the popup was canceled.
cancel_n_clicks_timestamp	Numeric. Last time the cancel button was clicked.
displayed	Logical. Is the modal currently displayed.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(htmlDiv(list(
      dccConfirmDialogProvider(
        children=htmlButton(
          'Click Me',
          n_clicks = 0
        ),
        id='danger-danger-provider',
        message='Danger danger! Are you sure you want to continue?',
        submit_n_clicks=NULL
      ),
      htmlDiv(id='output-provider',
              children='Click the button to submit')
    )))

    app$callback(
      output = list(id = 'output-provider', property = 'children'),
      params=list(input(id = 'danger-danger-provider', property = 'submit_n_clicks')),
      function(submit_n_clicks) {
        if (is.null(unlist(submit_n_clicks))) {
          return('')
        } else {
          paste0('That was a dangerous choice! Submitted ', submit_n_clicks, ' times.')
        }
        }

    )

  app$run_server()
}
```

---

dccDatePickerRange *DatePickerRange component*

Description

DatePickerRange is a tailor made component designed for selecting timespan across multiple days off of a calendar. The DatePicker integrates well with the Python datetime module with the startDate and endDate being returned in a string format suitable for creating datetime objects. This component is based off of Airbnb's react-dates react component which can be found here: https://github.com/airbnb/react-dates

Usage

```
dccDatePickerRange(id=NULL, start_date=NULL, start_date_id=NULL,
end_date_id=NULL, end_date=NULL, min_date_allowed=NULL,
max_date_allowed=NULL, disabled_days=NULL,
initial_visible_month=NULL,
start_date_placeholder_text=NULL,
end_date_placeholder_text=NULL, day_size=NULL,
calendar_orientation=NULL, is_RTL=NULL,
reopen_calendar_on_clear=NULL, number_of_months_shown=NULL,
with_portal=NULL, with_full_screen_portal=NULL,
first_day_of_week=NULL, minimum_nights=NULL,
stay_open_on_select=NULL, show_outside_days=NULL,
month_format=NULL, display_format=NULL, disabled=NULL,
clearable=NULL, style=NULL, className=NULL, updatemode=NULL,
loading_state=NULL, persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
start_date	Character. Specifies the starting date for the component. Accepts datetime.datetime objects or strings in the format 'YYYY-MM-DD'
start_date_id	Character. The HTML element ID of the start date input field. Not used by Dash, only by CSS.
end_date_id	Character. The HTML element ID of the end date input field. Not used by Dash, only by CSS.
end_date	Character. Specifies the ending date for the component. Accepts datetime.datetime objects or strings in the format 'YYYY-MM-DD'
min_date_allowed	Character. Specifies the lowest selectable date for the component. Accepts datetime.datetime objects or strings in the format 'YYYY-MM-DD'
max_date_allowed	Character. Specifies the highest selectable date for the component. Accepts datetime.datetime objects or strings in the format 'YYYY-MM-DD'
disabled_days	List of characters. Specifies additional days between min_date_allowed and max_date_allowed that should be disabled. Accepted datetime.datetime objects or strings in the format 'YYYY-MM-DD'
initial_visible_month	Character. Specifies the month that is initially presented when the user opens the calendar. Accepts datetime.datetime objects or strings in the format 'YYYY-MM-DD'
start_date_placeholder_text	Character. Text that will be displayed in the first input box of the date picker when no date is selected. Default value is 'Start Date'

`end_date_placeholder_text`
: Character. Text that will be displayed in the second input box of the date picker when no date is selected. Default value is 'End Date'

`day_size`
: Numeric. Size of rendered calendar days, higher number means bigger day size and larger calendar overall

`calendar_orientation`
: A value equal to: 'vertical', 'horizontal'. Orientation of calendar, either vertical or horizontal. Valid options are 'vertical' or 'horizontal'.

`is_RTL`
: Logical. Determines whether the calendar and days operate from left to right or from right to left

`reopen_calendar_on_clear`
: Logical. If True, the calendar will automatically open when cleared

`number_of_months_shown`
: Numeric. Number of calendar months that are shown when calendar is opened

`with_portal`
: Logical. If True, calendar will open in a screen overlay portal, not supported on vertical calendar

`with_full_screen_portal`
: Logical. If True, calendar will open in a full screen overlay portal, will take precedent over 'withPortal' if both are set to true, not supported on vertical calendar

`first_day_of_week`
: A value equal to: 0, 1, 2, 3, 4, 5, 6. Specifies what day is the first day of the week, values must be from [0, ..., 6] with 0 denoting Sunday and 6 denoting Saturday

`minimum_nights`
: Numeric. Specifies a minimum number of nights that must be selected between the startDate and the endDate

`stay_open_on_select`
: Logical. If True the calendar will not close when the user has selected a value and will wait until the user clicks off the calendar

`show_outside_days`
: Logical. If True the calendar will display days that rollover into the next month

`month_format`
: Character. Specifies the format that the month will be displayed in the calendar, valid formats are variations of "MM YY". For example: "MM YY" renders as '05 97' for May 1997 "MMMM, YYYY" renders as 'May, 1997' for May 1997 "MMM, YY" renders as 'Sep, 97' for September 1997

`display_format`
: Character. Specifies the format that the selected dates will be displayed valid formats are variations of "MM YY DD". For example: "MM YY DD" renders as '05 10 97' for May 10th 1997 "MMMM, YY" renders as 'May, 1997' for May 10th 1997 "M, D, YYYY" renders as '07, 10, 1997' for September 10th 1997 "MMMM" renders as 'May' for May 10 1997

`disabled`
: Logical. If True, no dates can be selected.

`clearable`
: Logical. Whether or not the dropdown is "clearable", that is, whether or not a small "x" appears on the right of the dropdown that removes the selected value.

`style`
: Named list. CSS styles appended to wrapper div

`className`
: Character. Appends a CSS class to the wrapper div component.

updatemode A value equal to: 'singledate', 'bothdates'. Determines when the component should update its value. If 'bothdates', then the DatePicker will only trigger its value when the user has finished picking both dates. If 'singledate', then the DatePicker will update its value as one date is picked.

loading_state Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If 'persisted' is truthy and hasn't changed from its previous value, any 'persisted_props' that the user has changed while using the app will keep those changes, as long as the new prop value also matches what was given originally. Used in conjunction with 'persistence_type' and 'persisted_props'.

persisted_props List of a value equal to: 'start_date', 'end_date's. Properties whose user interactions will persist after refreshing the component or the page.

persistence_type A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(
      dccDatePickerRange(
        id = "date-picker-range",
        start_date = as.Date("1997/5/10"),
        end_date_placeholder_text="Select a date!"
      )
    )

  app$run_server()
}
```

dccDatePickerSingle *DatePickerSingle component*

Description

DatePickerSingle is a tailor made component designed for selecting a single day off of a calendar. The DatePicker integrates well with the Python datetime module with the startDate and endDate being returned in a string format suitable for creating datetime objects. This component is based off of Airbnb's react-dates react component which can be found here: https://github.com/airbnb/react-dates

Usage

```
dccDatePickerSingle(id=NULL, date=NULL, min_date_allowed=NULL,
max_date_allowed=NULL, disabled_days=NULL,
initial_visible_month=NULL, day_size=NULL,
calendar_orientation=NULL, is_RTL=NULL, placeholder=NULL,
reopen_calendar_on_clear=NULL, number_of_months_shown=NULL,
with_portal=NULL, with_full_screen_portal=NULL,
first_day_of_week=NULL, stay_open_on_select=NULL,
show_outside_days=NULL, month_format=NULL,
display_format=NULL, disabled=NULL, clearable=NULL,
style=NULL, className=NULL, loading_state=NULL,
persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
date	Character. Specifies the starting date for the component, best practice is to pass value via datetime object
min_date_allowed	Character. Specifies the lowest selectable date for the component. Accepts datetime.datetime objects or strings in the format 'YYYY-MM-DD'
max_date_allowed	Character. Specifies the highest selectable date for the component. Accepts datetime.datetime objects or strings in the format 'YYYY-MM-DD'
disabled_days	List of characters. Specifies additional days between min_date_allowed and max_date_allowed that should be disabled. Accepted datetime.datetime objects or strings in the format 'YYYY-MM-DD'
initial_visible_month	Character. Specifies the month that is initially presented when the user opens the calendar. Accepts datetime.datetime objects or strings in the format 'YYYY-MM-DD'

day_size Numeric. Size of rendered calendar days, higher number means bigger day size and larger calendar overall

calendar_orientation
A value equal to: 'vertical', 'horizontal'. Orientation of calendar, either vertical or horizontal. Valid options are 'vertical' or 'horizontal'.

is_RTL Logical. Determines whether the calendar and days operate from left to right or from right to left

placeholder Character. Text that will be displayed in the input box of the date picker when no date is selected. Default value is 'Start Date'

reopen_calendar_on_clear
Logical. If True, the calendar will automatically open when cleared

number_of_months_shown
Numeric. Number of calendar months that are shown when calendar is opened

with_portal Logical. If True, calendar will open in a screen overlay portal, not supported on vertical calendar

with_full_screen_portal
Logical. If True, calendar will open in a full screen overlay portal, will take precedent over 'withPortal' if both are set to True, not supported on vertical calendar

first_day_of_week
A value equal to: 0, 1, 2, 3, 4, 5, 6. Specifies what day is the first day of the week, values must be from [0, ..., 6] with 0 denoting Sunday and 6 denoting Saturday

stay_open_on_select
Logical. If True the calendar will not close when the user has selected a value and will wait until the user clicks off the calendar

show_outside_days
Logical. If True the calendar will display days that rollover into the next month

month_format Character. Specifies the format that the month will be displayed in the calendar, valid formats are variations of "MM YY". For example: "MM YY" renders as '05 97' for May 1997 "MMMM, YYYY" renders as 'May, 1997' for May 1997 "MMM, YY" renders as 'Sep, 97' for September 1997

display_format Character. Specifies the format that the selected dates will be displayed valid formats are variations of "MM YY DD". For example: "MM YY DD" renders as '05 10 97' for May 10th 1997 "MMMM, YY" renders as 'May, 1997' for May 10th 1997 "M, D, YYYY" renders as '07, 10, 1997' for September 10th 1997 "MMMM" renders as 'May' for May 10 1997

disabled Logical. If True, no dates can be selected.

clearable Logical. Whether or not the dropdown is "clearable", that is, whether or not a small "x" appears on the right of the dropdown that removes the selected value.

style Named list. CSS styles appended to wrapper div

className Character. Appends a CSS class to the wrapper div component.

loading_state Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which

property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

`persistence` Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If 'persisted' is truthy and hasn't changed from its previous value, a 'date' that the user has changed while using the app will keep that change, as long as the new 'date' also matches what was given originally. Used in conjunction with 'persistence_type'.

`persisted_props`
List of a value equal to: 'date's. Properties whose user interactions will persist after refreshing the component or the page. Since only 'date' is allowed this prop can normally be ignored.

`persistence_type`
A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(
      dccDatePickerSingle(
        id = "date-picker-single",
        date = as.Date("1997/5/10")
      )
    )

  app$run_server()
}
```

---

dccDownload *Download component*

Description

The Download component opens a download dialog when the data property changes.

Usage

```
dccDownload(id=NULL, data=NULL, base64=NULL, type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks.
data	Lists containing elements 'filename', 'content', 'base64', 'type'. those elements have the following types: - filename (character; required): suggested filename in the download dialogue. - content (character; required): file content. - base64 (logical; optional): set to true, when data is base64 encoded. - type (character; optional): blob type, usually a mime-type.. On change, a download is invoked.
base64	Logical. Default value for base64, used when not set as part of the data property.
type	Character. Default value for type, used when not set as part of the data property.

Value

named list of JSON elements corresponding to React.js properties and their values

---

dccDropdown *Dropdown component*

Description

Dropdown is an interactive dropdown element for selecting one or more items. The values and labels of the dropdown items are specified in the 'options' property and the selected item(s) are specified with the 'value' property. Use a dropdown when you have many options (more than 5) or when you are constrained for space. Otherwise, you can use RadioItems or a Checklist, which have the benefit of showing the users all of the items at once.

Usage

```
dccDropdown(id=NULL, options=NULL, value=NULL, optionHeight=NULL,
className=NULL, clearable=NULL, disabled=NULL, multi=NULL,
placeholder=NULL, searchable=NULL, search_value=NULL,
style=NULL, loading_state=NULL, persistence=NULL,
persisted_props=NULL, persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
options	List of character | numeric | logicals | named list | list of lists containing elements 'label', 'value', 'disabled', 'title'. those elements have the following types: - label (character | numeric; required): the option's label - value (character | numeric | logical; required): the value of the option. this value corresponds to the items specified in the 'value' property. - disabled (logical; optional): if true, this option is disabled and cannot be selected. - title (character; optional): the html 'title' attribute for the option. allows for information on hover. for more information on this attribute, see https://developer.mozilla.org/en-us/docs/web/html/global_attributes/titles.

An array of options label: [string|number], value: [string|number], an optional disabled field can be used for each option

`value` Character | numeric | logical | list of character | numeric | logicals. The value of the input. If ‘multi‘ is false (the default) then value is just a string that corresponds to the values provided in the ‘options‘ property. If ‘multi‘ is true, then multiple values can be selected at once, and ‘value‘ is an array of items with values corresponding to those in the ‘options‘ prop.

`optionHeight` Numeric. height of each option. Can be increased when label lengths would wrap around

`className` Character. className of the dropdown element

`clearable` Logical. Whether or not the dropdown is "clearable", that is, whether or not a small "x" appears on the right of the dropdown that removes the selected value.

`disabled` Logical. If true, this dropdown is disabled and the selection cannot be changed.

`multi` Logical. If true, the user can select multiple values

`placeholder` Character. The grey, default text shown when no option is selected

`searchable` Logical. Whether to enable the searching feature or not

`search_value` Character. The value typed in the DropDown for searching.

`style` Named list. Defines CSS styles which will override styles previously set.

`loading_state` Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

`persistence` Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn’t changed from its previous value, a ‘value‘ that the user has changed while using the app will keep that change, as long as the new ‘value‘ also matches what was given originally. Used in conjunction with ‘persistence_type‘.

`persisted_props` List of a value equal to: ’value’s. Properties whose user interactions will persist after refreshing the component or the page. Since only ‘value‘ is allowed this prop can normally be ignored.

`persistence_type` A value equal to: ’local’, ’session’, ’memory’. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(
      htmlDiv(
        dccDropdown(
          options=list(
            list(label = "New York City", value = "NYC"),
            list(label = "Montreal", value = "MTL"),
            list(label = "San Francisco", value = "SF")
          ),
          value="MTL"
        )
      )
    )

  app$run_server()
}
```

---

dccGraph *Graph component*

Description

Graph can be used to render any plotly.js-powered data visualization. You can define callbacks based on user interaction with Graphs such as hovering, clicking or selecting

