Package: cascadess (via r-universe)

October 25, 2024

Type Package

Title A Style Pronoun for 'htmltools' Tags

Version 0.2.0

Description Apply styles to tag elements directly and with the .style pronoun. Using the pronoun, styles are created within the context of a tag element. Change borders, backgrounds, text, margins, layouts, and more.

License MIT + file LICENSE

URL https://nteetor.github.io/cascadess/, https://github.com/nteetor/cascadess

BugReports https://github.com/nteetor/cascadess/issues

Encoding UTF-8

RoxygenNote 7.2.3

Config/testthat/edition 3

Depends R (>= 3.5)

Imports htmltools (>= 0.4.0), magrittr (>= 1.5), rlang, utils

Suggests cli, rmarkdown, shiny, testthat (>= 2.1.0)

Collate 'background-color.R' 'theme.R' 'border-all.R' 'border-color.R' 'border-width.R' 'breakpoints.R' 'cascadess.R' 'display.R' 'flex-align.R' 'flex-content.R' 'flex-direction.R' 'flex-display.R' 'flex-justify.R' 'flex-wrap.R' 'float.R' 'focus-color.R' 'gap-all.R' 'height-relative.R' 'item-align.R' 'item-fill.R' 'item-grow.R' 'item-order.R' 'margin-all.R' 'overflow-all.R' 'padding-all.R' 'position-centered.R' 'position-offset.R' 'position-sticky.R' 'position.R' 'reexports.R' 'rounded-all.R' 'shadow.R' 'stack-vertical.R' 'style-pronoun.R' 'text-alignment.R' 'text-break.R' 'text-color.R' 'text-decoration.R' 'text-height.R' 'text-selection.R' 'text-size.R' 'text-style.R' 'text-transform.R' 'text-weight.R' 'text-wrap.R' 'utils-docs.R' 'utils.R' 'vertical-alignment.R' 'visible.R' 'width-relative.R' 'zzz.R'

NeedsCompilation no

Author Nathan Teetor [aut, cre, cph], The Bootstrap Authors [cph] (Bootstrap library), Twitter, Inc [cph] (Bootstrap library)

Maintainer Nathan Teetor <nate@haufin.ch>

Repository CRAN

Date/Publication 2024-10-24 14:10:07 UTC

Contents

`background_color` *Change background color*

Description

The `background_color()` and `background_subtle()` functions adjust the background color of a tag element.

Usage

```
background_color(x, color)

background_subtle(x, color)
```

Arguments

`x`	A tag element or .style pronoun.
`color`	A character string specifying a color. One of,

- `"primary"`
- `"secondary"`
- `"success"`
- `"danger"`
- `"warning"`
- `"info"`
- `"light"`
- `"dark"`

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

div(
  .style %>%
    background_color(theme_primary()),
  "Primary background"
)

div(
  .style %>%
    background_color(theme_danger()),
  "Danger background"
)

div(
  .style %>%
    background_subtle(theme_warning()) %>%
    border_subtle(theme_warning()) %>%
    text_emphasis(theme_warning()),
  "Warning!"
)


div(
  .style %>%
    background_subtle(theme_dark()) %>%
    border_subtle(theme_dark())
)
```

`border_all` *Add or remove borders*

Description

The `border_all()` and `border_<side>()` functions adjust a tag element's borders.

Usage

```
border_all(x, include = TRUE)

border_top(x, include = TRUE)

border_right(x, include = TRUE)

border_bottom(x, include = TRUE)

border_left(x, include = TRUE)
```

Arguments

x	A tag element or .style pronoun.
include	A boolean specifying to include a side. One of, • TRUE • FALSE Defaults to TRUE.

Value

An object of the same type as x.

Examples

```
library(htmltools)

h3(
  .style %>%
    border_bottom() %>%
    border_color(theme_warning()) %>%
    text_color(theme_warning()),
  "Warning"
)
```

---

border_color *Change border color*

Description

The border_color() and border_subtle() functions adjust the border color of a tag element.

Usage

```
border_color(x, color)

border_subtle(x, color)
```

Arguments

x	A tag element or .style pronoun.
color	A character string specifying a color. One of, • "primary" • "secondary" • "success" • "danger" • "warning"

- "info"
- "light"
- "dark"

Value

An object of the same type as x.

See Also

Other border utilities: border_width()

Examples