Usage

```
dccGraph(id=NULL, responsive=NULL, clickData=NULL,
clickAnnotationData=NULL, hoverData=NULL,
clear_on_unhover=NULL, selectedData=NULL, relayoutData=NULL,
extendData=NULL, prependData=NULL, restyleData=NULL,
figure=NULL, style=NULL, className=NULL, animate=NULL,
animation_options=NULL, config=NULL, loading_state=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
responsive	A value equal to: true, false, 'auto'. If True, the Plotly.js plot will be fully responsive to window resize and parent element resize event. This is achieved by overriding ‘config.responsive‘ to True, ‘figure.layout.autosize‘ to True and unsetting ‘figure.layout.height‘ and ‘figure.layout.width‘. If False, the Plotly.js plot not be responsive to window resize and parent element resize event. This is

achieved by overriding ‘config.responsive‘ to False and ‘figure.layout.autosize‘ to False. If ’auto’ (default), the Graph will determine if the Plotly.js plot can be made fully responsive (True) or not (False) based on the values in ‘config.responsive‘, ‘figure.layout.autosize‘, ‘figure.layout.height‘, ‘figure.layout.width‘. This is the legacy behavior of the Graph component.

Needs to be combined with appropriate dimension / styling through the ‘style‘ prop to fully take effect.

`clickData` Named list. Data from latest click event. Read-only.

`clickAnnotationData`
Named list. Data from latest click annotation event. Read-only.

`hoverData` Named list. Data from latest hover event. Read-only.

`clear_on_unhover`
Logical. If True, ‘clear_on_unhover‘ will clear the ‘hoverData‘ property when the user "unhovers" from a point. If False, then the ‘hoverData‘ property will be equal to the data from the last point that was hovered over.

`selectedData` Named list. Data from latest select event. Read-only.

`relayoutData` Named list. Data from latest relayout event which occurs when the user zooms or pans on the plot or other layout-level edits. Has the form ‘<attr string>: <value>‘ describing the changes made. Read-only.

`extendData` Unnamed list | named list. Data that should be appended to existing traces. Has the form ‘[updateData, traceIndices, maxPoints]‘, where ‘updateData‘ is an object containing the data to extend, ‘traceIndices‘ (optional) is an array of trace indices that should be extended, and ‘maxPoints‘ (optional) is either an integer defining the maximum number of points allowed or an object with key:value pairs matching ‘updateData‘ Reference the Plotly.extendTraces API for full usage: https://plotly.com/javascript/plotlyjs-function-reference/#plotlyextendtraces

`prependData` Unnamed list | named list. Data that should be prepended to existing traces. Has the form ‘[updateData, traceIndices, maxPoints]‘, where ‘updateData‘ is an object containing the data to prepend, ‘traceIndices‘ (optional) is an array of trace indices that should be prepended, and ‘maxPoints‘ (optional) is either an integer defining the maximum number of points allowed or an object with key:value pairs matching ‘updateData‘ Reference the Plotly.prependTraces API for full usage: https://plotly.com/javascript/plotlyjs-function-reference/#plotlyprependtraces

`restyleData` Unnamed list. Data from latest restyle event which occurs when the user toggles a legend item, changes parcoords selections, or other trace-level edits. Has the form ‘[edits, indices]‘, where ‘edits‘ is an object ‘<attr string>: <value>‘ describing the changes made, and ‘indices‘ is an array of trace indices that were edited. Read-only.

`figure` Lists containing elements ’data’, ’layout’, ’frames’. those elements have the following types: - data (list of named lists; optional) - layout (named list; optional) - frames (list of named lists; optional). Plotly ‘figure‘ object. See schema: https://plotly.com/javascript/reference

‘config‘ is set separately by the ‘config‘ property

`style` Named list. Generic style overrides on the plot div

`className` Character. className of the parent div

animate Logical. Beta: If true, animate between updates using plotly.js's 'animate' function

animation_options
Named list. Beta: Object containing animation settings. Only applies if 'animate' is 'true'

config Lists containing elements 'staticplot', 'plotlyserverurl', 'editable', 'edits', 'autosizable', 'responsive', 'queuelength', 'fillframe', 'framemargins', 'scrollzoom', 'doubleclick', 'doubleclickdelay', 'showtips', 'showaxisdraghandles', 'showaxisrangeentryboxes', 'showlink', 'senddata', 'linktext', 'displaymodebar', 'showsendtocloud', 'showeditinchartstudio', 'modebarbuttonstoremove', 'modebarbuttonstoadd', 'modebarbuttons', 'toimagebuttonoptions', 'displaylogo', 'watermark', 'plotglpixelratio', 'topojsonurl', 'mapboxaccesstoken', 'locale', 'locales'. those elements have the following types: - staticplot (logical; optional): no interactivity, for export or image generation - plotlyserverurl (character; optional): base url for a plotly cloud instance, if 'showsendtocloud' is enabled - editable (logical; optional): we can edit titles, move annotations, etc - sets all pieces of 'edits' unless a separate 'edits' config item overrides individual parts - edits (optional): a set of editable properties. edits has the following type: lists containing elements 'annotationposition', 'annotationtail', 'annotationtext', 'axistitletext', 'colorbarposition', 'colorbartitletext', 'legendposition', 'legendtext', 'shapeposition', 'titletext'. those elements have the following types: - annotationposition (logical; optional): the main anchor of the annotation, which is the text (if no arrow) or the arrow (which drags the whole thing leaving the arrow length & direction unchanged) - annotationtail (logical; optional): just for annotations with arrows, change the length and direction of the arrow - annotationtext (logical; optional) - axistitletext (logical; optional) - colorbarposition (logical; optional) - colorbartitletext (logical; optional) - legendposition (logical; optional) - legendtext (logical; optional): edit the trace name fields from the legend - shapeposition (logical; optional) - titletext (logical; optional): the global 'layout.title' - autosizable (logical; optional): do autosize once regardless of layout.autosize (use default width or height values otherwise) - responsive (logical; optional): whether to change layout size when the window size changes - queuelength (numeric; optional): set the length of the undo/redo queue - fillframe (logical; optional): if we do autosize, do we fill the container or the screen? - framemargins (numeric; optional): if we do autosize, set the frame margins in percents of plot size - scrollzoom (logical; optional): mousewheel or two-finger scroll zooms the plot - doubleclick (a value equal to: false, 'reset', 'autosize', 'reset+autosize'; optional): double click interaction (false, 'reset', 'autosize' or 'reset+autosize') - doubleclickdelay (numeric; optional): delay for registering a double-click event in ms. the minimum value is 100 and the maximum value is 1000. by default this is 300. - showtips (logical; optional): new users see some hints about interactivity - showaxisdraghandles (logical; optional): enable axis pan/zoom drag handles - showaxisrangeentryboxes (logical; optional): enable direct range entry at the pan/zoom drag points (drag handles must be enabled above) - showlink (logical; optional): link to open this plot in plotly - senddata (logical; optional): if we show a link, does it contain data or just link to a plotly file? - linktext (character; optional): text appearing in the senddata link - displaymodebar (a value equal to: true, false, 'hover'; optional): display the mode bar

(true, false, or 'hover') - showsendtocloud (logical; optional): should we include a modebar button to send this data to a plotly cloud instance, linked by 'plotlyserverurl'. by default this is false. - showeditinchartstudio (logical; optional): should we show a modebar button to send this data to a plotly chart studio plot. if both this and showsendtocloud are selected, only showeditinchartstudio will be honored. by default this is false. - modebarbuttonstoremove (unnamed list; optional): remove mode bar button by name. all modebar button names at https://github.com/plotly/plotly.js/blob/master/src/components/modebar/buttons.js common names include: senddatatocloud; (2d) zoom2d, pan2d, select2d, lasso2d, zoomin2d, zoomout2d, autoscale2d, resetscale2d; (cartesian) hoverclosestcartesian, hovercomparecartesian; (3d) zoom3d, pan3d, orbitrotation, tablerotation, handledrag3d, resetcameradefault3d, resetcameralastsave3d, hoverclosest3d; (geo) zoomingeo, zoomoutgeo, resetgeo, hoverclosestgeo; hoverclosestgl2d, hoverclosestpie, togglehover, resetviews. - modebarbuttonstoadd (unnamed list; optional): add mode bar button using config objects - modebarbuttons (logical | numeric | character | named list | unnamed list; optional): fully custom mode bar buttons as nested array, where the outer arrays represents button groups, and the inner arrays have buttons config objects or names of default buttons - toimagebuttonoptions (optional): modifications to how the toimage modebar button works. toimagebuttonoptions has the following type: lists containing elements 'format', 'filename', 'width', 'height', 'scale'. those elements have the following types: - format (a value equal to: 'jpeg', 'png', 'webp', 'svg'; optional): the file format to create - filename (character; optional): the name given to the downloaded file - width (numeric; optional): width of the downloaded file, in px - height (numeric; optional): height of the downloaded file, in px - scale (numeric; optional): extra resolution to give the file after rendering it with the given width and height - displaylogo (logical; optional): add the plotly logo on the end of the mode bar - watermark (logical; optional): add the plotly logo even with no modebar - plotglpixelratio (numeric; optional): increase the pixel ratio for gl plot images - topojsonurl (character; optional): url to topojson files used in geo charts - mapboxaccesstoken (logical | numeric | character | named list | unnamed list; optional): mapbox access token (required to plot mapbox trace types) if using an mapbox atlas server, set this option to '', so that plotly.js won't attempt to authenticate to the public mapbox server. - locale (character; optional): the locale to use. locales may be provided with the plot ('locales' below) or by loading them on the page, see: https://github.com/plotly/plotly.js/blob/master/dist/readme.md#to-include-localization - locales (named list; optional): localization definitions, if you choose to provide them with the plot rather than registering them globally.. Plotly.js config options. See https://plotly.com/javascript/configuration-options/ for more info.

`loading_state` Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)
    library(plotly)
    app <- Dash$new()

    year <- c(1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
      2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012)

    worldwide <- c(219, 146, 112, 127, 124, 180, 236, 207, 236, 263,
      350, 430, 474, 526, 488, 537, 500, 439)

    china <- c(16, 13, 10, 11, 28, 37, 43, 55, 56, 88, 105, 156, 270,
      299, 340, 403, 549, 499)

    data <- data.frame(year, worldwide, china)

    app$layout(
      htmlDiv(
        dccGraph(
          figure = layout(
                    add_trace(
                      plot_ly(data,
                              x = ~year,
                              y = ~worldwide,
                              type = "bar",
                              name = "Worldwide",
                              marker = list(color = "rgb(55, 83, 109)")
                              ),
                              y = ~china,
                              name = "China",
                              marker = list(color = "rgb(26, 118, 255)")
                      ),
                      yaxis = list(title = "Count"),
                      xaxis = list(title = "Year"),
                      barmode = "group",
                      title="US Export of Plastic Scrap"),
                      style = list("height" = 300),
                      id = "my_graph"
        )
      )
    )

  app$run_server()
}
```

dccInput *Input component*

Description

A basic HTML input control for entering text, numbers, or passwords. Note that checkbox and radio types are supported through the Checklist and RadioItems component. Dates, times, and file uploads are also supported through separate components.

Usage

```
dccInput(id=NULL, value=NULL, style=NULL, className=NULL,
debounce=NULL, type=NULL, autoComplete=NULL, autoFocus=NULL,
disabled=NULL, inputMode=NULL, list=NULL, max=NULL,
maxLength=NULL, min=NULL, minLength=NULL, multiple=NULL,
name=NULL, pattern=NULL, placeholder=NULL, readOnly=NULL,
required=NULL, selectionDirection=NULL, selectionEnd=NULL,
selectionStart=NULL, size=NULL, spellCheck=NULL, step=NULL,
n_submit=NULL, n_submit_timestamp=NULL, n_blur=NULL,
n_blur_timestamp=NULL, loading_state=NULL, persistence=NULL,
persisted_props=NULL, persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
value	Character | numeric. The value of the input
style	Named list. The input's inline styles
className	Character. The class of the input element
debounce	Logical. If true, changes to input will be sent back to the Dash server only on enter or when losing focus. If it's false, it will sent the value back on every change.
type	A value equal to: 'text', 'number', 'password', 'email', 'range', 'search', 'tel', 'url', 'hidden'. The type of control to render.
autoComplete	Character. This attribute indicates whether the value of the control can be automatically completed by the browser.
autoFocus	A value equal to: 'autofocus', 'autofocus', 'autofocus' | logical. The element should be automatically focused after the page loaded. autoFocus is an HTML boolean attribute - it is enabled by a boolean or 'autoFocus'. Alternative capitalizations 'autofocus' & 'AUTOFOCUS' are also acccepted.
disabled	A value equal to: 'disabled', 'disabled' | logical. If true, the input is disabled and can't be clicked on. disabled is an HTML boolean attribute - it is enabled by a boolean or 'disabled'. Alternative capitalizations 'DISABLED'

`inputMode` A value equal to: 'verbatim', 'latin', 'latin-name', 'latin-prose', 'full-width-latin', 'kana', 'katakana', 'numeric', 'tel', 'email', 'url'. Provides a hint to the browser as to the type of data that might be entered by the user while editing the element or its contents.

`list` Character. Identifies a list of pre-defined options to suggest to the user. The value must be the id of a <datalist> element in the same document. The browser displays only options that are valid values for this input element. This attribute is ignored when the type attribute's value is hidden, checkbox, radio, file, or a button type.

`max` Character | numeric. The maximum (numeric or date-time) value for this item, which must not be less than its minimum (min attribute) value.

`maxLength` Character | numeric. If the value of the type attribute is text, email, search, password, tel, or url, this attribute specifies the maximum number of characters (in UTF-16 code units) that the user can enter. For other control types, it is ignored. It can exceed the value of the size attribute. If it is not specified, the user can enter an unlimited number of characters. Specifying a negative number results in the default behavior (i.e. the user can enter an unlimited number of characters). The constraint is evaluated only when the value of the attribute has been changed.

`min` Character | numeric. The minimum (numeric or date-time) value for this item, which must not be greater than its maximum (max attribute) value.

`minLength` Character | numeric. If the value of the type attribute is text, email, search, password, tel, or url, this attribute specifies the minimum number of characters (in Unicode code points) that the user can enter. For other control types, it is ignored.

`multiple` Logical. This Boolean attribute indicates whether the user can enter more than one value. This attribute applies when the type attribute is set to email or file, otherwise it is ignored.

`name` Character. The name of the control, which is submitted with the form data.

`pattern` Character. A regular expression that the control's value is checked against. The pattern must match the entire value, not just some subset. Use the title attribute to describe the pattern to help the user. This attribute applies when the value of the type attribute is text, search, tel, url, email, or password, otherwise it is ignored. The regular expression language is the same as JavaScript RegExp algorithm, with the 'u' parameter that makes it treat the pattern as a sequence of unicode code points. The pattern is not surrounded by forward slashes.

`placeholder` Character | numeric. A hint to the user of what can be entered in the control . The placeholder text must not contain carriage returns or line-feeds. Note: Do not use the placeholder attribute instead of a <label> element, their purposes are different. The <label> attribute describes the role of the form element (i.e. it indicates what kind of information is expected), and the placeholder attribute is a hint about the format that the content should take. There are cases in which the placeholder attribute is never displayed to the user, so the form must be understandable without it.

`readOnly` Logical | a value equal to: 'readonly', 'readonly', 'readonly'. This attribute indicates that the user cannot modify the value of the control. The value of the

attribute is irrelevant. If you need read-write access to the input value, do not add the "readonly" attribute. It is ignored if the value of the type attribute is hidden, range, color, checkbox, radio, file, or a button type (such as button or submit). readOnly is an HTML boolean attribute - it is enabled by a boolean or 'readOnly'. Alternative capitalizations 'readonly' & 'READONLY' are also acccepted.

`required` A value equal to: 'required', 'required' | logical. This attribute specifies that the user must fill in a value before submitting a form. It cannot be used when the type attribute is hidden, image, or a button type (submit, reset, or button). The :optional and :required CSS pseudo-classes will be applied to the field as appropriate. required is an HTML boolean attribute - it is enabled by a boolean or 'required'. Alternative capitalizations 'REQUIRED' are also acccepted.

`selectionDirection`
Character. The direction in which selection occurred. This is "forward" if the selection was made from left-to-right in an LTR locale or right-to-left in an RTL locale, or "backward" if the selection was made in the opposite direction. On platforms on which it's possible this value isn't known, the value can be "none"; for example, on macOS, the default direction is "none", then as the user begins to modify the selection using the keyboard, this will change to reflect the direction in which the selection is expanding.

`selectionEnd` Character. The offset into the element's text content of the last selected character. If there's no selection, this value indicates the offset to the character following the current text input cursor position (that is, the position the next character typed would occupy).

`selectionStart` Character. The offset into the element's text content of the first selected character. If there's no selection, this value indicates the offset to the character following the current text input cursor position (that is, the position the next character typed would occupy).

`size` Character. The initial size of the control. This value is in pixels unless the value of the type attribute is text or password, in which case it is an integer number of characters. Starting in, this attribute applies only when the type attribute is set to text, search, tel, url, email, or password, otherwise it is ignored. In addition, the size must be greater than zero. If you do not specify a size, a default value of 20 is used.' simply states "the user agent should ensure that at least that many characters are visible", but different characters can have different widths in certain fonts. In some browsers, a certain string with x characters will not be entirely visible even if size is defined to at least x.

`spellCheck` A value equal to: 'true', 'false' | logical. Setting the value of this attribute to true indicates that the element needs to have its spelling and grammar checked. The value default indicates that the element is to act according to a default behavior, possibly based on the parent element's own spellcheck value. The value false indicates that the element should not be checked.

`step` Character | numeric. Works with the min and max attributes to limit the increments at which a numeric or date-time value can be set. It can be the string any or a positive floating point number. If this attribute is not set to any, the control accepts only values at multiples of the step value greater than the minimum.

n_submit Numeric. Number of times the ‘Enter‘ key was pressed while the input had focus.

n_submit_timestamp
Numeric. Last time that ‘Enter‘ was pressed.

n_blur Numeric. Number of times the input lost focus.

n_blur_timestamp
Numeric. Last time the input lost focus.

loading_state Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn’t changed from its previous value, a ‘value‘ that the user has changed while using the app will keep that change, as long as the new ‘value‘ also matches what was given originally. Used in conjunction with ‘persistence_type‘.

persisted_props
List of a value equal to: ’value’s. Properties whose user interactions will persist after refreshing the component or the page. Since only ‘value‘ is allowed this prop can normally be ignored.

persistence_type
A value equal to: ’local’, ’session’, ’memory’. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(
      htmlDiv(
        dccInput(
          placeholder = "Enter a value...",
          type = "text",
          value = ""
        )
      )
    )
```

```
  app$run_server()
}
```

---

dccInterval *Interval component*

Description

A component that repeatedly increments a counter ‘n_intervals‘ with a fixed time delay between each increment. Interval is good for triggering a component on a recurring basis. The time delay is set with the property "interval" in milliseconds.

Usage

```
dccInterval(id=NULL, interval=NULL, disabled=NULL, n_intervals=NULL,
max_intervals=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
interval	Numeric. This component will increment the counter ‘n_intervals‘ every ‘interval‘ milliseconds
disabled	Logical. If True, the counter will no longer update
n_intervals	Numeric. Number of times the interval has passed
max_intervals	Numeric. Number of times the interval will be fired. If -1, then the interval has no limit (the default) and if 0 then the interval stops running.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)
    library(plotly)

    app <- Dash$new()

    app$layout(
      htmlDiv(list(
        htmlH2('3 Second Updates'),
        dccInterval(id = '3s-interval',
                    interval= 3*1000,
                    n_intervals = 0),
        htmlDiv(list(
```

```
            dccGraph(id = 'live-graph')
          )
        )
        )
      )
    )

    app$callback(
      output = list(
        output('live-graph', 'figure')
      ),
      params = list(
        input('3s-interval', 'n_intervals')
      ),

      update_graph <- function(n_intervals) {
        df <- data.frame(
          'time' = c(1:8),
          'value' = sample(1:8, 8),
          'value-2' = sample(1:8, 8)
        )

        bar <- animation_opts(plot_ly(
          data = df, x=~time, y=~value, type = "bar"),
          1000, easing = "cubic-in-out"
        )

        return(list(bar))
      }
    )

   app$run_server()
}
```

---

dccLink *Link component*

Description

Link allows you to create a clickable link within a multi-page app. For links with destinations outside the current app, 'html.A' is a better component to use.

Usage

```
dccLink(children=NULL, id=NULL, href=NULL, refresh=NULL,
className=NULL, style=NULL, title=NULL, target=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
href	Character. The URL of a linked resource.
refresh	Logical. Controls whether or not the page will refresh when the link is clicked
className	Character. Often used with CSS to style elements with common properties.
style	Named list. Defines CSS styles which will override styles previously set.
title	Character. Adds the title attribute to your link, which can contain supplementary information.
target	Character. Specifies where to open the link reference.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(htmlDiv(list(
              # represents the URL bar, doesn't render anything
              dccLocation(id = 'url', refresh=FALSE),
              dccLink('Navigate to "/"', href='/'),
              htmlBr(),
              dccLink('Navigate to "/page-2"', href='/page-2'),
              # content will be rendered in this element
              htmlDiv(id='page-content')
            )
        )
    )

    app$callback(output=list(id='page-content', property='children'),
                 params=list(
              input(id='url', property='pathname')),
              function(pathname) {
              paste0('You are on page ', pathname)
              }
```

```
    )

  app$run_server()
}
```

---

dccLoading *Loading component*

Description

A Loading component that wraps any other component and displays a spinner until the wrapped component has rendered.

Usage

```
dccLoading(children=NULL, id=NULL, type=NULL, fullscreen=NULL,
debug=NULL, className=NULL, parent_className=NULL,
style=NULL, parent_style=NULL, color=NULL,
loading_state=NULL)
```

Arguments

children	List of a list of or a singular dash component, string or numbers | a list of or a singular dash component, string or number. Array that holds components to render
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
type	A value equal to: 'graph', 'cube', 'circle', 'dot', 'default'. Property that determines which spinner to show one of 'graph', 'cube', 'circle', 'dot', or 'default'.
fullscreen	Logical. Boolean that makes the spinner display full-screen
debug	Logical. If true, the spinner will display the component_name and prop_name while loading
className	Character. Additional CSS class for the spinner root DOM node
parent_className	Character. Additional CSS class for the outermost dcc.Loading parent div DOM node
style	Named list. Additional CSS styling for the spinner root DOM node
parent_style	Named list. Additional CSS styling for the outermost dcc.Loading parent div DOM node
color	Character. Primary colour used for the loading spinners
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(htmlDiv(
      children=list(
        htmlH3("Edit text input to see loading state"),
        dccInput(id="input-1", value='Input triggers local spinner'),
     dccLoading(id="loading-1", children=list(htmlDiv(id="loading-output-1")), type="default"),
        htmlDiv(
          list(
            dccInput(id="input-2", value='Input triggers nested spinner'),
            dccLoading(
              id="loading-2",
              children=list(htmlDiv(list(htmlDiv(id="loading-output-2")))),
              type="circle"
            )
          )
        )
      )
    ))

    app$callback(
      output = list(id='loading-output-1', property = 'children'),
      params = list(input(id = 'input-1', property = 'value')),
      function(value){
        Sys.sleep(1)
        return(value)
      }
    )

    app$callback(
      output = list(id='loading-output-2', property = 'children'),
      params = list(input(id = 'input-2', property = 'value')),
      function(value){
        Sys.sleep(1)
        return(value)
      }
    )

  app$run_server()
}
```

dccLocation *Location component*

Description

Update and track the current window.location object through the window.history state. Use in conjunction with the ‘dash_core_components.Link‘ component to make apps with multiple pages.

Usage

```
dccLocation(id=NULL, pathname=NULL, search=NULL, hash=NULL, href=NULL,
refresh=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
pathname	Character. pathname in window.location - e.g., "/my/full/pathname"
search	Character. search in window.location - e.g., "?myargument=1"
hash	Character. hash in window.location - e.g., "#myhash"
href	Character. href in window.location - e.g., "/my/full/pathname?myargument=1#myhash"
refresh	Logical. Refresh the page when the location is updated?

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(htmlDiv(list(
              # represents the URL bar, doesn't render anything
              dccLocation(id = 'url', refresh=FALSE),
              dccLink('Navigate to "/"', href='/'),
              htmlBr(),
              dccLink('Navigate to "/page-2"', href='/page-2'),

              # content will be rendered in this element
              htmlDiv(id='page-content')
            )
        )
    )
```

```
    app$callback(output=list(id='page-content', property='children'),
                 params=list(
                input(id='url', property='pathname')),
                function(pathname)
                {
                paste0('You are on page ', pathname)
                }
    )

  app$run_server()
}
```

---

dccLogoutButton *LogoutButton component*

Description

Logout button to submit a form post request to the ‘logout_url‘ prop. Usage is intended for dash-deployment-server authentication. DDS usage: ‘dcc.LogoutButton(logout_url=os.getenv(’DASH_LOGOUT_URL’))‘ Custom usage: - Implement a login mechanism. - Create a flask route with a post method handler. ‘@app.server.route(’/logout’, methods=[’POST’])‘ - The logout route should perform what’s necessary for the user to logout. - If you store the session in a cookie, clear the cookie: ‘rep = flask.Response(); rep.set_cookie(’session’, ”, expires=0)‘ - Create a logout button component and assign it the logout_url ‘dcc.LogoutButton(logout_url=’/logout’)‘ See https://dash.plotly.com/dash-core-components/logout_button for more documentation and examples.

Usage

```
dccLogoutButton(id=NULL, label=NULL, logout_url=NULL, style=NULL,
method=NULL, className=NULL, loading_state=NULL)
```

Arguments

id	Character. Id of the button.
label	Character. Text of the button
logout_url	Character. Url to submit a post logout request.
style	Named list. Style of the button
method	Character. Http method to submit the logout form.
className	Character. CSS class for the button.
loading_state	Lists containing elements ’is_loading’, ’prop_name’, ’component_name’. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(
      dccLogoutButton(logout_url='/custom-auth/logout')
    )

  app$run_server()
}
```

---

dccMarkdown *Markdown component*

Description

A component that renders Markdown text as specified by the GitHub Markdown spec. These component uses [react-markdown](https://rexxars.github.io/react-markdown/) under the hood.

Usage

```
dccMarkdown(children=NULL, id=NULL, className=NULL,
dangerously_allow_html=NULL, dedent=NULL,
highlight_config=NULL, loading_state=NULL, style=NULL)
```

Arguments

children	Character | list of characters. A markdown string (or array of strings) that adhreres to the CommonMark spec
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
className	Character. Class name of the container element
dangerously_allow_html	Logical. A boolean to control raw HTML escaping. Setting HTML from code is risky because it's easy to inadvertently expose your users to a cross-site scripting (XSS) (https://en.wikipedia.org/wiki/Cross-site_scripting) attack.
dedent	Logical. Remove matching leading whitespace from all lines. Lines that are empty, or contain *only* whitespace, are ignored. Both spaces and tab characters are removed, but only if they match; we will not convert tabs to spaces or vice versa.

highlight_config
: Lists containing elements 'theme'. those elements have the following types: - theme (a value equal to: 'dark', 'light'; optional): color scheme; default 'light'. Config options for syntax highlighting.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

style
: Named list. User-defined inline styles for the rendered Markdown

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$title("dccMarkdown Syntax Highlighting Demo")

    # dccMarkdown leverages Highlight.js, which allows
    # app developers to specify the language inline
    # and highlight its syntax properly:
    app$layout(
      htmlDiv(
        list(
          htmlDiv(htmlH2("Syntax markdown demo:")),
          dccMarkdown(children = "
          ```r
          library(dash)

          app <- Dash$new()
          app$layout(htmlDiv('Dash app code wrapped within an app'))
          app$run_server()
          ```")
)
)
)

  app$run_server()
}
```

dccRadioItems *RadioItems component*

Description

RadioItems is a component that encapsulates several radio item inputs. The values and labels of the RadioItems is specified in the ‘options‘ property and the seleced item is specified with the ‘value‘ property. Each radio item is rendered as an input with a surrounding label.

Usage

```
dccRadioItems(id=NULL, options=NULL, value=NULL, style=NULL,
className=NULL, inputStyle=NULL, inputClassName=NULL,
labelStyle=NULL, labelClassName=NULL, loading_state=NULL,
persistence=NULL, persisted_props=NULL,
persistence_type=NULL, inline=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
options	List of character | numeric | logicals | named list | list of lists containing elements 'label', 'value', 'disabled', 'title'. those elements have the following types: - label (character | numeric | logical; required): the option's label - value (character | numeric | logical; required): the value of the option. this value corresponds to the items specified in the ‘value‘ property. - disabled (logical; optional): if true, this option is disabled and cannot be selected. - title (character; optional): the html 'title' attribute for the option. allows for information on hover. for more information on this attribute, see https://developer.mozilla.org/en-us/docs/web/html/global_attributes/titles. An array of options, or inline dictionary of options
value	Character | numeric | logical. The currently selected value
style	Named list. The style of the container (div)
className	Character. The class of the container (div)
inputStyle	Named list. The style of the <input> radio element
inputClassName	Character. The class of the <input> radio element
labelStyle	Named list. The style of the <label> that wraps the radio input and the option's label
labelClassName	Character. The class of the <label> that wraps the radio input and the option's label
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If 'persisted' is truthy and hasn't changed from its previous value, a 'value' that the user has changed while using the app will keep that change, as long as the new 'value' also matches what was given originally. Used in conjunction with 'persistence_type'.

persisted_props List of a value equal to: 'value's. Properties whose user interactions will persist after refreshing the component or the page. Since only 'value' is allowed this prop can normally be ignored.

persistence_type A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

inline Logical. Indicates whether labelStyle should be inline or not True: Automatically set 'display': 'inline-block' to labelStyle False: No additional styles are passed into labelStyle.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(
      htmlDiv(
        dccRadioItems(
          options=list(
            list("label" = "New York City", "value" = "NYC"),
            list("label" = "Montreal", "value" = "MTL"),
            list("label" = "San Francisco", "value" = "SF")
          ),
          value = "MTL"
        )
      )
    )

  app$run_server()
}
```

dccRangeSlider *RangeSlider component*

Description

A double slider with two handles. Used for specifying a range of numerical values.

Usage

```
dccRangeSlider(id=NULL, min=NULL, max=NULL, step=NULL, marks=NULL,
value=NULL, drag_value=NULL, allowCross=NULL,
className=NULL, count=NULL, disabled=NULL, dots=NULL,
included=NULL, pushable=NULL, tooltip=NULL, vertical=NULL,
verticalHeight=NULL, updatemode=NULL, loading_state=NULL,
persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
min	Numeric. Minimum allowed value of the slider
max	Numeric. Maximum allowed value of the slider
step	Numeric. Value by which increments or decrements are made
marks	List with named elements and values of type character | lists containing elements 'label', 'style'. those elements have the following types: - label (character; optional) - style (named list; optional). Marks on the slider. The key determines the position (a number), and the value determines what will show. If you want to set the style of a specific mark point, the value should be an object which contains style and label properties.
value	List of numerics. The value of the input
drag_value	List of numerics. The value of the input during a drag
allowCross	Logical. allowCross could be set as true to allow those handles to cross.
className	Character. Additional CSS class for the root DOM node
count	Numeric. Determine how many ranges to render, and multiple handles will be rendered (number + 1).
disabled	Logical. If true, the handles can't be moved.
dots	Logical. When the step value is greater than 1, you can set the dots to true if you want to render the slider with dots.
included	Logical. If the value is true, it means a continuous value is included. Otherwise, it is an independent value.
pushable	Logical | numeric. pushable could be set as true to allow pushing of surrounding handles when moving an handle. When set to a number, the number will be the minimum ensured distance between handles.

`tooltip` Lists containing elements 'always_visible', 'placement'. those elements have the following types: - always_visible (logical; optional): determines whether tooltips should always be visible (as opposed to the default, visible on hover) - placement (a value equal to: 'left', 'right', 'top', 'bottom', 'topleft', 'topright', 'bottomleft', 'bottomright'; optional): determines the placement of tooltips see https://github.com/react-component/tooltip#api top/bottom* sets the _origin_ of the tooltip, so e.g. ‘topleft‘ will in reality appear to be on the top right of the handle. Configuration for tooltips describing the current slider values

`vertical` Logical. If true, the slider will be vertical

`verticalHeight` Numeric. The height, in px, of the slider if it is vertical.

`updatemode` A value equal to: 'mouseup', 'drag'. Determines when the component should update its ‘value‘ property. If ‘mouseup‘ (the default) then the slider will only trigger its value when the user has finished dragging the slider. If ‘drag‘, then the slider will update its value continuously as it is being dragged. Note that for the latter case, the ‘drag_value‘ property could be used instead.

`loading_state` Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

`persistence` Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If ‘persisted‘ is truthy and hasn't changed from its previous value, a ‘value‘ that the user has changed while using the app will keep that change, as long as the new ‘value‘ also matches what was given originally. Used in conjunction with ‘persistence_type‘.

`persisted_props` List of a value equal to: 'value's. Properties whose user interactions will persist after refreshing the component or the page. Since only ‘value‘ is allowed this prop can normally be ignored.

`persistence_type` A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()
```

```
    app$layout(
      htmlDiv(
        dccRangeSlider(
          count = 1,
          min = -5,
          max = 10,
          step = 0.5,
          value = list(-3, 7),
          marks = as.list(
            setNames(-5:10, as.character(-5:10))
          )
        )
      )
    )

  app$run_server()
}
```

---

dccSlider *Slider component*

Description

A slider component with a single handle.

Usage

```
dccSlider(id=NULL, min=NULL, max=NULL, step=NULL, marks=NULL,
value=NULL, drag_value=NULL, className=NULL, disabled=NULL,
dots=NULL, included=NULL, tooltip=NULL, vertical=NULL,
verticalHeight=NULL, updatemode=NULL, loading_state=NULL,
persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
min	Numeric. Minimum allowed value of the slider
max	Numeric. Maximum allowed value of the slider
step	Numeric. Value by which increments or decrements are made
marks	List with named elements and values of type character | lists containing elements 'label', 'style'. those elements have the following types: - label (character; optional) - style (named list; optional). Marks on the slider. The key determines the position (a number), and the value determines what will show. If you want to set the style of a specific mark point, the value should be an object which contains style and label properties.

value Numeric. The value of the input

drag_value Numeric. The value of the input during a drag

className Character. Additional CSS class for the root DOM node

disabled Logical. If true, the handles can't be moved.

dots Logical. When the step value is greater than 1, you can set the dots to true if you want to render the slider with dots.

included Logical. If the value is true, it means a continuous value is included. Otherwise, it is an independent value.

tooltip Lists containing elements 'always_visible', 'placement'. those elements have the following types: - always_visible (logical; optional): determines whether tooltips should always be visible (as opposed to the default, visible on hover) - placement (a value equal to: 'left', 'right', 'top', 'bottom', 'topleft', 'topright', 'bottomleft', 'bottomright'; optional): determines the placement of tooltips see https://github.com/react-component/tooltip#api top/bottom* sets the _origin_ of the tooltip, so e.g. 'topleft' will in reality appear to be on the top right of the handle. Configuration for tooltips describing the current slider value

vertical Logical. If true, the slider will be vertical

verticalHeight Numeric. The height, in px, of the slider if it is vertical.

updatemode A value equal to: 'mouseup', 'drag'. Determines when the component should update its 'value' property. If 'mouseup' (the default) then the slider will only trigger its value when the user has finished dragging the slider. If 'drag', then the slider will update its value continuously as it is being dragged. If you want different actions during and after drag, leave 'updatemode' as 'mouseup' and use 'drag_value' for the continuously updating value.

loading_state Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If 'persisted' is truthy and hasn't changed from its previous value, a 'value' that the user has changed while using the app will keep that change, as long as the new 'value' also matches what was given originally. Used in conjunction with 'persistence_type'.

persisted_props

List of a value equal to: 'value's. Properties whose user interactions will persist after refreshing the component or the page. Since only 'value' is allowed this prop can normally be ignored.

persistence_type

A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(
      htmlDiv(
        list(
          dccSlider(
            id = "slider-input",
            min = -5,
            max = 10,
            step = 0.5,
            value = -3
          ),
          htmlDiv(
            id = "slider-output",
            children = "Make a selection on the slider to see the value appear here."
          )
        )
      )
    )

    app$callback(
      output("slider-output", "children"),
      list(input("slider-input", "value")),
      function(value) {
        return(paste0("You have chosen ", value, " on the slider above."))
      }
    )

   app$run_server()
}
```

---

dccStore *Store component*

Description

Easily keep data on the client side with this component. The data is not inserted in the DOM. Data can be in memory, localStorage or sessionStorage. The data will be kept with the id as key.

Usage

```
dccStore(id=NULL, storage_type=NULL, data=NULL, clear_data=NULL,
modified_timestamp=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
storage_type	A value equal to: 'local', 'session', 'memory'. The type of the web storage. memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.
data	Named list | unnamed list | numeric | character | logical. The stored data for the id.
clear_data	Logical. Set to true to remove the data contained in 'data_key'.
modified_timestamp	Numeric. The last time the storage was modified.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(htmlDiv(list(
      # The memory store reverts to the default on every page refresh
      dccStore(id='memory'),
      # The local store will take the initial data
      # only the first time the page is loaded
      # and keep it until it is cleared.
      dccStore(id='local', storage_type='local'),
      # Same as the local store but will lose the data
      # when the browser/tab closes.
      dccStore(id='session', storage_type='session'),
      htmlTable(list(
        htmlThead(list(
          htmlTr(htmlTh('Click to store in:', colSpan='3')),
          htmlTr(list(
            htmlTh(htmlButton('memory', id='memory-button')),
            htmlTh(htmlButton('localStorage', id='local-button')),
            htmlTh(htmlButton('sessionStorage', id='session-button'))
          )),
          htmlTr(list(
            htmlTh('Memory clicks'),
            htmlTh('Local clicks'),
            htmlTh('Session clicks')
          ))
        )),
        htmlTbody(list(
          htmlTr(list(
```

```
            htmlTd(0, id='memory-clicks'),
            htmlTd(0, id='local-clicks'),
            htmlTd(0, id='session-clicks')
          ))
        ))
      ))
    )))

    for (i in c('memory', 'local', 'session')) {
      app$callback(
        output(id = i, property = 'data'),
        params = list(
          input(id = paste0(i, '-button'), property = 'n_clicks'),
          state(id = i, property = 'data')
        ),
        function(n_clicks, data){
          if(is.null(n_clicks)){
            return()
          }
          if(is.null(data[[1]])){
            data = list('clicks' = 0)
          } else{
            data = data
          }
          data['clicks'] = data$clicks + 1
          return(data)
        }
      )
    }

    for (i in c('memory', 'local', 'session')) {
      app$callback(
        output(id = paste0(i, '-clicks'), property = 'children'),
        params = list(
          input(id = i, property = 'modified_timestamp'),
          state(id = i, property = 'data')
        ),
        function(ts, data){
          if(is.null(ts)){
            return()
          }
          if(is.null(data[[1]])){
            data = list()
          } else {
            data = data
          }
          return(data$clicks[[1]])
        }
      )
    }

  app$run_server()
}
```

dccTab *Tab component*

Description

Part of dcc.Tabs - this is the child Tab component used to render a tabbed page. Its children will be set as the content of that tab, which if clicked will become visible.

Usage

```
dccTab(children=NULL, id=NULL, label=NULL, value=NULL,
disabled=NULL, disabled_style=NULL, disabled_className=NULL,
className=NULL, selected_className=NULL, style=NULL,
selected_style=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The content of the tab - will only be displayed if this tab is selected
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
label	Character. The tab's label
value	Character. Value for determining which Tab is currently selected
disabled	Logical. Determines if tab is disabled or not - defaults to false
disabled_style	Named list. Overrides the default (inline) styles when disabled
disabled_className	Character. Appends a class to the Tab component when it is disabled.
className	Character. Appends a class to the Tab component.
selected_className	Character. Appends a class to the Tab component when it is selected.
style	Named list. Overrides the default (inline) styles for the Tab component.
selected_style	Named list. Overrides the default (inline) styles for the Tab component when it is selected.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(htmlDiv(list(
      dccTabs(id="tabs", value='tab-1', children=list(
        dccTab(label='Tab one', value='tab-1'),
        dccTab(label='Tab two', value='tab-2')
        )
      ),
      htmlDiv(id='tabs-content')
      )
      )
    )

    app$callback(output('tabs-content', 'children'),
        params = list(input('tabs', 'value')),
    function(tab){
      if(tab == 'tab-1'){
      return(htmlDiv(list(
        htmlH3('Tab content 1')
        )))}
      else if(tab == 'tab-2'){
      return(htmlDiv(list(
        htmlH3('Tab content 2')
        )))}
      }
    )

  app$run_server()
}
```

---

dccTabs *Tabs component*

Description

A Dash component that lets you render pages with tabs - the Tabs component's children can be dcc.Tab components, which can hold a label that will be displayed as a tab, and can in turn hold children components that will be that tab's content.

Usage

```
dccTabs(children=NULL, id=NULL, value=NULL, className=NULL,
content_className=NULL, parent_className=NULL, style=NULL,
parent_style=NULL, content_style=NULL, vertical=NULL,
mobile_breakpoint=NULL, colors=NULL, loading_state=NULL,
```

```
persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

children	List of a list of or a singular dash component, string or numbers | a list of or a singular dash component, string or number. Array that holds Tab components
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
value	Character. The value of the currently selected Tab
className	Character. Appends a class to the Tabs container holding the individual Tab components.
content_className	Character. Appends a class to the Tab content container holding the children of the Tab that is selected.
parent_className	Character. Appends a class to the top-level parent container holding both the Tabs container and the content container.
style	Named list. Appends (inline) styles to the Tabs container holding the individual Tab components.
parent_style	Named list. Appends (inline) styles to the top-level parent container holding both the Tabs container and the content container.
content_style	Named list. Appends (inline) styles to the tab content container holding the children of the Tab that is selected.
vertical	Logical. Renders the tabs vertically (on the side)
mobile_breakpoint	Numeric. Breakpoint at which tabs are rendered full width (can be 0 if you don't want full width tabs on mobile)
colors	Lists containing elements 'border', 'primary', 'background'. those elements have the following types: - border (character; optional) - primary (character; optional) - background (character; optional). Holds the colors used by the Tabs and Tab components. If you set these, you should specify colors for all properties, so: colors: border: '#d6d6d6', primary: '#1975FA', background: '#f9f9f9'
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
persistence	Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If 'persisted' is truthy and hasn't changed from its previous value, a 'value' that the user has changed while using the app will keep that change, as long as the new 'value' also matches what was given originally. Used in conjunction with 'persistence_type'.

persisted_props
: List of a value equal to: 'value's. Properties whose user interactions will persist after refreshing the component or the page. Since only 'value' is allowed this prop can normally be ignored.

persistence_type
: A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(htmlDiv(list(
      dccTabs(id="tabs", value='tab-1', children=list(
        dccTab(label='Tab one', value='tab-1'),
        dccTab(label='Tab two', value='tab-2')
        )
      ),
      htmlDiv(id='tabs-content')
      )
      )
    )

    app$callback(output('tabs-content', 'children'),
        params = list(input('tabs', 'value')),
    function(tab){
      if(tab == 'tab-1'){
      return(htmlDiv(list(
        htmlH3('Tab content 1')
        )))}
      else if(tab == 'tab-2'){
      return(htmlDiv(list(
        htmlH3('Tab content 2')
        )))}
      }
    )

  app$run_server()
}
```

dccTextarea *Textarea component*

Description

A basic HTML textarea for entering multiline text.

Usage

```
dccTextarea(id=NULL, value=NULL, autoFocus=NULL, cols=NULL,
disabled=NULL, form=NULL, maxLength=NULL, minLength=NULL,
name=NULL, placeholder=NULL, readOnly=NULL, required=NULL,
rows=NULL, wrap=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, n_blur=NULL,
n_blur_timestamp=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, loading_state=NULL,
persistence=NULL, persisted_props=NULL,
persistence_type=NULL)
```

Arguments

id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
value	Character. The value of the textarea
autoFocus	Character. The element should be automatically focused after the page loaded.
cols	Character | numeric. Defines the number of columns in a textarea.
disabled	Character | logical. Indicates whether the user can interact with the element.
form	Character. Indicates the form that is the owner of the element.
maxLength	Character | numeric. Defines the maximum number of characters allowed in the element.
minLength	Character | numeric. Defines the minimum number of characters allowed in the element.
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
placeholder	Character. Provides a hint to the user of what can be entered in the field.
readOnly	Logical | a value equal to: 'readonly', 'readonly', 'readonly'. Indicates whether the element can be edited. readOnly is an HTML boolean attribute - it is enabled by a boolean or 'readOnly'. Alternative capitalizations 'readonly' & 'READONLY' are also acccepted.
required	A value equal to: 'required', 'required' | logical. Indicates whether this element is required to fill out or not. required is an HTML boolean attribute - it is enabled by a boolean or 'required'. Alternative capitalizations 'REQUIRED' are also acccepted.

rows Character | numeric. Defines the number of rows in a text area.

wrap Character. Indicates whether the text should be wrapped.

accessKey Character. Defines a keyboard shortcut to activate or add focus to the element.

className Character. Often used with CSS to style elements with common properties.

contentEditable Character | logical. Indicates whether the element's content is editable.

contextMenu Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable A value equal to: 'true', 'false' | logical. Defines whether the element can be dragged.

hidden Character. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang Character. Defines the language used in the element.

spellCheck A value equal to: 'true', 'false' | logical. Indicates whether spell checking is allowed for the element.

style Named list. Defines CSS styles which will override styles previously set.

tabIndex Character | numeric. Overrides the browser's default tab order and follows the one specified instead.

title Character. Text to be displayed in a tooltip when hovering over the element.

n_blur Numeric. Number of times the textarea lost focus.

n_blur_timestamp Numeric. Last time the textarea lost focus.

n_clicks Numeric. Number of times the textarea has been clicked.

n_clicks_timestamp Numeric. Last time the textarea was clicked.

loading_state Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

persistence Logical | character | numeric. Used to allow user interactions in this component to be persisted when the component - or the page - is refreshed. If 'persisted' is truthy and hasn't changed from its previous value, a 'value' that the user has changed while using the app will keep that change, as long as the new 'value' also matches what was given originally. Used in conjunction with 'persistence_type'.

persisted_props List of a value equal to: 'value's. Properties whose user interactions will persist after refreshing the component or the page. Since only 'value' is allowed this prop can normally be ignored.

persistence_type
: A value equal to: 'local', 'session', 'memory'. Where persisted user changes will be stored: memory: only kept in memory, reset on page refresh. local: window.localStorage, data is kept after the browser quit. session: window.sessionStorage, data is cleared once the browser quit.

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)

    app <- Dash$new()

    app$layout(
      htmlDiv(
        dccTextarea(
          placeholder = 'Enter a value...',
          value = 'This is a TextArea component'
        )
      )
    )

  app$run_server()
}
```

---

dccTooltip *Tooltip component*

Description

A tooltip with an absolute position.