```
library(htmltools)

div(
  .style %>%
    border_color(theme_primary())
)

div(
  .style %>%
    background_subtle(theme_danger()) %>%
    border_subtle(theme_danger()) %>%
    text_emphasis(theme_danger()),
  "Danger theme with some emphasis"
)


div(
  .style %>%
    background_subtle(theme_light()) %>%
    text_emphasis(theme_light()) %>%
    border_subtle(theme_light())
)
```

---

border_width *Increase or decrease border width*

Description

Adjust the boder width of a tag element.

Usage

```
border_width(x, width)
```

Arguments

x	A tag element or .style pronoun.
width	A number between 1 and 5.

Value

An object of the same type as x.

See Also

Other border utilities: border_color()

Examples

```
library(htmltools)

div(
  .style %>%
    border_all() %>%
    border_width(3) %>%
    border_color(theme_primary())
)
```

---

breakpoints *Different values for different browser sizes*

Description

Many functions in cascadess accept multiple name-value pairs via .... Each argument's name specifies a breakpoint, a browser width. At this breakpoint, at this browser width, the argument value will supercede any values specified for smaller breakpoints, smaller browser widths.

Breakpoints are browser or device widths and enable developing responsive web applications. A responsive web application will adjust its layout and style to accomodate the size of a user's browser or device. A web application with a responsive design allows users to constructively interact from a browser, phone, tablet, or other device.

Available breakpoints:

extra small (unnamed or xs)

Specify an unnamed value for this breakpoint or use the name xs. This is the only breakpoint which may be unnamed. However, when specifying multiple breakpoints the best practice is to use the xs name.

The value and style are always applied unless superceded by a larger breakpoint.

small (sm)

Specify a value for this breakpoint using the name sm.

The value and style are applied when the viewport is at least 576px wide, think landscape phone.

medium (md)

Specify a value for this breakpoint using the name `md`.

The value and style are applied when the viewport is at least 768px wide, think tablet.

large (lg)

Specify a value for this breakpoint using the name `lg`.

The value and style are applied when the viewport is at least 992px wide, think laptop or smaller desktops.

extra large (xl)

Specify a value for this breakpoint using the name `xl`.

The value and style are applied when the viewport is at least 1200px wide, think large desktops.

extra extra large (xxl)

Specify a value for this breakpoint using the name `xxl`.

The value and style are applied when the viewport is at least 1400px wide, think larger desktops.

Further reading:

These breakpoints are chosen by and are a part of the Bootstrap library cascadess is built upon. The Bootstrap website goes into greater detail on their design, construction, and usage, see https://getbootstrap.com/docs/5.3/layout/breakpoints/.

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display() %>%
    flex_direction(xs = "column", lg = "row")
)

div(
  .style %>%
    flex_display() %>%
    gap_all(xs = 0, md = 2, xl = 4)
)
```

---

cascadess_dependencies

Cascadess' CSS dependencies

Description

For CSS styles to be applied, you must include a call to `cascadess_dependencies()` in your UI or use the bslib package.

Usage

```
cascadess_dependencies()
```

Value

An htmltools::htmlDependency().

Examples

```
## Not run:
library(shiny)

shinyApp(
  ui = list(
    cascadess_dependencies(),
    div(
      .style %>%
        padding_all(3) %>%
        background_color(theme_light()),
      "Etiam laoreet quam sed arcu."
    )
  ),
  server = function(input, output) {}
)

## End(Not run)

## Not run:
library(shiny)
library(bslib)

shinyApp(
  ui = page(
    .style %>%
      background_color(theme_primary()),
    card(
      .style %>%
        margin_all(3) %>%
        background_color(theme_light()),
      "Hello, world!"
    )
  ),
  server = function(input, output) {}
)

## End(Not run)
```

---

display *Display*

Description

The display() function adjusts how a tag element and its contents are rendered.

Usage

```
display(x, ...)
```

Arguments

x	A tag element or .style pronoun.
...	A character string specifying a display type. One of, • "none" • "inline" • "inline-block" • "block" • "grid" • "inline-grid" • "table" • "table-cell" • "table-row" • "flex" • "inline-flex" Use name-value pairs to specify breakpoints.

Value

An object of the same type as x.