Usage

```
dccTooltip(children=NULL, id=NULL, className=NULL, style=NULL,
bbox=NULL, show=NULL, direction=NULL, border_color=NULL,
background_color=NULL, loading_text=NULL, zindex=NULL,
targetable=NULL, loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number. The contents of the tooltip
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

`className`	Character. The class of the tooltip
`style`	Named list. The style of the tooltip
`bbox`	Lists containing elements 'x0', 'y0', 'x1', 'y1'. those elements have the following types: - x0 (numeric; optional) - y0 (numeric; optional) - x1 (numeric; optional) - y1 (numeric; optional). The bounding box coordinates of the item to label, in px relative to the positioning parent of the Tooltip component.
`show`	Logical. Whether to show the tooltip
`direction`	A value equal to: 'top', 'right', 'bottom', 'left'. The side of the 'bbox' on which the tooltip should open.
`border_color`	Character. Color of the tooltip border, as a CSS color string.
`background_color`	Character. Color of the tooltip background, as a CSS color string.
`loading_text`	Character. The text displayed in the tooltip while loading
`zindex`	Numeric. The 'z-index' CSS property to assign to the tooltip. Components with higher values will be displayed on top of components with lower values.
`targetable`	Logical. Whether the tooltip itself can be targeted by pointer events. For tooltips triggered by hover events, typically this should be left 'false' to avoid the tooltip interfering with those same events.
`loading_state`	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

---

`dccUpload` *Upload component*

Description

Upload components allow your app to accept user-uploaded files via drag'n'drop

Usage

```
dccUpload(children=NULL, id=NULL, contents=NULL, filename=NULL,
last_modified=NULL, accept=NULL, disabled=NULL,
disable_click=NULL, max_size=NULL, min_size=NULL,
multiple=NULL, className=NULL, className_active=NULL,
className_reject=NULL, className_disabled=NULL, style=NULL,
style_active=NULL, style_reject=NULL, style_disabled=NULL,
loading_state=NULL)
```

Arguments

children	A list of or a singular dash component, string or number | character. Contents of the upload component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
contents	Character | list of characters. The contents of the uploaded file as a binary string
filename	Character | list of characters. The name of the file(s) that was(were) uploaded. Note that this does not include the path of the file (for security reasons).
last_modified	Numeric | list of numerics. The last modified date of the file that was uploaded in unix time (seconds since 1970).
accept	Character. Allow specific types of files. See https://github.com/okonet/attr-accept for more information. Keep in mind that mime type determination is not reliable across platforms. CSV files, for example, are reported as text/plain under macOS but as application/vnd.ms-excel under Windows. In some cases there might not be a mime type set at all. See: https://github.com/react-dropzone/react-dropzone/issues/276
disabled	Logical. Enable/disable the upload component entirely
disable_click	Logical. Disallow clicking on the component to open the file dialog
max_size	Numeric. Maximum file size in bytes. If '-1', then infinite
min_size	Numeric. Minimum file size in bytes
multiple	Logical. Allow dropping multiple files
className	Character. HTML class name of the component
className_active	Character. HTML class name of the component while active
className_reject	Character. HTML class name of the component if rejected
className_disabled	Character. HTML class name of the component if disabled
style	Named list. CSS styles to apply
style_active	Named list. CSS styles to apply while active
style_reject	Named list. CSS styles if rejected
style_disabled	Named list. CSS styles if disabled
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
    library(dash)
    library(jsonlite)

    app <- Dash$new()

    app$layout(htmlDiv(list(
      dccUpload(
        id='upload-image',
        children=htmlDiv(list(
          'Drag and Drop or ',
          htmlA('Select Files')
        )),
        style=list(
          'height'= '60px',
          'lineHeight'= '60px',
          'borderWidth'= '1px',
          'borderStyle'= 'dashed',
          'borderRadius'= '5px',
          'textAlign'= 'center',
          'margin'= '10px'
        ),
        # Allow multiple files to be uploaded
        multiple=TRUE
      ),
      htmlDiv(id='output-image-upload')
    )))

    parse_content = function(contents, filename, date) {
      return(htmlDiv(list(
        htmlH5(filename),
        htmlH6(as.POSIXct(date, origin="1970-01-01")),
        htmlImg(src=contents),
        htmlHr(),
        htmlDiv('Raw Content'),
        htmlPre(paste(substr(toJSON(contents), 1, 100), "..."), style=list(
          'whiteSpace'= 'pre-wrap',
          'wordBreak'= 'break-all'
        ))
      )))
    }

    app$callback(
      output = list(id='output-image-upload', property = 'children'),
      params = list(input(id = 'upload-image', property = 'contents'),
                    state(id = 'upload-image', property = 'filename'),
                    state(id = 'upload-image', property = 'last_modified')),
      function(list_of_contents, list_of_names, list_of_dates) {
     if (!is.null(list_of_contents) && !is.null(list_of_names) && !is.null(list_of_dates[[1]])) {
          children = lapply(1:length(list_of_contents), function(x){
            parse_content(list_of_contents[[x]], list_of_names[[x]], list_of_dates[[x]])
```

```
            })
          }
          else {
            children = "Upload a file to see the raw data."
          }
          return(children)
        }
      )

  app$run_server()
}
```

---

dependencies *Input/Output/State definitions*

Description

Use in conjunction with the `callback()` method from the Dash class to define the update logic in your application.

Usage

```
output(id, property)

input(id, property)

state(id, property)

dashNoUpdate()
```

Arguments

`id`	a component id
`property`	the component property to use

Details

The `dashNoUpdate()` function permits application developers to prevent a single output from updating the layout. It has no formal arguments.

df_to_list *df_to_list*

Description

Helper function to convert a dataframe into the nested list format required for input into Dash DataTable.

Usage

```
df_to_list(df)
```

Arguments

df	A data frame object to be coerced into a list of lists for DataTable.

htmlA *A component*

Description

A is a wrapper for the <a> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a

Usage

```
htmlA(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, download=NULL,
href=NULL, hrefLang=NULL, media=NULL, referrerPolicy=NULL,
rel=NULL, shape=NULL, target=NULL, accessKey=NULL,
className=NULL, contentEditable=NULL, contextMenu=NULL,
dir=NULL, draggable=NULL, hidden=NULL, lang=NULL,
spellCheck=NULL, style=NULL, tabIndex=NULL, title=NULL,
loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

download
: Character. Indicates that the hyperlink is to be used for downloading a resource.

href
: Character. The URL of a linked resource.

hrefLang
: Character. Specifies the language of the linked resource.

media
: Character. Specifies a hint of the media for which the linked resource was designed.

referrerPolicy
: Character. Specifies which referrer is sent when fetching the resource.

rel
: Character. Specifies the relationship of the target object to the link object.

shape
: Character.

target
: Character. Specifies where to open the linked document (in the case of an <a> element) or where to display the response received (in the case of a <form> element)

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'‘

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlA(children='Link to external site',
            href='https://plotly.com',
            target='_blank')
      )
    )
  )

  app$run_server()
}
```

---

htmlAbbr *Abbr component*

Description

Abbr is a wrapper for the <abbr> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/abbr

Usage

```
htmlAbbr(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
```

```
    htmlDiv(list(
      htmlAbbr(children='Hello! htmlAbbr at work!',
    title='\U{1F50D} Hover over this line for a few seconds and see the text box appear...')
      )
    )
  )

  app$run_server()
}
```

`htmlAcronym` *Acronym component*

Description

Acronym is a wrapper for the <acronym> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/acronym

Usage

```
htmlAcronym(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlAcronym(children='ASAP',
      title='Mouse over these words to see the acronym for \'as soon as possible\'.')
        )
    )
  )

  app$run_server()
}
```

`htmlAddress` *Address component*

Description

Address is a wrapper for the <address> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/address

Usage

```
htmlAddress(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
`dir`	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
`draggable`	Character. Defines whether the element can be dragged.
`hidden`	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlAddress(children='5555 Avenue de Gaspe, Montreal QC H2T 2A3')
      )
    )
  )

  app$run_server()
}
```

---

htmlArea *Area component*

Description

Area is a wrapper for the <area> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/area

Usage

```
htmlArea(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, alt=NULL,
coords=NULL, download=NULL, href=NULL, hrefLang=NULL,
media=NULL, referrerPolicy=NULL, rel=NULL, shape=NULL,
target=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
alt	Character. Alternative text in case an image can't be displayed.
coords	Character. A set of values specifying the coordinates of the hot-spot region.
download	Character. Indicates that the hyperlink is to be used for downloading a resource.
href	Character. The URL of a linked resource.
hrefLang	Character. Specifies the language of the linked resource.
media	Character. Specifies a hint of the media for which the linked resource was designed.
referrerPolicy	Character. Specifies which referrer is sent when fetching the resource.
rel	Character. Specifies the relationship of the target object to the link object.
shape	Character.
target	Character. Specifies where to open the linked document (in the case of an <a> element) or where to display the response received (in the case of a <form> element)
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# The URL below has been chunked to comply with CRAN
# requirements; the use of file.path is optional and not required
# for this component.
if (interactive()) {
  library(dash)

  app$layout(
    htmlDiv(list(
      htmlImg(src = file.path('https://upload.wikimedia.org',
              'wikipedia/commons/0/0c',
              'PIA17351-ApparentSizes-MarsDeimosPhobos-EarthMoon.jpg',
              fsep = '/'),
              useMap = '#image-map'),
      htmlMapEl(list(
        htmlArea(target='_blank',
                 alt='Deimos',
                 title='Deimos',
                 href='https://en.wikipedia.org/wiki/Deimos_(moon)',
```

```
                coords='5,114,32,147',
                shape='rect'),
        htmlArea(target='_blank',
                alt='Phobos',
                title='Phobos',
                href='https://en.wikipedia.org/wiki/Phobos_(moon)',
                coords='113,196,32,103',
                shape='rect'),
        htmlArea(target='_blank',
                alt='Moon',
                title='Moon',
                href='https://en.wikipedia.org/wiki/Moon',
                coords='127,285,294,1',
                shape='rect')
        ),
        name = 'image-map'
      ),
      htmlDiv(children = 'Click on the image to visit a Wikipedia article',
              id = 'object-name')
      )
    )
  )

  app$run_server()
}
```

`htmlArticle` *Article component*

Description

Article is a wrapper for the <article> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/article

Usage

```
htmlArticle(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)
```

```
  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlArticle(list(
        htmlH2('Dash for R launched!'),
        htmlP('Dash is a user interface library for creating analytical\n
              web applications. Those who use R for data analysis, data\n
              exploration, visualization, modelling, instrument control,\n
              and reporting will find immediate use for Dash for R.'),
        htmlAside('Plotly is a technical computing company with offices\n
                   in Montreal, Canada and Cambridge, Massachusetts.')
        )
       )
    )
    )
  )

  app$run_server()
}
```

`htmlAside` *Aside component*

Description

Aside is a wrapper for the <aside> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/aside

Usage

```
htmlAside(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
```

```
    htmlDiv(list(
      htmlArticle(list(
        htmlH2('Dash for R launched!'),
        htmlP('Dash is a user interface library for creating analytical\n
              web applications. Those who use R for data analysis, data\n
              exploration, visualization, modelling, instrument control,\n
              and reporting will find immediate use for Dash for R.'),
        htmlAside('Plotly is a technical computing company with offices\n
                  in Montreal, Canada and Cambridge, Massachusetts.')
        )
      )
    )
    )
  )

  app$run_server()
}
```

`htmlAudio` *Audio component*

Description

Audio is a wrapper for the <audio> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio

Usage

```
htmlAudio(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, autoPlay=NULL,
controls=NULL, crossOrigin=NULL, loop=NULL, muted=NULL,
preload=NULL, src=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
autoPlay	A value equal to: 'autoplay', 'autoplay', 'autoplay' | logical. The audio or video should play as soon as possible.
controls	A value equal to: 'controls', 'controls' | logical. Indicates whether the browser should show playback controls to the user.
crossOrigin	Character. How the element handles cross-origin requests
loop	A value equal to: 'loop', 'loop' | logical. Indicates whether the media should start playing from the start when it's finished.
muted	A value equal to: 'muted', 'muted' | logical. Indicates whether the audio will be initially silenced on page load.
preload	Character. Indicates whether the whole resource, parts of it or nothing should be preloaded.
src	Character. The URL of the embeddable content.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlAudio(src='https://www.nasa.gov/62284main_onesmall2.wav',
                controls=TRUE,
                title='Apollo 11 - July 16, 1969 - Neil Armstrong')
        )
    )
  )

  app$run_server()
}
```

---

htmlB *B component*

Description

B is a wrapper for the <b> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/b

Usage

```
htmlB(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
```

```
    htmlDiv(list(
      htmlB(children="This is a bold statement!"),
      htmlP(children="This is not so bold.")
      )
    )
  )

  app$run_server()
}
```

---

htmlBase *Base component*

Description

Base is a wrapper for the <base> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/base

Usage

```
htmlBase(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, href=NULL,
target=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
href	Character. The URL of a linked resource.

target	Character. Specifies where to open the linked document (in the case of an <a> element) or where to display the response received (in the case of a <form> element)
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlBase(href="https://www.w3schools.com/"),
      htmlA(children="HTML base tag", href="tags/tag_base.asp")
      )
    )
  )
```

```
  app$run_server()
}
```

`htmlBasefont` *Basefont component*

Description

Basefont is a wrapper for the <basefont> HTML5 element. OBSOLETE: <basefont> is included for completeness, but should be avoided as it is only supported by Internet Explorer. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/basefont

Usage

```
htmlBasefont(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  # This feature is obsolete. It may still work in some
  # browsers, but could stop working at any time. Try to
  # avoid using this component.
  #
  # Instead, use CSS properties to set font, font-family,
  # font-size and color.
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlBasefont(color="FF0000",
                   face="Helvetica",
                   size="+2"),
     htmlP(children="If it works, this will be Helvetica but a couple point sizes larger.")
      )
    )
  )

  app$run_server()
}
```

htmlBdi *Bdi component*

Description

Bdi is a wrapper for the <bdi> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/bdi

Usage

```
htmlBdi(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
      htmlDiv(list(
          htmlP(children="This text is 'Aladdin', but in Arabic script:"),
          htmlBdi(children=paste0("\U{0639}\U{0644}\U{0627}\U{0621}",
                                  "\U{0627}\U{0644}\U{062F}\U{064A}\U{0646}"))
      )
      )
  )

  app$run_server()
}
```

---

htmlBdo *Bdo component*

Description

Bdo is a wrapper for the <bdo> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/bdo

Usage

```
htmlBdo(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  # This element supports bidirectional text override.
  # We can force text to render from right to left instead
  # of left to right.
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP(children="This text will print from left to right."),
      htmlP(children="Below, we use bidirectional override to print right to left:"),
      htmlBdo(children="This text will print from right to left.",
              dir="rtl")
    )
    )
  )

  app$run_server()
}
```

---

htmlBig *Big component*

Description

Big is a wrapper for the <big> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/big

Usage

```
htmlBig(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
```

```
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  # The <big> tag is not supported in HTML5.
  # Instead, use the font-size property in
  # CSS to enlarge text.
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP(children="This text will display in standard size."),
      htmlBig(children="This text may or may not appear slightly larger.")
    )
    )
  )

  app$run_server()
}
```

`htmlBlink` *Blink component*

Description

Blink is a wrapper for the <blink> HTML5 element. OBSOLETE: <blink> is included for completeness, but should be avoided as it is not supported by any modern browsers. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blink

Usage

```
htmlBlink(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  # The blink tag is now obsolete and deprecated.
  # It may not function properly in all browsers,
```

```
  # and it may cease working without warning.
  #
  # This element is generally unsupported on all
  # modern browser releases.
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP(children="Here is a bit of text."),
      htmlBlink(children="Here is a bit of blinking text.")
    )
    )
  )

  app$run_server()
}
```

`htmlBlockquote` *Blockquote component*

Description

Blockquote is a wrapper for the <blockquote> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote

Usage

```
htmlBlockquote(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, cite=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
cite	Character. Contains a URI which points to the source of the quote or change.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP("Here is some text."),
      htmlBlockquote(children=list(
```

```
        htmlP("And here is a quotation in block format.")
        )
      )
    )
    )
  )

  app$run_server()
}
```

---

htmlBr *Br component*

Description

Br is a wrapper for the
 HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/br

Usage

```
htmlBr(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'‘

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP("Here is some text."),
      htmlBr(),
      htmlP("Here is additional text."),
      htmlBr(),
      htmlP("See the gap in between the lines?")
      )
    )
  )

  app$run_server()
}
```

htmlButton *Button component*

Description

Button is a wrapper for the <button> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/button

Usage

```
htmlButton(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
autoFocus=NULL, disabled=NULL, form=NULL, formAction=NULL,
formEncType=NULL, formMethod=NULL, formNoValidate=NULL,
formTarget=NULL, name=NULL, type=NULL, value=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
autoFocus	A value equal to: 'autofocus', 'autofocus', 'autofocus' | logical. The element should be automatically focused after the page loaded.
disabled	A value equal to: 'disabled', 'disabled' | logical. Indicates whether the user can interact with the element.
form	Character. Indicates the form that is the owner of the element.
formAction	Character. Indicates the action of the element, overriding the action defined in the <form>.

formEncType Character. If the button/input is a submit button (type="submit"), this attribute sets the encoding type to use during form submission. If this attribute is specified, it overrides the enctype attribute of the button's form owner.

formMethod Character. If the button/input is a submit button (type="submit"), this attribute sets the submission method to use during form submission (GET, POST, etc.). If this attribute is specified, it overrides the method attribute of the button's form owner.

formNoValidate A value equal to: 'formnovalidate', 'formnovalidate', 'formnovalidate' | logical. If the button/input is a submit button (type="submit"), this boolean attribute specifies that the form is not to be validated when it is submitted. If this attribute is specified, it overrides the novalidate attribute of the button's form owner.

formTarget Character. If the button/input is a submit button (type="submit"), this attribute specifies the browsing context (for example, tab, window, or inline frame) in which to display the response that is received after submitting the form. If this attribute is specified, it overrides the target attribute of the button's form owner.

name Character. Name of the element. For example used by the server to identify the fields in form submits.

type Character. Defines the type of the element.

value Character. Defines a default value which will be displayed in the element on page load.

accessKey Character. Keyboard shortcut to activate or add focus to the element.

className Character. Often used with CSS to style elements with common properties.

contentEditable
Character. Indicates whether the element's content is editable.

contextMenu Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable Character. Defines whether the element can be dragged.

hidden A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang Character. Defines the language used in the element.

spellCheck Character. Indicates whether spell checking is allowed for the element.

style Named list. Defines CSS styles which will override styles previously set.

tabIndex Character. Overrides the browser's default tab order and follows the one specified instead.

title Character. Text to be displayed in a tooltip when hovering over the element.

loading_state Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

... wildcards allowed have the form: ''data-*', 'aria-*'‘

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlButton("Click me!")
      )
    )
  )

  app$run_server()
}
```

`htmlCanvas` *Canvas component*

Description

Canvas is a wrapper for the <canvas> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas

Usage

```
htmlCanvas(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, height=NULL,
width=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

height
: Character | numeric. Specifies the height of elements listed here. For all other elements, use the CSS height property. Note: In some instances, such as , this is a legacy attribute, in which case the CSS height property should be used instead.

width
: Character | numeric. For the elements listed here, this establishes the element's width. Note: For all other instances, such as , this is a legacy attribute, in which case the CSS width property should be used instead.

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'‘

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  # this component requires JavaScript code to draw on the canvas
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlCanvas(id="canvas-component")
      )
    )
  )

  app$run_server()
}
```

`htmlCaption` *Caption component*

Description

Caption is a wrapper for the <caption> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/caption

Usage

```
htmlCaption(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlTable(list(
        htmlCaption("Elevations of a few Cascade Range volcanoes"),
        htmlThead(
```

```
            htmlTr(list(
              htmlTh("Mountain"),
              htmlTh("Elevation (m)"),
              htmlTh("Elevation (ft)")
            )
          )),
          htmlTbody(list(
            htmlTr(list(
              htmlTd("Mount Rainier"),
              htmlTd("4,392"),
              htmlTd("14,411")
            )
            ),
            htmlTr(list(
              htmlTd("Mount Hood"),
              htmlTd("3,429"),
              htmlTd("11,249")
            )
            ),
            htmlTr(list(
              htmlTd("Lassen Peak"),
              htmlTd("3,187"),
              htmlTd("10,457")
            )
            ),
            htmlTr(list(
              htmlTd("Mount St. Helens"),
              htmlTd("2,549"),
              htmlTd("8,363")
            )
            )
          )
          )
        ), style = list(
              border = "1px black solid"
        )
        )
      )
    )
    )

    app$run_server()
  }
```

`htmlCenter` *Center component*

Description

Center is a wrapper for the HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/center

Usage

```
htmlCenter(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlCenter("Centered text!")
      )
    )
  )

  app$run_server()
}
```

---

htmlCite *Cite component*

Description

Cite is a wrapper for the <cite> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/cite

Usage

```
htmlCite(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlCite("Click me!")
      )
    )
  )

  app$run_server()
}
```

`htmlCode` *Code component*

Description

Code is a wrapper for the <code> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/code

Usage

```
htmlCode(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlCode(
      children = 'cat("Hello world!")'
        )
      )
    )
  )

  app$run_server()
}
```

htmlCol *Col component*

Description

Col is a wrapper for the <col> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/col

Usage

```
htmlCol(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, span=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
span	Character.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.

hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  # Used within htmlColgroup to define columns.
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlTable(list(
          htmlColgroup(
            list(
              htmlCol(span = 2, style = list("background-color"= "red"))
            )
          ),
          htmlTr(
            list(
              htmlTd("Cell A"),
              htmlTd("Cell B"),
              htmlTd("Cell C")
            )
          )
        ))
      )
    )
  )

  app$run_server()
}
```

htmlColgroup *Colgroup component*

Description

Colgroup is a wrapper for the <colgroup> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/colgroup

Usage

```
htmlColgroup(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, span=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
span	Character.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.

hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlTable(list(
          htmlColgroup(
            list(
              htmlCol(span = 2, style = list("background-color"= "red"))
            )
          ),
          htmlTr(
            list(
              htmlTd("Cell A"),
              htmlTd("Cell B"),
              htmlTd("Cell C")
            )
          )
        ))
      )
    )
  )

  app$run_server()
}
```

htmlContent *Content component*

Description

Content is a wrapper for the <content> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/content

Usage

```
htmlContent(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# This feature is obsolete and no longer supported. It is recommended
# that you use the htmlSlot component instead.
```

---

htmlData *Data component*

Description

Data is a wrapper for the <data> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/data

Usage

```
htmlData(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, value=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
value	Character. Defines a default value which will be displayed in the element on page load.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlUl(list(
          htmlLi(list(htmlData(value = 398, "First Element"))),
          htmlLi(list(htmlData(value = 399, "Second Element"))),
          htmlLi(list(htmlData(value = 400, "First Element")))
        ))
      )
    )
  )

  # Include the following in a separate CSS file in an
  # `assets` directory in the root of your app.
  #
  # data:hover::after {
  #   content: ' (ID ' attr(value) ')';
  #   font-size: .7em;
  # }

  app$run_server()
}
```

---

htmlDatalist *Datalist component*

Description

Datalist is a wrapper for the <datalist> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/datalist

Usage

```
htmlDatalist(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    dccInput(
      placeholder = 'Enter here',
      list = 'list-of-options'),
    htmlDatalist(id = 'list-of-options',
      children=list(
        htmlOption("Option 1"),
        htmlOption("Option 2"),
        htmlOption("Option 3")
          )
        )
      )
    )
  )

  app$run_server()
}
```

`htmlDd` *Dd component*

Description

Dd is a wrapper for the <dd> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dd

Usage

```
htmlDd(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.

n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)
```

```
  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlDl(
      children= list(htmlDt("Dash for R"),
                     htmlDd('HtmlDt and htmlDD must be used
                             within htmlDl'))
      )
    )
  )

  app$run_server()
}
```

`htmlDel` *Del component*

Description

Del is a wrapper for the <del> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/del

Usage

```
htmlDel(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, cite=NULL,
dateTime=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role	Character. The ARIA role attribute
cite	Character. Contains a URI which points to the source of the quote or change.
dateTime	Character. Indicates the date and time associated with the element.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlDel(
          children ="Deleted Hello"
        )
      )
    )
```

```
  )

  app$run_server()
}
```

`htmlDetails` *Details component*

Description

Details is a wrapper for the <details> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/details

Usage

```
htmlDetails(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, open=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`open`	A value equal to: 'open', 'open' | logical. Indicates whether the the contents are currently visible (in the case of a <details> element) or whether the dialog is active and can be interacted with (in the case of a <dialog> element).
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.

contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlDetails(
        children = list(
          htmlSummary(
        children = "Within a details element, the summary can act as a clickable description"
          ),
          "And the rest is hidden until the summary is clicked"
        )
      )
      )
    )
  )

  app$run_server()
}
```

htmlDfn *Dfn component*

Description

Dfn is a wrapper for the <dfn> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dfn

Usage

```
htmlDfn(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlDfn(
          children ="Hello"
        )
      )
    )
  )

  app$run_server()
}
```

`htmlDialog` *Dialog component*

Description

Dialog is a wrapper for the <dialog> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dialog

Usage

```
htmlDialog(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, open=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
open	A value equal to: 'open', 'open' | logical. Indicates whether the the contents are currently visible (in the case of a <details> element) or whether the dialog is active and can be interacted with (in the case of a <dialog> element).
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.

title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlDialog(
      children = htmlP('Greetings')
        )
      )
    )
  )

  app$run_server()
}
```

---

htmlDiv *Div component*

Description

Div is a wrapper for the HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div

Usage

```
htmlDiv(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlDiv('This Title is Wrapped inside an inner Div')
      )
    )
  )

  app$run_server()
}
```

`htmlDl` *Dl component*

Description

Dl is a wrapper for the <dl> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dl

Usage

```
htmlDl(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlDl(
      children= list(htmlDt("Dash for R"),
                     htmlDd('HtmlDt and htmlDD must be used
                             within htmlDl'))
      )
    )
  )
```

```
  app$run_server()
}
```

`htmlDt` *Dt component*

Description

Dt is a wrapper for the <dt> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dt

Usage

```
htmlDt(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
`dir`	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlDl(
      children= list(htmlDt("Dash for R"),
                     htmlDd('HtmlDt and htmlDD must be used
                             within htmlDl'))
        )
      )
    )
  )

  app$run_server()
}
```

htmlEm *Em component*

Description

Em is a wrapper for the HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/em

Usage

```
htmlEm(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlH1(list('Dash is a very ',
                htmlEm(' important '),
                'framework')
        )
      )
    )
  )

  app$run_server()
}
```

---

htmlEmbed *Embed component*

Description

Embed is a wrapper for the <embed> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/embed

Usage

```
htmlEmbed(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, height=NULL,
src=NULL, type=NULL, width=NULL, accessKey=NULL,
className=NULL, contentEditable=NULL, contextMenu=NULL,
dir=NULL, draggable=NULL, hidden=NULL, lang=NULL,
spellCheck=NULL, style=NULL, tabIndex=NULL, title=NULL,
loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
height	Character | numeric. Specifies the height of elements listed here. For all other elements, use the CSS height property. Note: In some instances, such as , this is a legacy attribute, in which case the CSS height property should be used instead.
src	Character. The URL of the embeddable content.
type	Character. Defines the type of the element.
width	Character | numeric. For the elements listed here, this establishes the element's width. Note: For all other instances, such as , this is a legacy attribute, in which case the CSS width property should be used instead.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.

hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlEmbed(
      src = 'https://archive.org/embed/VintageCartoonsSet1Mp4',
      width = '500',
      height = '500')
      )
    )
  )

  app$run_server()
}
```

---

htmlFieldset *Fieldset component*

Description

Fieldset is a wrapper for the <fieldset> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/fieldset

Usage

```
htmlFieldset(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, disabled=NULL,
form=NULL, name=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
disabled	A value equal to: 'disabled', 'disabled' | logical. Indicates whether the user can interact with the element.
form	Character. Indicates the form that is the owner of the element.
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.

style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlFieldset(
      children = list('Choose your favorite Dash HTML component',
      dccRadioItems(
        options=list(
          list("label"= "htmlDiv", "value"= "htmlDiv"),
          list("label"= "htmlBase", "value"= "htmlBase"),
          list("label"= "htmlArticle", "value"= "htmlArticle")
              )
            )
          )
        )
      )
    )
  )

  app$run_server()
}
```

---

htmlFigcaption *Figcaption component*

Description

Figcaption is a wrapper for the <figcaption> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/figcaption

Usage

```
htmlFigcaption(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlFigure(children = list(
    htmlImg(src = 'https://brand.plotly.com/static/images/plotly-logo-01-stripe@2x.png'),
      htmlFigcaption(children = 'Plotly Logo')))
      )
    )
  )

  app$run_server()
}
```

---

htmlFigure *Figure component*

Description

Figure is a wrapper for the <figure> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/figure

Usage

```
htmlFigure(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlFigure(children = list(
     htmlImg(src = 'https://brand.plotly.com/static/images/plotly-logo-01-stripe@2x.png',
               width = '400',
               height = '150')
          )
        )
      )
    )
  )

  app$run_server()
}
```

`htmlFont` *Font component*

Description

Font is a wrapper for the <font> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/font

Usage

```
htmlFont(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# Starting with HTML 4, HTML does not convey styling information
# anymore (outside the <style> element or the style attribute of each
# element). CSS should be used for styling instead.
```

htmlFooter *Footer component*

Description

Footer is a wrapper for the <footer> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/footer

Usage

```
htmlFooter(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlFooter(list(
      htmlH1('Dash'),
      htmlLi('Pointer1'),
      htmlLi('Pointer2')
          )
        )
      )
    )
  )

  app$run_server()
}
```

---

htmlForm *Form component*

Description

Form is a wrapper for the <form> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form

Usage

```
htmlForm(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, accept=NULL,
acceptCharset=NULL, action=NULL, autoComplete=NULL,
encType=NULL, method=NULL, name=NULL, noValidate=NULL,
target=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accept	Character. List of types the server accepts, typically a file type.
acceptCharset	Character. List of supported charsets.
action	Character. The URI of a program that processes the information submitted via the form.
autoComplete	Character. Indicates whether controls in this form can by default have their values automatically completed by the browser.
encType	Character. Defines the content type of the form data when the method is POST.
method	Character. Defines which HTTP method to use when submitting the form. Can be GET (default) or POST.
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
noValidate	A value equal to: 'novalidate', 'novalidate', 'novalidate' | logical. This attribute indicates that the form shouldn't be validated when submitted.
target	Character. Specifies where to open the linked document (in the case of an <a> element) or where to display the response received (in the case of a <form> element)
accessKey	Character. Keyboard shortcut to activate or add focus to the element.

`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
`dir`	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
`draggable`	Character. Defines whether the element can be dragged.
`hidden`	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
`lang`	Character. Defines the language used in the element.
`spellCheck`	Character. Indicates whether spell checking is allowed for the element.
`style`	Named list. Defines CSS styles which will override styles previously set.
`tabIndex`	Character. Overrides the browser's default tab order and follows the one specified instead.
`title`	Character. Text to be displayed in a tooltip when hovering over the element.
`loading_state`	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
`...`	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlForm(children=list(
      htmlP(children=list('Username: ',
                          dccInput(type='text',
                                   id='username',
                                   placeholder='username'))),
      htmlP(children=list('Password: ',
                          dccInput(type='password',
                                   id='password',
                                   placeholder='password'))),
      htmlButton(children=list('Login'),
                 type='submit',
```

```
                  id='login_button')
          )
        )
      )
    )
  )

  app$run_server()
}
```

`htmlFrame` *Frame component*

Description

Frame is a wrapper for the <frame> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/frame

Usage

```
htmlFrame(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'‘

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# htmlFrame is now deprecated. htmlIFrame is recommended instead.
```

---

htmlFrameset *Frameset component*

Description

Frameset is a wrapper for the <frameset> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/frameset

Usage

```
htmlFrameset(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# htmlFrameset is now deprecated. htmlIFrame is recommended instead.
```

`htmlH1` *H1 component*

Description

H1 is a wrapper for the <h1> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h1

Usage

```
htmlH1(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
`dir`	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlH1(list(
        'Dash Html',
        htmlBr(), #We can customize
        htmlSpan('Dash', style = list('opacity' = '0.8')),
        htmlSpan(' Core')))
      )
    )
  )

  app$run_server()
}
```

htmlH2 *H2 component*

Description

H2 is a wrapper for the <h2> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h2

Usage

```
htmlH2(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlH2(list(
        'Dash Html',
        htmlBr(), #We can customize
        htmlSpan('Dash', style = list('opacity' = '0.8')),
        htmlSpan(' Core')))
      )
    )
  )

  app$run_server()
}
```

---

htmlH3 *H3 component*

Description

H3 is a wrapper for the <h3> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h3

Usage

```
htmlH3(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlH3(list(
        'Dash Html',
        htmlBr(), #We can customize
        htmlSpan('Dash', style = list('opacity' = '0.8')),
        htmlSpan(' Core')))
      )
    )
  )

  app$run_server()
}
```

---

htmlH4 *H4 component*

Description

H4 is a wrapper for the <h4> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h4

Usage

```
htmlH4(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlH4(list(
        'Dash Html',
        htmlBr(), #We can customize
        htmlSpan('Dash', style = list('opacity' = '0.8')),
        htmlSpan(' Core')))
      )
    )
  )

  app$run_server()
}
```

`htmlH5` *H5 component*

Description

H5 is a wrapper for the <h5> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h5

Usage

```
htmlH5(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
```

```
    htmlDiv(list(
      htmlH5(list(
        'Dash Html',
        htmlBr(), #We can customize
        htmlSpan('Dash', style = list('opacity' = '0.8')),
        htmlSpan(' Core')))
      )
    )
  )

  app$run_server()
}
```

`htmlH6` *H6 component*

Description

H6 is a wrapper for the <h6> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h6

Usage

```
htmlH6(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.

`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
`dir`	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
`draggable`	Character. Defines whether the element can be dragged.
`hidden`	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
`lang`	Character. Defines the language used in the element.
`spellCheck`	Character. Indicates whether spell checking is allowed for the element.
`style`	Named list. Defines CSS styles which will override styles previously set.
`tabIndex`	Character. Overrides the browser's default tab order and follows the one specified instead.
`title`	Character. Text to be displayed in a tooltip when hovering over the element.
`loading_state`	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
`...`	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlH6(list(
        'Dash Html',
        htmlBr(), #We can customize
        htmlSpan('Dash', style = list('opacity' = '0.8')),
        htmlSpan(' Core')))
      )
    )
  )

  app$run_server()
}
```

`htmlHeader` *Header component*

Description

Header is a wrapper for the <header> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/header

Usage

```
htmlHeader(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
`dir`	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
`draggable`	Character. Defines whether the element can be dragged.
`hidden`	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlHeader("This is a header"),
      htmlP("And here is some text.")
    )
  )
  )

  app$run_server()
}
```

---

htmlHgroup *Hgroup component*

Description

Hgroup is a wrapper for the <hgroup> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/hgroup

Usage

```
htmlHgroup(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlHgroup(list(
        htmlH1('MultiLevel Title'),
        htmlHr(),
        htmlH2('Header')
          )
        )
      )
    )
  )

  app$run_server()
}
```

---

htmlHr *Hr component*

Description

Hr is a wrapper for the <hr> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/hr

Usage

```
htmlHr(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlDiv(list(
        htmlH1('Dash'),
        htmlHr(),
        htmlH2('Components')
          )
        )
      )
    )
  )

  app$run_server()
}
```

---

htmlI *I component*

Description

I is a wrapper for the <i> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/i

Usage

```
htmlI(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
```

```
    htmlDiv(list(
        htmlI('Italicized Text')
      )
    )
  )

  app$run_server()
}
```

`htmlIframe` *Iframe component*

Description

Iframe is a wrapper for the <iframe> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe

Usage

```
htmlIframe(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, allow=NULL,
height=NULL, name=NULL, referrerPolicy=NULL, sandbox=NULL,
src=NULL, srcDoc=NULL, width=NULL, accessKey=NULL,
className=NULL, contentEditable=NULL, contextMenu=NULL,
dir=NULL, draggable=NULL, hidden=NULL, lang=NULL,
spellCheck=NULL, style=NULL, tabIndex=NULL, title=NULL,
loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`allow`	Character. Specifies a feature-policy for the iframe.

height	Character | numeric. Specifies the height of elements listed here. For all other elements, use the CSS height property. Note: In some instances, such as , this is a legacy attribute, in which case the CSS height property should be used instead.
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
referrerPolicy	Character. Specifies which referrer is sent when fetching the resource.
sandbox	Character. Stops a document loaded in an iframe from using certain features (such as submitting forms or opening new windows).
src	Character. The URL of the embeddable content.
srcDoc	Character.
width	Character | numeric. For the elements listed here, this establishes the element's width. Note: For all other instances, such as , this is a legacy attribute, in which case the CSS width property should be used instead.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlIframe(width = "600px", height = "600px",
               src = "https://dashr.plotly.com/")
      )
    )
  )

  app$run_server()
}
```

`htmlImg` *Img component*

Description

Img is a wrapper for the <img> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img

Usage

```
htmlImg(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, alt=NULL,
crossOrigin=NULL, height=NULL, referrerPolicy=NULL,
sizes=NULL, src=NULL, srcSet=NULL, useMap=NULL, width=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
alt	Character. Alternative text in case an image can't be displayed.
crossOrigin	Character. How the element handles cross-origin requests
height	Character | numeric. Specifies the height of elements listed here. For all other elements, use the CSS height property. Note: In some instances, such as , this is a legacy attribute, in which case the CSS height property should be used instead.
referrerPolicy	Character. Specifies which referrer is sent when fetching the resource.
sizes	Character.
src	Character. The URL of the embeddable content.
srcSet	Character. One or more responsive image candidates.
useMap	Character.
width	Character | numeric. For the elements listed here, this establishes the element's width. Note: For all other instances, such as , this is a legacy attribute, in which case the CSS width property should be used instead.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
     htmlImg(src ='https://brand.plotly.com/static/images/plotly-logo-01-stripe@2x.png',
              height='200', width='400')
      )
    )
  )


  app$run_server()
}
```

---

htmlIns *Ins component*

Description

Ins is a wrapper for the <ins> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ins

Usage

```
htmlIns(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, cite=NULL,
dateTime=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

cite
: Character. Contains a URI which points to the source of the quote or change.

dateTime
: Character. Indicates the date and time associated with the element.

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)
```

```
  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlIns('This text has been inserted')
      )
    )
  )

  app$run_server()
}
```

---

htmlKbd *Kbd component*

Description

Kbd is a wrapper for the <kbd> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/kbd

Usage

```
htmlKbd(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.

`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
`dir`	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
`draggable`	Character. Defines whether the element can be dragged.
`hidden`	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
`lang`	Character. Defines the language used in the element.
`spellCheck`	Character. Indicates whether spell checking is allowed for the element.
`style`	Named list. Defines CSS styles which will override styles previously set.
`tabIndex`	Character. Overrides the browser's default tab order and follows the one specified instead.
`title`	Character. Text to be displayed in a tooltip when hovering over the element.
`loading_state`	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
`...`	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP('Please Press: '),
      htmlKbd(list(
        'Ctl + ',
        'Alt + ',
        'Delete'))
      )
    )
  )

  app$run_server()
}
```

`htmlKeygen` *Keygen component*

Description

Keygen is a wrapper for the <keygen> HTML5 element. DEPRECATED: <keygen> is included for completeness, but should be avoided as it is not supported by all browsers and may be removed at any time from those that do support it. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/keygen

Usage

```
htmlKeygen(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
autoFocus=NULL, challenge=NULL, disabled=NULL, form=NULL,
keyType=NULL, name=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`autoFocus`	A value equal to: 'autofocus', 'autofocus', 'autofocus' | logical. The element should be automatically focused after the page loaded.
`challenge`	Character. A challenge string that is submitted along with the public key.
`disabled`	A value equal to: 'disabled', 'disabled' | logical. Indicates whether the user can interact with the element.
`form`	Character. Indicates the form that is the owner of the element.
`keyType`	Character. Specifies the type of key generated.

name	Character. Name of the element. For example used by the server to identify the fields in form submits.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# This feature is obsolete. It may still work in some
# browsers, but could stop working at any time. Try to
# avoid using this component.
```

htmlLabel *Label component*

Description

Label is a wrapper for the <label> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/label

Usage

```
htmlLabel(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, htmlFor=NULL,
form=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
htmlFor	Character. Describes elements which belongs to this one.
form	Character. Indicates the form that is the owner of the element.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(
      htmlLabel(
        list(htmlDiv(list("Time points")),
            dccInput(
              id = "times-input",
              placeholder = "Enter a value...",
              type = "number",
              value = 1,
              min = 3,
              max = 999)
        )
      )
    )
  )

  app$run_server()
}
```

htmlLegend *Legend component*

Description

Legend is a wrapper for the <legend> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/legend

Usage

```
htmlLegend(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlFieldset(
      children = list(
                     htmlLegend('Select your favorite component'),
                     dccRadioItems(
                       options=list(
                         list("label"= "htmlDiv", "value"= "htmlDiv"),
                         list("label"= "htmlBase", "value"= "htmlBase"),
                         list("label"= "htmlArticle", "value"= "htmlArticle")
            )
          )
        )
      )
    )
  )

  app$run_server()
}
```

htmlLi *Li component*

Description

Li is a wrapper for the <li> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/li

Usage

```
htmlLi(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, value=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
value	Character. Defines a default value which will be displayed in the element on page load.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.

hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlOl(list(
        htmlLi("Montreal"),
        htmlLi("Toronto"),
        htmlLi("Halifax")
      )),
      htmlUl(list(
        htmlLi("Montreal"),
        htmlLi("Toronto"),
        htmlLi("Halifax")
      ))
    ))
  )

  app$run_server()
}
```

htmlLink *Link component*

Description

Link is a wrapper for the <link> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link

Usage

```
htmlLink(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
crossOrigin=NULL, href=NULL, hrefLang=NULL, integrity=NULL,
media=NULL, referrerPolicy=NULL, rel=NULL, sizes=NULL,
type=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
crossOrigin	Character. How the element handles cross-origin requests
href	Character. The URL of a linked resource.
hrefLang	Character. Specifies the language of the linked resource.
integrity	Character. Specifies a Subresource Integrity value that allows browsers to verify what they fetch.
media	Character. Specifies a hint of the media for which the linked resource was designed.
referrerPolicy	Character. Specifies which referrer is sent when fetching the resource.

rel	Character. Specifies the relationship of the target object to the link object.
sizes	Character.
type	Character. Defines the type of the element.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlLink(rel = "stylesheet",
               type = "text/css",
               href = "https://codepen.io/chriddyp/pen/bWLwgP.css")
    ))
```

```
  )

  app$run_server()
}
```

`htmlMain` *Main component*

Description

Main is a wrapper for the <main> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/main

Usage

```
htmlMain(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlMain(
        list(htmlH1("Benjamin Franklin"),
            htmlP("Benjamin Franklin was an American polymath
                  and one of the Founding Fathers of the United States.
                  Franklin was a leading author, printer, political theorist,
                  politician, Freemason, postmaster, scientist, inventor,
                  humorist, civic activist, statesman, and diplomat."))
      )
    ))
  )

  app$run_server()
}
```

htmlMapEl *MapEl component*

Description

MapEl is a wrapper for the <map> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/map

Usage

```
htmlMapEl(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, name=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.

hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# The URL below has been chunked to comply with CRAN
# requirements; the use of file.path is optional and not required
# for this component.
if (interactive()) {
  app$layout(
    htmlDiv(list(
      htmlImg(src = file.path('https://upload.wikimedia.org',
              'wikipedia/commons/0/0c',
              'PIA17351-ApparentSizes-MarsDeimosPhobos-EarthMoon.jpg',
              fsep = '/'),
              useMap = '#image-map'),
      htmlMapEl(list(
        htmlArea(target='_blank',
                 alt='Deimos',
                 title='Deimos',
                 href='https://en.wikipedia.org/wiki/Deimos_(moon)',
                 coords='5,114,32,147',
                 shape='rect'),
        htmlArea(target='_blank',
                 alt='Phobos',
                 title='Phobos',
                 href='https://en.wikipedia.org/wiki/Phobos_(moon)',
                 coords='113,196,32,103',
                 shape='rect'),
        htmlArea(target='_blank',
                 alt='Moon',
                 title='Moon',
```

```
                href='https://en.wikipedia.org/wiki/Moon',
                coords='127,285,294,1',
                shape='rect')
        ),
        name = 'image-map'
      ),
      htmlDiv(children = 'Click on the image to visit a Wikipedia article',
              id = 'object-name')
      )
    )
  )

  app$run_server()
}
```

`htmlMark` *Mark component*

Description

Mark is a wrapper for the <mark> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/mark

Usage

```
htmlMark(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP(list(
        htmlMark("Plotly"),
        " develops online data analytics and visualization tools."
      ))
    ))
  )

  app$run_server()
}
```

htmlMarquee *Marquee component*

Description

Marquee is a wrapper for the <marquee> HTML5 element. DEPRECATED: <marquee> is included for completeness, but should be avoided as browsers may remove it at any time. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/marquee

Usage

```
htmlMarquee(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, loop=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
loop	A value equal to: 'loop', 'loop' | logical. Indicates whether the media should start playing from the start when it's finished.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  # This feature is obsolete. It may still work in some
  # browsers, but could stop working at any time. Try to
  # avoid using this component.

  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlMarquee("Here is some sliding text that uses htmlMarquee")

    ))
  )

  app$run_server()
}
```

htmlMeta *Meta component*

Description

Meta is a wrapper for the <meta> HTML5 element. CAUTION: <meta> is included for completeness, but generally will not behave as expected since <meta> tags should be static HTML content in the <head> of the document. Dash components are dynamic <body> content. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta

Usage

```
htmlMeta(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, charSet=NULL,
content=NULL, httpEquiv=NULL, name=NULL, accessKey=NULL,
className=NULL, contentEditable=NULL, contextMenu=NULL,
dir=NULL, draggable=NULL, hidden=NULL, lang=NULL,
spellCheck=NULL, style=NULL, tabIndex=NULL, title=NULL,
loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
charSet	Character. Declares the character encoding of the page or script.
content	Character. A value associated with http-equiv or name depending on the context.
httpEquiv	Character. Defines a pragma directive.
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.

contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP("The <meta> tag provides metadata about the HTML document.
            Metadata will not be displayed on the page, but will be machine parsable.
            To view meta tag the content of this page can be inspected."),
      htmlMeta(name = "author", content = "Edward Tufte")
    ))
  )

  app$run_server()
}
```

htmlMeter *Meter component*

Description

Meter is a wrapper for the <meter> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meter

Usage

```
htmlMeter(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, form=NULL,
high=NULL, low=NULL, max=NULL, min=NULL, optimum=NULL,
value=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
form	Character. Indicates the form that is the owner of the element.
high	Character. Indicates the lower bound of the upper range.
low	Character. Indicates the upper bound of the lower range.
max	Character | numeric. Indicates the maximum value allowed.
min	Character | numeric. Indicates the minimum value allowed.
optimum	Character. Indicates the optimal numeric value.
value	Character. Defines a default value which will be displayed in the element on page load.

accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlLabel("Sample Level Meter: "),
      htmlMeter(id = "sample-meter",
                min = 0,
                max = 100,
                low = 33,
                high = 66,
                optimum = 80,
                value = 80
```

```
                )
    ))
  )

  app$run_server()
}
```

`htmlNav` *Nav component*

Description

Nav is a wrapper for the <nav> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/nav

Usage

```
htmlNav(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlNav(
        list(
          htmlA("Plotly", href = "https://plotly.com/"),
          "> ",
          htmlA("Dash", href = "https://plotly.com/dash"),
          "> ",
          htmlA("Request Trial", href = "https://go.plotly.com/dash-enterprise-trial")
        )
      )
    ))
  )

  app$run_server()
}
```

htmlNobr *Nobr component*

Description

Nobr is a wrapper for the <nobr> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/nobr

Usage

```
htmlNobr(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlNobr("Lorem ipsum dolor sit amet,
               consectetur adipiscing elit, sed do eiusmod
               tempor incididunt ut labore et dolore magna aliqua.
               Ut enim ad minim veniam, quis nostrud exercitation
               ullamco laboris nisi ut aliquip ex ea commodo consequat.
               Duis aute irure dolor in reprehenderit in voluptate
               velit esse cillum dolore eu fugiat nulla pariatur.
               Excepteur sint occaecat cupidatat non proident,
               sunt in culpa qui officia deserunt mollit anim id est laborum."
      )
    ))
  )

  app$run_server()
}
```

htmlNoscript *Noscript component*

Description

Noscript is a wrapper for the <noscript> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/noscript

Usage

```
htmlNoscript(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# This component may be used in the index template to define
# alternate content in browsers which have disabled scripts,
# or in which scripts are not supported.
```

---

htmlObjectEl *ObjectEl component*

Description

ObjectEl is a wrapper for the <object> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object

Usage

```
htmlObjectEl(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, data=NULL,
form=NULL, height=NULL, name=NULL, type=NULL, useMap=NULL,
width=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
data	Character. Specifies the URL of the resource.
form	Character. Indicates the form that is the owner of the element.
height	Character | numeric. Specifies the height of elements listed here. For all other elements, use the CSS height property. Note: In some instances, such as , this is a legacy attribute, in which case the CSS height property should be used instead.
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
type	Character. Defines the type of the element.
useMap	Character.
width	Character | numeric. For the elements listed here, this establishes the element's width. Note: For all other instances, such as , this is a legacy attribute, in which case the CSS width property should be used instead.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.

tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  # Note that htmlObjectEl generates the <object> HTML element;
  # for more information, please visit the link in this
  # component's description.
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlObjectEl(
        width = 100,
        height = 97
        #data = "https://i.postimg.cc/tJd8PSVf/Plotly-logo-01-square.png"
      )
    ))
  )

  app$run_server()
}
```

---

htmlOl *Ol component*

Description

Ol is a wrapper for the <ol> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ol

Usage

```
htmlOl(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, reversed=NULL,
start=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
reversed	A value equal to: 'reversed', 'reversed' | logical. Indicates whether the list should be displayed in a descending order instead of a ascending.
start	Character. Defines the first number if other than 1.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.

tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlOl(list(
        htmlLi("Un"),
        htmlLi("Deux"),
        htmlLi("Trois")
      ))
    ))
  )

  app$run_server()
}
```

---

htmlOptgroup *Optgroup component*

Description

Optgroup is a wrapper for the <optgroup> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/optgroup

Usage

```
htmlOptgroup(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, disabled=NULL,
label=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
disabled	A value equal to: 'disabled', 'disabled' | logical. Indicates whether the user can interact with the element.
label	Character. Specifies a user-readable title of the element.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.

tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlLabel(
        htmlFor = "option-select", "Please select car brand/model: "
      ),
      htmlSelect(id = "option-select", list(
      htmlOptgroup("Audi"), #label = "Audi"
      htmlOption("TT"),
      htmlOption("A4"),
      htmlOptgroup("BMW"), #label = "BMW"
      htmlOption("3 Series"),
      htmlOption("5 Series")
      ))
    ))
  )

  app$run_server()
}
```

---

htmlOption *Option component*

Description

Option is a wrapper for the <option> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/option

Usage

```
htmlOption(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, disabled=NULL,
label=NULL, selected=NULL, value=NULL, accessKey=NULL,
className=NULL, contentEditable=NULL, contextMenu=NULL,
dir=NULL, draggable=NULL, hidden=NULL, lang=NULL,
spellCheck=NULL, style=NULL, tabIndex=NULL, title=NULL,
loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
disabled	A value equal to: 'disabled', 'disabled' | logical. Indicates whether the user can interact with the element.
label	Character. Specifies a user-readable title of the element.
selected	A value equal to: 'selected', 'selected' | logical. Defines a value which will be selected on page load.
value	Character. Defines a default value which will be displayed in the element on page load.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlSelect(list(
        htmlOption("d'Artagnan"),
        htmlOption("Athos"),
        htmlOption("Porthos"),
        htmlOption("Aramis")
      ))
    ))
  )

  app$run_server()
}
```

---

htmlOutput *Output component*

Description

Output is a wrapper for the <output> HTML5 element. CAUTION: <output> is included for completeness, but its typical usage requires the oninput attribute of the enclosing <form> element, which is not accessible to Dash. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/output

Usage

```
htmlOutput(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, htmlFor=NULL,
form=NULL, name=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
htmlFor	Character. Describes elements which belongs to this one.
form	Character. Indicates the form that is the owner of the element.
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.

tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# This component works with htmlForm and htmlInput
# and may be used to present the result of an
# executed script.
```

---

htmlP *P component*

Description

P is a wrapper for the <p> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/p

Usage

```
htmlP(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
```

```
    htmlDiv(list(
      htmlP("The <p> tag defines a paragraph.")
    ))
  )

  app$run_server()
}
```

---

htmlParam *Param component*

Description

Param is a wrapper for the <param> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/param

Usage

```
htmlParam(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, name=NULL,
value=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
value	Character. Defines a default value which will be displayed in the element on page load.

accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP("The <param> element is used to specify the parameters that apply to
      plugin-powered content embedded with an <object> element.
      Read more: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/param"),
      htmlObjectEl(
        #data = "link-to-data-file"
        htmlParam(name = "controller", value = TRUE)
      )
    ))
```

```
  )

  app$run_server()
}
```

`htmlPicture` *Picture component*

Description

Picture is a wrapper for the <picture> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/picture

Usage

```
htmlPicture(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# The URL below has been chunked to comply with CRAN
# requirements; the use of file.path is optional and not required
# for this component.
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlPicture(list(
        htmlSource(srcSet = file.path("https://upload.wikimedia.org",
                                      "wikipedia/commons/a/a7",
                                      "Winter_and_the_City.jpg",
                                      fsep = "/"),
                   media = "(min-width: 800px)"),
        htmlImg(src = file.path("https://upload.wikimedia.org",
                                      "wikipedia/commons/5/56",
                                      "Summer_and_the_City.jpg",
                                      fsep = "/")),
        htmlP("Resize screen to see image changing...")
      ))
    ))
```

```
  )

  app$run_server()
}
```

`htmlPlaintext` *Plaintext component*

Description

Plaintext is a wrapper for the <plaintext> HTML5 element. OBSOLETE: <plaintext> is included for completeness, but should be avoided as browsers may remove it at any time, and its behavior when added dynamically by Dash is not what it would be statically on page load. Use <pre> or <code> instead. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/plaintext

Usage

```
htmlPlaintext(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.

contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  # Warning: The <plaintext> tag is obsolete,
  # it might not work as intended.
  # Use the <pre> tag instead.

  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlPlaintext(),
      htmlBr(),
      htmlH4("The HTML Plaintext Element (<plaintext>) renders everything following
          the start tag as raw text, ignoring any following HTML. There is no closing tag,
            since everything after it is considered raw text.")
    ))
  )

  app$run_server()
}
```

htmlPre *Pre component*

Description

Pre is a wrapper for the <pre> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/pre

Usage

```
htmlPre(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlPre(
        "
        Text in a <pre> element is displayed
        in a fixed-width font (usually Courier),
        and it preserves both spaces and line breaks.
        "
      )
    ))
  )

  app$run_server()
}
```

---

htmlProgress *Progress component*

Description

Progress is a wrapper for the <progress> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/progress

Usage

```
htmlProgress(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, form=NULL,
max=NULL, value=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
form	Character. Indicates the form that is the owner of the element.
max	Character | numeric. Indicates the maximum value allowed.
value	Character. Defines a default value which will be displayed in the element on page load.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.

tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP("Below is an example of htmlProgress"),
      htmlProgress(value = 80, max = 100)
    ))
  )

  app$run_server()
}
```

---

htmlQ *Q component*

Description

Q is a wrapper for the <q> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/q