Examples

```
library(htmltools)

div(
  .style %>%
    display("flex") %>%
    flex_justify("center"),
  "Powerful stuff"
)

div(
  .style %>%
    display(
      xs = "inline",
      md = "block"
    ),
  "Block and roll"
)
```

dot-style .style *pronoun*

Description

The `.style` pronoun allows defining styles within the function call of a tag element. Without the `.style` pronoun tag element styles are applied outside and after constructing a tag element.

```
div() %>%
  background_color("primary") %>%
  display("flex") %>%
  flex_justify("between")
```

Once the content of a tag element grows to more than a few lines, associating the element's styles with the element becomes less and less intuitive. In these situations, make use of the `.style` pronoun.

```
div(
  .style %>%
    border_color(theme_primary()) %>%
    text_color(theme_primary()),
  p("Paragraph"),
  p("Paragraph"),
  p("Paragraph")
)
```

flex_align *Flex cross axis alignment*

Description

The `flex_align()` function adjusts a tag element's cross axis alignment. By default, the cross axis is the y-axis. When using `flex_direction("column")` the cross axis becomes the x-axis.

Usage

```
flex_align(x, ...)
```

Arguments

x	A tag element or .style pronoun.
...	A character string specifying the cross axis alignment. One of, • "start" • "end" • "center" • "baseline" • "stretch" Use name-value pairs to specify breakpoints.

Value

An object of the same type as x.

See Also

Other flex utilities: flex_content(), flex_direction(), flex_display(), flex_justify(), flex_wrap(), item_align(), item_fill(), item_grow(), item_order()

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display() %>%
    flex_align("center"),
  div("Flex item"),
  div("Flex item"),
  div("Flex item")
)
```

---

flex_content *Flex content*

Description

The flex_content() function adjusts how a tag element's child elements align as a group on the cross axis (see flex_justify() for more info about the main and cross axis of a flex element). Note, this alignment has no effect on a single row of child elements.

Usage

```
flex_content(x, ...)
```

Arguments

x	A tag element or .style pronoun.
...	A character string specifying the cross axis alignment. One of, • "start" • "end" • "center" • "between" • "around" • "stretch" Use name-value pairs to specify breakpoints.

Value

An object of the same type as x.

See Also

Other flex utilities: flex_align(), flex_direction(), flex_display(), flex_justify(), flex_wrap(), item_align(), item_fill(), item_grow(), item_order()

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display() %>%
    flex_content("center") %>%
    flex_wrap(TRUE),
  div("Flex item"),
  div("Flex item"),
  div("Flex item"),
  div("Flex item"),
  div("Flex item")
)
```

---

flex_direction *Flex direction*

Description

The flex_direction() function adjusts the

Usage

```
flex_direction(x, ...)
```

Arguments

x	A tag element or .style pronoun.
...	A character string specifying a direction. One of, • "row" • "column" Use name-value pairs to specify breakpoints.

Value

An object of the same type as x.

See Also

Other flex utilities: flex_align(), flex_content(), flex_display(), flex_justify(), flex_wrap(), item_align(), item_fill(), item_grow(), item_order()

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display() %>%
    flex_direction("column")
)
```

---

flex_display *Flex display*

Description

The flex_*() functions adjust the flexbox layout of an element. The flexbox layout is incredibly powerful and allows centering of elements vertically and horizontally, automatic adjustment of space between and around child elements, and more. To use flexbox make sure to include flex_display() when styling an element. To adjust an element's display at breakpoints see display().

Direct child elements of a flex box container are automatically considered flex items and may be adjusted with the item_*() functions, see item_align().

Usage

```
flex_display(x)
```

Arguments

x	A tag element or .style pronoun.

Details

Using flexbox, `flex_display()`, a tag element's child elements are considered **flex items**. The `item_*()` functions are used to modify the bahvior of these flex items. So, while `flex_*()` functions are applied to the parent element, all the `item_*()` functions are applied to the individual child flex item elements.

Value

An object of the same type as `x`.

See Also

Other flex utilities: `flex_align()`, `flex_content()`, `flex_direction()`, `flex_justify()`, `flex_wrap()`, `item_align()`, `item_fill()`, `item_grow()`, `item_order()`

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display() %>%
    flex_justify("end"),
  div("Flex item"),
  div("Flex item"),
  div("Flex item")
)
```

---

flex_justify *Flex main axis alignment*

Description

The `flex_justify()` function adjusts a tag element's main axis alignment. By default, the main axis is the x-axis. When using `flex_direction("column")` the main axis becomes the y-axis.