Usage

```
htmlQ(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, cite=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
cite	Character. Contains a URI which points to the source of the quote or change.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP(list(
      "The <q> tag defines a short quotation: ",
      htmlQ("This example text is wrapped in htmlQ")
      ))
    ))
  )

  app$run_server()
}
```

`htmlRb` *Rb component*

Description

Rb is a wrapper for the <rb> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/rb

Usage

```
htmlRb(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlRuby(list(
        "\U{6f22}",
        htmlRp("("),
```

```
        htmlRt("kan"),
        htmlRp(")")
      )),
      htmlRuby(list(
        "\U{5b57}",
        htmlRp("("),
        htmlRt("ji"),
        htmlRp(")")
      ))
    ))
  )

  app$run_server()
}
```

`htmlRp` *Rp component*

Description

Rp is a wrapper for the <rp> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/rp

Usage

```
htmlRp(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlRuby(list(
        "\U{6f22}",
        htmlRp("("),
        htmlRt("kan"),
        htmlRp(")")
      )),
      htmlRuby(list(
```

```
        "\U{5b57}",
        htmlRp("("),
        htmlRt("ji"),
        htmlRp(")")
      ))
    ))
  )

  app$run_server()
}
```

---

htmlRt *Rt component*

Description

Rt is a wrapper for the <rt> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/rt

Usage

```
htmlRt(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.

contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlRuby(list(
        "\U{6f22}",
        htmlRp("("),
        htmlRt("kan"),
        htmlRp(")")
      )),
      htmlRuby(list(
        "\U{5b57}",
        htmlRp("("),
        htmlRt("ji"),
        htmlRp(")")
```

```
      ))
    ))
  )

  app$run_server()
}
```

`htmlRtc` *Rtc component*

Description

Rtc is a wrapper for the <rtc> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/rtc

Usage

```
htmlRtc(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlRuby(list(
        "\U2661",
        htmlRtc(htmlRt("Heart"))
      ))
    ))
  )

  app$run_server()
}
```

htmlRuby *Ruby component*

Description

Ruby is a wrapper for the <ruby> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ruby

Usage

```
htmlRuby(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlRuby("\U{54d0}")
    ))
  )

  app$run_server()
}
```

---

htmlS *S component*

Description

S is a wrapper for the <s> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/s

Usage

```
htmlS(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlS("htmlS  generates strikethrough text"),
      htmlP(),
      htmlB("htmlB  generates bold text")
    ))
  )

  app$run_server()
}
```

---

htmlSamp *Samp component*

Description

Samp is a wrapper for the <samp> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/samp

Usage

```
htmlSamp(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlSamp("htmlSamp formats text to computer program output.")
    ))
  )

  app$run_server()
}
```

`htmlScript` *Script component*

Description

Script is a wrapper for the <script> HTML5 element. CAUTION: <script> is included for completeness, but you cannot execute JavaScript code by providing it to a <script> element. Use a clientside callback for this purpose instead. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script

Usage

```
htmlScript(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, async=NULL,
charSet=NULL, crossOrigin=NULL, defer=NULL, integrity=NULL,
referrerPolicy=NULL, src=NULL, type=NULL, accessKey=NULL,
className=NULL, contentEditable=NULL, contextMenu=NULL,
dir=NULL, draggable=NULL, hidden=NULL, lang=NULL,
spellCheck=NULL, style=NULL, tabIndex=NULL, title=NULL,
loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
async	A value equal to: 'async', 'async' | logical. Executes the script asynchronously.
charSet	Character. Declares the character encoding of the page or script.
crossOrigin	Character. How the element handles cross-origin requests
defer	A value equal to: 'defer', 'defer' | logical. Indicates that the script should be executed after the page has been parsed.
integrity	Character. Specifies a Subresource Integrity value that allows browsers to verify what they fetch.
referrerPolicy	Character. Specifies which referrer is sent when fetching the resource.
src	Character. The URL of the embeddable content.
type	Character. Defines the type of the element.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# This component is retained for compatibility reasons, but we suggest
# using Dash's capability for embedding scripts within the assets folder
# instead.
```

`htmlSection` *Section component*

Description

Section is a wrapper for the <section> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/section

Usage

```
htmlSection(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlSection(
        children = list(
          htmlH1("This is a section title"),
          htmlDiv("This is some text within a section")
        )
      )
      )
    )
  )

  app$run_server()
}
```

htmlSelect *Select component*

Description

Select is a wrapper for the <select> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/select

Usage

```
htmlSelect(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
autoComplete=NULL, autoFocus=NULL, disabled=NULL, form=NULL,
multiple=NULL, name=NULL, required=NULL, size=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
autoComplete	Character. Indicates whether controls in this form can by default have their values automatically completed by the browser.
autoFocus	A value equal to: 'autofocus', 'autofocus', 'autofocus' | logical. The element should be automatically focused after the page loaded.
disabled	A value equal to: 'disabled', 'disabled' | logical. Indicates whether the user can interact with the element.
form	Character. Indicates the form that is the owner of the element.
multiple	A value equal to: 'multiple', 'multiple' | logical. Indicates whether multiple values can be entered in an input of the type email or file.

name	Character. Name of the element. For example used by the server to identify the fields in form submits.
required	A value equal to: 'required', 'required' | logical. Indicates whether this element is required to fill out or not.
size	Character | numeric. Defines the width of the element (in pixels). If the element's type attribute is text or password then it's the number of characters.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlSelect(
```

```
          children = list(
            htmlOption("This is an option in htmlSelect"),
            htmlOption("But you might want to check out dccDropdown as well"),
            htmlOption("dccDropdown is part of the dashCoreComponents library")
          )
        )
        )
      )
    )

    app$run_server()
}
```

`htmlShadow` *Shadow component*

Description

Shadow is a wrapper for the <shadow> HTML5 element. DEPRECATED: <shadow> is included for completeness, but should be avoided as it is not supported by all browsers and may be removed at any time from those that do support it. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/shadow

Usage

```
htmlShadow(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# The Shadow element requires a browser that supports
# Web Components. It is experimental and should be used
# with caution; it is now considered an obsolete element
# within the Web Components suite. It is retained here
# solely for backwards compatibility reasons.
#
# For more information, please see the MDN link above.
```

htmlSlot *Slot component*

Description

Slot is a wrapper for the <slot> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/slot

Usage

```
htmlSlot(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# Please see https://mdn.github.io/web-components-examples/element-details/
# and https://github.com/mdn/web-components-examples/tree/master/element-details
# for a useful example of this element (with accompanying JavaScript) in action.
```

---

htmlSmall *Small component*

Description

Small is a wrapper for the <small> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/small

Usage

```
htmlSmall(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "This is normal text",
      htmlBr(),
      htmlSmall("And this is text in an htmlSmall component")
      )
    )
  )

  app$run_server()
}
```

`htmlSource` *Source component*

Description

Source is a wrapper for the <source> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/source

Usage

```
htmlSource(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, media=NULL,
sizes=NULL, src=NULL, srcSet=NULL, type=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
media	Character. Specifies a hint of the media for which the linked resource was designed.
sizes	Character.
src	Character. The URL of the embeddable content.
srcSet	Character. One or more responsive image candidates.
type	Character. Defines the type of the element.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# The URL below has been chunked to comply with CRAN
# requirements; the use of file.path is optional and not required
# for this component.
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
     "Resize your browser window to see the image source change based on the browser width",
       htmlBr(),
       htmlPicture(
         list(
           htmlSource(
             media = "(min-width: 1000px)",
            srcSet = "https://apod.nasa.gov/apod/image/1907/FishheadNebula_Pham_2401.jpg"
           ),
           htmlImg(
             src = "https://apod.nasa.gov/apod/image/1907/ngc3576_campbell_1824.jpg"
           )
         )
       )
     )
    )
  )

  app$run_server()
}
```

---

htmlSpacer *Spacer component*

Description

Spacer is a wrapper for the <spacer> HTML5 element. OBSOLETE: <spacer> is included for completeness, but should be avoided as it is not supported by any modern browsers. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/spacer

Usage

```
htmlSpacer(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# This component provides an HTML element that is now obsolete
# and not supported by modern web browsers; it is retained for
# backwards compatibility.
```

`htmlSpan` *Span component*

Description

Span is a wrapper for the HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span

Usage

```
htmlSpan(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "This is some text",
      htmlBr(),
      htmlSpan(
        children = "And some text within an italicized span",
        style = list(fontStyle = "italic")
      )
      )
    )
  )

  app$run_server()
}
```

htmlStrike *Strike component*

Description

Strike is a wrapper for the <strike> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/strike

Usage

```
htmlStrike(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "This is normal text",
      htmlStrike("Text within an htmlStrike element will be stricken out")
      )
    )
  )

  app$run_server()
}
```

---

htmlStrong *Strong component*

Description

Strong is a wrapper for the HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/strong

Usage

```
htmlStrong(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.

loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "This is normal text",
      htmlBr(),
      htmlStrong("Text within an htmlStrong element will be Bold")
      )
    )
  )

  app$run_server()
}
```

---

htmlSub *Sub component*

Description

Sub is a wrapper for the <sub> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sub

Usage

```
htmlSub(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "This is normal text ",
      htmlSub("And this is subscript text within an htmlSub")
      )
    )
  )

  app$run_server()
}
```

`htmlSummary` *Summary component*

Description

Summary is a wrapper for the <summary> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/summary

Usage

```
htmlSummary(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlDetails(
        children = list(
          htmlSummary(
```

```
          children = "Within a details element, the summary can act as a clickable description"
            ),
            "And the rest is hidden until the summary is clicked"
          )
        )
        )
      )
    )

    app$run_server()
}
```

`htmlSup` *Sup component*

Description

Sup is a wrapper for the <sup> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sup

Usage

```
htmlSup(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.

`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
`dir`	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
`draggable`	Character. Defines whether the element can be dragged.
`hidden`	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
`lang`	Character. Defines the language used in the element.
`spellCheck`	Character. Indicates whether spell checking is allowed for the element.
`style`	Named list. Defines CSS styles which will override styles previously set.
`tabIndex`	Character. Overrides the browser's default tab order and follows the one specified instead.
`title`	Character. Text to be displayed in a tooltip when hovering over the element.
`loading_state`	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
`...`	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "This is normal text",
      htmlSup("And this is superscript text within an htmlSup")
      )
    )
  )

  app$run_server()
}
```

`htmlTable` *Table component*

Description

Table is a wrapper for the <table> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/table

Usage

```
htmlTable(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, summary=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`summary`	Character.
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
`dir`	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
`draggable`	Character. Defines whether the element can be dragged.

hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "You can create a table with htmlTable:",
      htmlBr(),
      htmlTable(
        list(
          htmlTr(
            list(
              htmlTh("Table Header 1"),
              htmlTh("Table Header 2")
            )
          ),
          htmlTr(
            list(
              htmlTd("row 1 under Header 1"),
              htmlTd("row 1 under Header 2")
            )
          )
        )
      )
      )
    )
```

```
  )

  app$run_server()
}
```

`htmlTbody` *Tbody component*

Description

Tbody is a wrapper for the <tbody> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/tbody

Usage

```
htmlTbody(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "Within an htmlTable, htmlTbody specifies rows for the table body",
      htmlBr(),
      htmlTable(
        list(
          htmlThead(
            htmlTr(
              htmlTh("This is in the header of the table")
            )
          ),
          htmlTbody(
            htmlTr(
              htmlTd("This is in the body of the table")
            )
          ),
          htmlTfoot(
            htmlTr(
```

```
              htmlTd("This is in the footer of the table")
            )
          )
        )
      )
      )
    )
  )

  app$run_server()
}
```

`htmlTd` *Td component*

Description

Td is a wrapper for the <td> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/td

Usage

```
htmlTd(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, colSpan=NULL,
headers=NULL, rowSpan=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute

colSpan	Character	numeric. The colspan attribute defines the number of columns a cell should span.
headers	Character. IDs of the <th> elements which applies to this element.
rowSpan	Character	numeric. Defines the number of rows a table cell should span over.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden'	logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "Within an htmlTable, individual cells can be made with htmlTd",
      htmlBr(),
      htmlTable(
```

```
          list(
            htmlTr(
              list(
                htmlTh("Header 1"),
                htmlTh("Header 2")
              )
            ),
            htmlTr(
              list(
                htmlTd("this is a cell"),
                htmlTd("this is another cell")
              )
            )
          )
        )
        )
        )
      )
    )

    app$run_server()
  }
```

`htmlTemplate` *Template component*

Description

Template is a wrapper for the <template> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/template

Usage

```
htmlTemplate(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.

n_clicks_timestamp
: Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key
: Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info

role
: Character. The ARIA role attribute

accessKey
: Character. Keyboard shortcut to activate or add focus to the element.

className
: Character. Often used with CSS to style elements with common properties.

contentEditable
: Character. Indicates whether the element's content is editable.

contextMenu
: Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir
: Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)

draggable
: Character. Defines whether the element can be dragged.

hidden
: A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang
: Character. Defines the language used in the element.

spellCheck
: Character. Indicates whether spell checking is allowed for the element.

style
: Named list. Defines CSS styles which will override styles previously set.

tabIndex
: Character. Overrides the browser's default tab order and follows the one specified instead.

title
: Character. Text to be displayed in a tooltip when hovering over the element.

loading_state
: Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer

...
: wildcards allowed have the form: ''data-*', 'aria-*'`

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
```

```
    htmlDiv(list(
      "You can create an HTML template to be populated later via js",
      htmlBr(),
      htmlTable(
        id = "myTable",
        htmlTr(
          list(
            htmlTh("Header 1"),
            htmlTh("Header 2")
          )
        )
      ),
      htmlTemplate(
        id = "myRowTemplate",
        htmlTr(
          list(
            htmlTd(className = "someRowValue"),
            htmlTd()
          )
        )
      )
      )
    )
  )

  app$run_server()
}
```

`htmlTextarea` *Textarea component*

Description

Textarea is a wrapper for the <textarea> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea

Usage

```
htmlTextarea(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
autoComplete=NULL, autoFocus=NULL, cols=NULL, disabled=NULL,
form=NULL, inputMode=NULL, maxLength=NULL, minLength=NULL,
name=NULL, placeholder=NULL, readOnly=NULL, required=NULL,
rows=NULL, wrap=NULL, accessKey=NULL, className=NULL,
contentEditable=NULL, contextMenu=NULL, dir=NULL,
draggable=NULL, hidden=NULL, lang=NULL, spellCheck=NULL,
style=NULL, tabIndex=NULL, title=NULL, loading_state=NULL,
...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
autoComplete	Character. Indicates whether controls in this form can by default have their values automatically completed by the browser.
autoFocus	A value equal to: 'autofocus', 'autofocus', 'autofocus' | logical. The element should be automatically focused after the page loaded.
cols	Character | numeric. Defines the number of columns in a textarea.
disabled	A value equal to: 'disabled', 'disabled' | logical. Indicates whether the user can interact with the element.
form	Character. Indicates the form that is the owner of the element.
inputMode	Character. Provides a hint as to the type of data that might be entered by the user while editing the element or its contents. The attribute can be used with form controls (such as the value of textarea elements), or in elements in an editing host (e.g., using contenteditable attribute).
maxLength	Character | numeric. Defines the maximum number of characters allowed in the element.
minLength	Character | numeric. Defines the minimum number of characters allowed in the element.
name	Character. Name of the element. For example used by the server to identify the fields in form submits.
placeholder	Character. Provides a hint to the user of what can be entered in the field.
readOnly	Character. Indicates whether the element can be edited.
required	A value equal to: 'required', 'required' | logical. Indicates whether this element is required to fill out or not.
rows	Character | numeric. Defines the number of rows in a text area.
wrap	Character. Indicates whether the text should be wrapped.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.

contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlTextarea(
        rows = 4, cols = 50,
        children = "A text area allows users to input text"
      )
      )
    )
  )

  app$run_server()
}
```

`htmlTfoot` *Tfoot component*

Description

Tfoot is a wrapper for the <tfoot> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/tfoot

Usage

```
htmlTfoot(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
`dir`	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
`draggable`	Character. Defines whether the element can be dragged.
`hidden`	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "Within an htmlTable, you can create footer rows with htmlTfoot",
      htmlBr(),
      htmlTable(
        list(
          htmlThead(
            htmlTr(
              htmlTh("This is in the header of the table")
            )
          ),
          htmlTbody(
            htmlTr(
              htmlTd("This is in the body of the table")
            )
          ),
          htmlTfoot(
            htmlTr(
              htmlTd("This is in the footer of the table")
            )
          )
        )
      )
    ))
  )
```

```
  )

  app$run_server()
}
```

---

htmlTh *Th component*

Description

Th is a wrapper for the <th> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/th

Usage

```
htmlTh(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, colSpan=NULL,
headers=NULL, rowSpan=NULL, scope=NULL, accessKey=NULL,
className=NULL, contentEditable=NULL, contextMenu=NULL,
dir=NULL, draggable=NULL, hidden=NULL, lang=NULL,
spellCheck=NULL, style=NULL, tabIndex=NULL, title=NULL,
loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
colSpan	Character | numeric. The colspan attribute defines the number of columns a cell should span.
headers	Character. IDs of the <th> elements which applies to this element.
rowSpan	Character | numeric. Defines the number of rows a table cell should span over.
scope	Character. Defines the cells that the header test (defined in the th element) relates to.

accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlTable(
        list(
          # the following row contains headers
          htmlTr(
            list(
              htmlTh("Header 1"),
              htmlTh("Header 2")
            )
```

```
          )
        )
      )
      )
    )
  )

  app$run_server()
}
```

`htmlThead` *Thead component*

Description

Thead is a wrapper for the <thead> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/thead

Usage

```
htmlThead(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.

contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "Within an htmlTable, you can create a header with htmlThead",
      htmlBr(),
      htmlTable(
        list(
          htmlThead(
            htmlTr(
              htmlTh("This is in the header of the table")
            )
          ),
          htmlTbody(
            htmlTr(
```

```
              htmlTd("This is in the body of the table")
            )
          ),
          htmlTfoot(
            htmlTr(
              htmlTd("This is in the footer of the table")
            )
          )
        )
      )
      )
    )
  )

  app$run_server()
}
```

---

htmlTime *Time component*

Description

Time is a wrapper for the <time> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/time

Usage

```
htmlTime(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, dateTime=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
dateTime	Character. Indicates the date and time associated with the element.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlP(
        list(
```

```
          "It might be useful to wrap dates like ",
          htmlTime(dateTime = "2019-07-29", children = "July 29th"),
          " in an htmlTime to make your datetime strings machine-readable."
        )
      )
      )
    )
  )

  app$run_server()
}
```

`htmlTitle` *Title component*

Description

Title is a wrapper for the <title> HTML5 element. CAUTION: <title> is included for completeness, but is not expected to do anything outside of <head>. Dash components are always created in the <body>. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/title

Usage

```
htmlTitle(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute

accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# This component's effects will be overridden by the index
# template in Dash for R. We suggest using Dash's API to
# set the page title instead:
#
# app$title('My page title')
```

htmlTr *Tr component*

Description

Tr is a wrapper for the <tr> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/tr

Usage

```
htmlTr(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "Within an htmlTable, individual rows can be made with htmlTr",
      htmlBr(),
      htmlTable(
        list(
          # the following row contains headers
          htmlTr(
            list(
              htmlTh("Header 1"),
              htmlTh("Header 2")
            )
          ),
          # the following row contains cells
          htmlTr(
            list(
              htmlTd("this is a cell"),
              htmlTd("this is another cell")
            )
          )
        )
      )
      )
    )
  )
```

```
  app$run_server()
}
```

`htmlTrack` *Track component*

Description

Track is a wrapper for the <track> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/track