Usage

```
flex_justify(x, ...)
```

Arguments

`x`	A tag element or .style pronoun.
`...`	A character string specifying the main axis alignment. One of, • `"start"` • `"end"`

- "center"
- "between"
- "around"
- "evenly"

Use name-value pairs to specify breakpoints.

Value

An object of the same type as x.

See Also

Other flex utilities: flex_align(), flex_content(), flex_direction(), flex_display(), flex_wrap(), item_align(), item_fill(), item_grow(), item_order()

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display() %>%
    flex_justify("end"),
  div("Flex item"),
  div("Flex item"),
  div("Flex item")
)
```

---

flex_wrap *Flex wrapping*

Description

The flex_wrap() function adjusts how a tag element's child elements wrap, or don't wrap, onto new lines.

Usage

```
flex_wrap(x, ...)
```

Arguments

x	A tag element or .style pronoun.
...	A boolean specifying to wrap or not wrap. One of, • TRUE • FALSE

Value

An object of the same type as `x`.

See Also

Other flex utilities: `flex_align()`, `flex_content()`, `flex_direction()`, `flex_display()`, `flex_justify()`, `item_align()`, `item_fill()`, `item_grow()`, `item_order()`

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display() %>%
    flex_wrap(FALSE),
  div("Flex item"),
  div("Flex item"),
  div("Flex item"),
  div("Flex item")
)
```

`float` *Floats*

Description

The `float()` function places an element to the left or right side of its parent element. Other text and inline elements wrap around floated elements. Note, `float()` has no effect on flex items.

Usage

```
float(x, ...)
```

Arguments

`x`	A tag element or .style pronoun.
`...`	One of the following, • `"left"` • `"l"` • `"right"` • `"r"` • `"none"` Name-value pairs to specify breakpoints.

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

div(
  div(
    .style %>%
      border_color("danger") %>%
      float("left"),
    "6.5/10"
  ),
  div(
    "Considering the need for opening sentences.",
    "We may want to reconsider the necessity of second or third sentences.",
    "The whole problem may be avoided by never creating a problem."
  )
)
```

`focus_color` *Focus ring color*

Description

The `focus_color()` function adjusts the focus shadow color of a tag element.

Usage

```
focus_color(x, color)
```

Arguments

`x`	A tag element or .style pronoun.
`color`	A character string specifying a color. One of,

- `"primary"`
- `"secondary"`
- `"success"`
- `"danger"`
- `"warning"`
- `"info"`
- `"light"`
- `"dark"`
- `"body"`
- `"black"`
- `"white"`

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

tags$button(
  .style %>%
    background_color(theme_primary()) %>%
    focus_color(theme_primary()),
  "Primary themed button with primary themed focus ring"
)
```

`gap_all` *Grid and flex margins*

Description

The `gap_*()` functions adjust the margins of child elements of a tag element with a grid or flex display. Instead of applying margin_*() to each child element, a single `gap_*()` function is applied to the parent element.

Usage

```
gap_all(x, ...)

gap_horizontal(x, ...)

gap_vertical(x, ...)
```

Arguments

`x`	A tag element or .style pronoun.
`...`	A number specifying the space between child elements. One of, • `0` • `1` • `2` • `3` • `4` • `5` Use name-value pairs to specify breakpoints.

Value

An object of the same type as `x`.

See Also

margin_all() for margins on non flex item elements.

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display(),
  div(
    .style %>%
      margin_all(2)
  ),
  div(
    .style %>%
      margin_all(2)
  )
)

div(
  .style %>%
    flex_display() %>%
    gap_all(2),
  div(),
  div()
)
```

---

height_relative *Relative height*

Description

The height_relative() function adjusts a tag element's height relative to the height of its parent element.

Usage

```
height_relative(x, percent)
```

Arguments

x	A tag element or .style pronoun.
percent	A number specifying a percent. One of, • 25 • 50 • 75 • 100

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

div(
  .style %>%
    height_relative(50)
)

div(
  .style %>%
    height_relative(75)
)
```

`item_align` *Flex align self*

Description

The `item_align()` function adjusts

Usage

```
item_align(x, ...)
```

Arguments

`x`	A tag element or .style pronoun.
`...`	A character string specifying an alignment. One of, • `"start"` • `"end"` • `"center"` • `"baseline"` • `"stretch"` Use name-value pairs to specify breakpoints.

Value

An object of the same type as `x`.