Usage

```
htmlTrack(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, default=NULL,
kind=NULL, label=NULL, src=NULL, srcLang=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`default`	A value equal to: 'default', 'default' | logical. Indicates that the track should be enabled unless the user's preferences indicate something different.
`kind`	Character. Specifies the kind of text track.
`label`	Character. Specifies a user-readable title of the element.
`src`	Character. The URL of the embeddable content.
`srcLang`	Character.

accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
# The URL below has been chunked to comply with CRAN
# requirements; the use of file.path is optional and not required
# for this component.
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlVideo(children = list(
        htmlSource(src = file.path("https://interactive-examples.mdn.mozilla.net",
                                   "media/examples",
                                   "friday.mp4",
                                   fsep = "/"),
```

```
                type = 'video/mp4'),
        htmlTrack(kind = 'captions',
                srcLang = 'en',
                src = file.path("https://interactive-examples.mdn.mozilla.net",
                                "media/examples",
                                "friday.vtt",
                                fsep = "/"),
                default = 'default',
                label = 'English')
      ),
      controls = TRUE
    )
    )
  )
  )

  app$run_server()
}
```

---

htmlU *U component*

Description

U is a wrapper for the <u> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/u

Usage

```
htmlU(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.

key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlU("Wrap your text in htmlU to have it underlined")
      )
    )
```

```
  )

  app$run_server()
}
```

`htmlUl` *Ul component*

Description

Ul is a wrapper for the <ul> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ul

Usage

```
htmlUl(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

`children`	A list of or a singular dash component, string or number. The children of this component
`id`	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
`n_clicks`	Numeric. An integer that represents the number of times that this element has been clicked on.
`n_clicks_timestamp`	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
`key`	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
`role`	Character. The ARIA role attribute
`accessKey`	Character. Keyboard shortcut to activate or add focus to the element.
`className`	Character. Often used with CSS to style elements with common properties.
`contentEditable`	Character. Indicates whether the element's content is editable.
`contextMenu`	Character. Defines the ID of a <menu> element which will serve as the element's context menu.

dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "You can make an unordered list with htmlUl",
      htmlBr(),
      htmlUl(
        children = list(
          htmlLi("Some item"),
          htmlLi("Some other item")
        )
      )
      )
    )
  )

  app$run_server()
}
```

htmlVar *Var component*

Description

Var is a wrapper for the <var> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/var

Usage

```
htmlVar(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      "You can use htmlVar to represent the name of a variable",
      htmlBr(),
      htmlVar("myVariable")
      )
    )
  )

  app$run_server()
}
```

---

htmlVideo *Video component*

Description

Video is a wrapper for the <video> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/video

Usage

```
htmlVideo(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL, autoPlay=NULL,
controls=NULL, crossOrigin=NULL, height=NULL, loop=NULL,
muted=NULL, poster=NULL, preload=NULL, src=NULL, width=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
autoPlay	A value equal to: 'autoplay', 'autoplay', 'autoplay' | logical. The audio or video should play as soon as possible.
controls	A value equal to: 'controls', 'controls' | logical. Indicates whether the browser should show playback controls to the user.
crossOrigin	Character. How the element handles cross-origin requests
height	Character | numeric. Specifies the height of elements listed here. For all other elements, use the CSS height property. Note: In some instances, such as , this is a legacy attribute, in which case the CSS height property should be used instead.
loop	A value equal to: 'loop', 'loop' | logical. Indicates whether the media should start playing from the start when it's finished.
muted	A value equal to: 'muted', 'muted' | logical. Indicates whether the audio will be initially silenced on page load.
poster	Character. A URL indicating a poster frame to show until the user plays or seeks.
preload	Character. Indicates whether the whole resource, parts of it or nothing should be preloaded.
src	Character. The URL of the embeddable content.

width	Character	numeric. For the elements listed here, this establishes the element's width. Note: For all other instances, such as , this is a legacy attribute, in which case the CSS width property should be used instead.
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden'	logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlVideo(
        src = file.path('https://ia800303.us.archive.org',
              '18/items/bacteria_friend_and_foe',
              'bacteria_friend_and_foe_512kb.mp4',
              fsep = '/'),
```

```
        controls = TRUE,
        title = "Bacteria: Friend and Foe"
      )
      )
    )
  )

  app$run_server()
}
```

---

htmlWbr *Wbr component*

Description

Wbr is a wrapper for the <wbr> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/wbr

Usage

```
htmlWbr(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.

contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
     "In a long string, it might be a good idea to add an htmlWbr to specify word breaks",
      htmlP("Thisverylongstringwithnowhitespaceswon'tlookverygood"),
      htmlWbr(),
      htmlP("butatleastyoucanspecifya'natural'placeforthestringtobebrokenup")
      )
    )
  )

  app$run_server()
}
```

htmlXmp *Xmp component*

Description

Xmp is a wrapper for the <xmp> HTML5 element. For detailed attribute info see: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/xmp

Usage

```
htmlXmp(children=NULL, id=NULL, n_clicks=NULL,
n_clicks_timestamp=NULL, key=NULL, role=NULL,
accessKey=NULL, className=NULL, contentEditable=NULL,
contextMenu=NULL, dir=NULL, draggable=NULL, hidden=NULL,
lang=NULL, spellCheck=NULL, style=NULL, tabIndex=NULL,
title=NULL, loading_state=NULL, ...)
```

Arguments

children	A list of or a singular dash component, string or number. The children of this component
id	Character. The ID of this component, used to identify dash components in callbacks. The ID needs to be unique across all of the components in an app.
n_clicks	Numeric. An integer that represents the number of times that this element has been clicked on.
n_clicks_timestamp	Numeric. An integer that represents the time (in ms since 1970) at which n_clicks changed. This can be used to tell which button was changed most recently.
key	Character. A unique identifier for the component, used to improve performance by React.js while rendering components See https://reactjs.org/docs/lists-and-keys.html for more info
role	Character. The ARIA role attribute
accessKey	Character. Keyboard shortcut to activate or add focus to the element.
className	Character. Often used with CSS to style elements with common properties.
contentEditable	Character. Indicates whether the element's content is editable.
contextMenu	Character. Defines the ID of a <menu> element which will serve as the element's context menu.
dir	Character. Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
draggable	Character. Defines whether the element can be dragged.
hidden	A value equal to: 'hidden', 'hidden' | logical. Prevents rendering of given element, while keeping child elements, e.g. script elements, active.

lang	Character. Defines the language used in the element.
spellCheck	Character. Indicates whether spell checking is allowed for the element.
style	Named list. Defines CSS styles which will override styles previously set.
tabIndex	Character. Overrides the browser's default tab order and follows the one specified instead.
title	Character. Text to be displayed in a tooltip when hovering over the element.
loading_state	Lists containing elements 'is_loading', 'prop_name', 'component_name'. those elements have the following types: - is_loading (logical; optional): determines if the component is loading or not - prop_name (character; optional): holds which property is loading - component_name (character; optional): holds the name of the component that is loading. Object that holds the loading state object coming from dash-renderer
...	wildcards allowed have the form: ''data-*', 'aria-*''

Value

named list of JSON elements corresponding to React.js properties and their values

Examples

```
if (interactive()) {
  library(dash)

  app <- Dash$new()

  app$layout(
    htmlDiv(list(
      htmlXmp("xmp elements will be rendered in monospace font"),
      htmlXmp("Note that this element is obsolete in HTML5"),
      htmlA(
        "See this for more details",
        href = "https://developer.mozilla.org/en-US/docs/Web/HTML/Element/xmp"
      )
      )
    )
  )

  app$run_server()
}
```

---

install_snippet *Install Dash RStudio snippet*

Description

Install the Dash code snippet for RStudio, for quickly creating a new Dash app.

Usage

```
install_snippet()
```

Value

boolean Whether or not the snippet was installed

---

is_dash_app *Is the given object a Dash app?*

Description

Is the given object a Dash app?

Usage

```
is_dash_app(x)
```

Arguments

x	Any object.

---

prevent_update *Prevent a callback from updating its output*

Description

When used inside Dash callbacks, if any of the arguments evaluate to TRUE, then the callback's outputs do not update.

Usage

```
prevent_update(...)
```

Arguments

...	Values to check

Examples

```
if (interactive()) {
    app <- dash_app()

    app %>% set_layout(
      button('Click here', id = 'btn'),
      p('The number of times the button was clicked does not
       update when the number is divisible by 5'),
      div(id = 'body-div')
    )
    app %>% add_callback(
      output(id='body-div', property='children'),
      list(
        input(id='btn', property='n_clicks')
      ),
      function(n_clicks) {
        prevent_update(is.null(n_clicks[[1]]), n_clicks[[1]] %% 5 == 0)
        paste(n_clicks[[1]], "clicks")
      }
    )

    app %>% run_app()
}
```

---

print.dash_component *Output a dash component object as JSON*

Description

Objects of the dash_component class support a print method, which first processes the nested list object, and then returns its JSON representation.

Usage

```
## S3 method for class 'dash_component'
print(x, ...)
```

Arguments

x	an object of class dash_component
...	not currently used

run_app *Run a Dash app*

Description

Run a Dash app

Usage

```
run_app(
  app,
  host = Sys.getenv("DASH_HOST", Sys.getenv("HOST", "127.0.0.1")),
  port = Sys.getenv("DASH_PORT", Sys.getenv("PORT", 8050)),
  browser = interactive()
)
```

Arguments

app	A dash application created with dash_app()
host	Hostname to run the app.
port	Port number to run the app.
browser	Whether or not to launch a browser to the app's URL.

selectors *Pattern-Matching Callback Selectors*

Description

Symbols which reference pattern-matching callback selectors with the same names

Usage

```
ALL

ALLSMALLER

MATCH
```

Format

An object of class name of length 1.

An object of class name of length 1.

An object of class name of length 1.

Details

ALL, ALLSMALLER and MATCH are symbols corresponding to the pattern-matching callback selectors with the same names. These allow you to write callbacks that respond to or update an arbitrary or dynamic number of components. Because they are symbols (see name) rather than functions, each has no arguments. For more information, see the callback section in Dash.

For pattern-matching callbacks, the id field of a component is written in JSON-like syntax. The resulting id is then transformed into a dictionary object when serialized for use by the Dash renderer within the web browser. The fields are arbitrary keys, which describe the targets of the callback.

For example, when we write input(id=list("foo" = ALL, "bar" = "dropdown"), Dash interprets this as "match any input that has an ID list where 'foo' is 'ALL' and 'bar' is anything." If any of the dropdown value properties change, all of their values are returned to the callback.

However, for readability, we recommend using keys like type, index, or id. type can be used to refer to the class or set of dynamic components and index or id could be used to refer to the component you are matching within that set. While your application may have a single set of dynamic components, it's possible to specify multiple sets of dynamic components in more complex apps or if you are using MATCH.

Like ALL, MATCH will fire the callback when any of the component's properties change. However, instead of passing all of the values into the callback, MATCH will pass just a single value into the callback. Instead of updating a single output, it will update the dynamic output that is "matched" with.

ALLSMALLER is used to pass in the values of all of the targeted components on the page that have an index smaller than the index corresponding to the div. For example, ALLSMALLER makes it possible to filter results that are increasingly specific as the user applies each additional selection.

ALLSMALLER can only be used in input and state items, and must be used on a key that has MATCH in the output item(s). ALLSMALLER it isn't always necessary (you can usually use ALL and filter out the indices in your callback), but it will make your logic simpler.

Examples

```
if (interactive() ) {
  library(dash)

  # Simple example illustrating use of ALL selector
  app <- Dash$new()

  app$layout(htmlDiv(list(
    htmlButton("Add Filter", id="add-filter", n_clicks=0),
    htmlDiv(id="dropdown-container", children=list()),
    htmlDiv(id="dropdown-container-output")
  )))


  app$callback(
    output(id="dropdown-container", property = "children"),
    params = list(
      input(id = "add-filter", property = "n_clicks"),
      state(id = "dropdown-container", property = "children")
    ),
```

```
  display_dropdowns <- function(n_clicks, children){
    new_dropdown = dccDropdown(
      id=list(
        "index" = n_clicks,
        "type" = "filter-dropdown"
      ),
      options = lapply(c("NYC", "MTL", "LA", "TOKYO"), function(x){
        list("label" = x, "value" = x)
      })
    )
    children[[n_clicks + 1]] <- new_dropdown
    return(children)
  }
)

app$callback(
  output(id="dropdown-container-output", property="children"),
  params = list(
    input(id=list("index" = ALL, "type" = "filter-dropdown"), property= "value")
  ),
  display_output <- function(test){
    ctx <- app$callback_context()
    return(htmlDiv(
      lapply(seq_along(test), function(x){
        return(htmlDiv(sprintf("Dropdown %s = %s", x, test[[x]])))
      })
    ))
  }
)

app$run_server()

# Simple example illustrating use of ALLSMALLER selector
library(dash)

df <- read.csv('https://raw.githubusercontent.com/plotly/datasets/master/gapminder2007.csv',
               stringsAsFactors = FALSE)

app <- Dash$new()

app$layout(htmlDiv(list(
  htmlButton("Add Filter", id = "add-filter-ex3", n_clicks = 0),
  htmlDiv(id = "container-ex3", children = list())
)))

app$callback(
  output('container-ex3', 'children'),
  params = list(
    input('add-filter-ex3', 'n_clicks'),
    state('container-ex3', 'children')
  ),
  display_dropdowns <- function(n_clicks, existing_children){
    new_children <- htmlDiv(list(
```

```
      dccDropdown(
        id = list("index" = n_clicks, "type" = "filter-dropdown-ex3"),
        options = lapply(unique(df$country), function(x){
          list("label" = x, "value" = x)
        }),
        value = unique(df$country)[n_clicks + 1]
      ),
      htmlDiv(id = list("index" = n_clicks, "type" = "output-ex3"),
              children = list(unique(df$country)[n_clicks + 1]))
    ))

    existing_children <- c(existing_children, list(new_children))
  }
)

app$callback(
  output(id = list("type" = "output-ex3", "index" = MATCH), property = "children"),
  params = list(
   input(id = list("type" = "filter-dropdown-ex3", "index" = MATCH), property = "value"),
  input(id = list("type" = "filter-dropdown-ex3", "index" = ALLSMALLER), property = "value")
  ),
  display_output <- function(matching_value, previous_values){
    previous_values_in_reversed_order = rev(previous_values)
    all_values = c(matching_value, previous_values_in_reversed_order)
    all_values = unlist(all_values)

    dff = df[df$country %in% all_values,]
    avgLifeExp = round(mean(dff$lifeExp), digits = 2)

    if (length(all_values) == 1) {
      return(
        htmlDiv(sprintf("%s is the life expectancy of %s.",
                avgLifeExp,
                matching_value))
      )
    } else if (length(all_values) == 2) {
      return(
        htmlDiv(sprintf("%s is the life expectancy of %s.",
                        avgLifeExp,
                        paste(all_values, collapse = " and ")))
      )
    } else {
      return(
        htmlDiv(sprintf("%s is the life expectancy of %s, and %s.",
                avgLifeExp,
                paste(all_values[-length(all_values)], collapse = " , "),
                paste(all_values[length(all_values)])))
      )
    }
  }
)

app$run_server()
```

```
# Simple example illustrating use of MATCH selector
library(dash)

app <- Dash$new()

app$layout(htmlDiv(list(
  htmlButton("Add Filter", id="dynamic-add-filter", n_clicks=0),
  htmlDiv(id="dynamic-dropdown-container", children = list())
)))

app$callback(
  output(id="dynamic-dropdown-container", "children"),
  params = list(
    input("dynamic-add-filter", "n_clicks"),
    state("dynamic-dropdown-container", "children")
  ),
  display_dropdown <- function(n_clicks, children){
    new_element = htmlDiv(list(
      dccDropdown(
        id = list("index" = n_clicks, "type" = "dynamic-dropdown"),
        options = lapply(c("NYC", "MTL", "LA", "TOKYO"), function(x){
          list("label" = x, "value" = x)
        })
      ),
      htmlDiv(
        id = list("index" = n_clicks, "type" = "dynamic-output"),
        children = list()
      )
    ))

    children <- c(children, list(new_element))
    return(children)
  }
)

app$callback(
  output(id = list("index" = MATCH, "type" = "dynamic-output"), property= "children"),
  params = list(
    input(id=list("index" = MATCH, "type" = "dynamic-dropdown"), property= "value"),
    state(id=list("index" = MATCH, "type" = "dynamic-dropdown"), property= "id")
  ),
  display_output <- function(value, id){
    return(htmlDiv(sprintf("Dropdown %s = %s", id$index, value)))
  }
)

app$run_server()
}
```

---

set_layout *Set the layout of a Dash app*

Description

Set the layout of a Dash app

Usage

```
set_layout(app, ...)
```

Arguments

app	A dash application created with dash_app()
...	Dash components to create the user interface, provided either as comma-separated components or a list of components. You can also provide a function returning a Dash component if you want the layout to re-render on every page load.

Examples

```
app <- dash_app()

app %>% set_layout("hello", "Dash")
app %>% set_layout(div("hello"), "Dash")
app %>% set_layout(list(div("hello"), "Dash"))
app %>% set_layout("Conditional UI using an if statement: ",
                   if (TRUE) "rendered",
                   if (FALSE) "not rendered")
app %>% set_layout(function() { div("Current time: ", Sys.time()) })
```

---

simple_table *Simple HTML table*

Description

Simple HTML table

Usage

```
simple_table(data, colnames = TRUE, rownames = FALSE)
```

Arguments

data	A data.frame
colnames	*(logical)* Whether or not to show the column names (a header row)
rownames	*(logical)* Whether or not to show the row names

Examples

```
if (interactive()) {
    app <- dash_app() %>%
      set_layout(
        dccChecklist(
          id = "table_params",
          labelStyle = list(display = "block"),
          options = list(
            list(label = "Header", value = "colnames"),
            list(label = "Row names", value = "rownames")
          )
        ),
        br(),
        div(id = "table")
      )

    app %>% add_callback(
      output(id = 'table', property = 'children'),
      input(id = 'table_params', property = 'value'),
      function(val) {
      simple_table(mtcars, colnames = "colnames" %in% val, rownames = "rownames" %in% val)
      }
    )

    app %>% run_app()
}
```

`tags` *Create HTML tags*

Description

Create an HTML tag to place in a Dash app layout. All tags are available in the `html` list, and some common tags have shortcuts as functions for convenience (e.g. `h1()` produces <h1> and is equivalent to `html$h1()`).

Usage

```
html

h1(...)

h2(...)

h3(...)

h4(...)
```

```
div(...)

span(...)

p(...)

strong(...)

br(...)

button(...)

a(...)

img(...)

dash_tag(tag_name, content = list())
```

Arguments

...	Any named arguments become tag attributes, and any unnamed arguments become children. A named argument with a value of NULL will be removed, and a named argument with a value of NA will be rendered as a boolean argument. See 'Special attributes' below for more information.
tag_name	The name of the HTML tag.
content	List of attributes and children.

Special attributes

There are a few HTML attributes that are treated in a special way:

- To add a `class` attribute, use the `className` parameter
- To add a `for` attribute, use the `htmlFor` parameter
- The `style` attribute is not provided as a string. Instead, it's provided as a named list, where the name and value of each element correspond to the CSS property and value. Each CSS property should be written in camelCase.
- A special property `n_clicks` is automatically added to every HTML tag. This property represents the number of times that this element has been clicked on. If not explicitly initialized to a certain integer, its default value is `NULL` initially.

Examples

```
if (interactive()) {
    app <- dash_app()
    app %>% set_layout(
      html$div(
        h1(
          "title",
          style = list(
```

```
            "color" = "red",
            "backgroundColor" = "blue"
          )
        ),
        "some text",
        button(
          "can't click me",
          disabled = NA,
          className = "mybtn"
        )
      )
    )
    app %>% run_app()
}
```

Index