See Also

Other flex utilities: `flex_align()`, `flex_content()`, `flex_direction()`, `flex_display()`, `flex_justify()`, `flex_wrap()`, `item_fill()`, `item_grow()`, `item_order()`

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display() %>%
    flex_align("end"),
  div("Flex item"),
  div(
    .style %>%
      item_align("start"),
    "Flex item (self aligned)"
  ),
  div("Flex item")
)
```

`item_fill` *Flex fill*

Description

The `item_fill()` adjusts how tag elements fill a flex element.

Usage

```
item_fill(x, ...)
```

Arguments

`x`	`param_subject()`
`...`	A boolean specifying to fill. One of • `TRUE` Use name-value pairs to specifying breakpoints.

Value

An object of the same type as `x`.

See Also

Other flex utilities: `flex_align()`, `flex_content()`, `flex_direction()`, `flex_display()`, `flex_justify()`, `flex_wrap()`, `item_align()`, `item_grow()`, `item_order()`

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display(),
  div(
    .style %>%
      item_fill(TRUE),
    "Flex item"
  ),
  div(
    .style %>%
      item_fill(TRUE),
    "Flex item"
  ),
  div(
    .style %>%
      item_fill(TRUE),
    "Flex item"
  )
)
```

`item_grow` *Flex grow and shrink*

Description

The `item_grow()` and `item_shrink()` adjust a tag element's ability to grow or shrink inside a flex element.

Usage

```
item_grow(x, ...)

item_shrink(x, ...)
```

Arguments

`x`	A tag element or .style pronoun.
`...`	A boolean specifying to grow or shrink. One of, • `TRUE` • `FALSE` Use name-value pairs to specify breakpoints.

Value

An object of the same type as `x`.

See Also

Other flex utilities: flex_align(), flex_content(), flex_direction(), flex_display(), flex_justify(), flex_wrap(), item_align(), item_fill(), item_order()

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display(),
  div(
    .style %>%
      padding_all(2) %>%
      item_grow(TRUE),
    "Flex item"
  ),
  div(
    .style %>%
      padding_all(2),
    "Flex item"
  ),
  div(
    .style %>%
      padding_all(2),
    "Flex item"
  )
)
```

---

item_order *Flex reordering*

Description

The item_order() function adjusts the visual order of a tag element.

Usage

```
item_order(x, ...)
```

Arguments

x	A tag element or .style pronoun.
...	A number or character string specifying a position. One of, • 0 • 1 • 2

- 3
- 4
- 5
- "first"
- "last"

  Use name-value pairs to specify breakpoints.

Value

An object of the same type as x.

See Also

Other flex utilities: flex_align(), flex_content(), flex_direction(), flex_display(), flex_justify(), flex_wrap(), item_align(), item_fill(), item_grow()

Examples

```
library(htmltools)

div(
  .style %>%
    flex_display(),
  div(
    .style %>%
      item_order(3),
    "Third flex item"
  ),
  div(
    .style %>%
      item_order(2),
    "Second flex item"
  ),
  div(
    "First flex item"
  )
)
```

---

margin_all *Margins*

Description

The margin_*() functions adjust a tag element's margin, the space outside and around the element, its border, and its content.

Usage

```
margin_all(x, ...)

margin_top(x, ...)

margin_right(x, ...)

margin_bottom(x, ...)

margin_left(x, ...)

margin_horizontal(x, ...)

margin_vertical(x, ...)
```

Arguments

x	A tag element or .style pronoun.
...	A number or character string specifying a margin. One or more of, • "-5" • "-4" • "-3" • "-2" • "-1" • 0 • 1 • 2 • 3 • 4 • 5 • "auto" Use name-value pairs to specify breakpoints.

Value

An object of the same type as x.

See Also

gap_all() for flex spacing.

Examples

```
library(htmltools)

div(
  .style %>%
```

```
    margin_left(3) %>%
    margin_right(3),
  "Left margin, right margin"
)

div(
  .style %>%
    margin_horizontal(3),
  "Shorthand for left and right margins"
)

div(
  .style %>%
    margin_horizontal("auto"),
  "A centered element."
)
```

`overflow_all` *Content overflow*

Description

The `overflow_*()` functions adjust how an element's content scrolls.

Usage

```
overflow_all(x, behavior)

overflow_horizontal(x, behavior)

overflow_vertical(x, behavior)
```

Arguments

`x`	A tag element or .style pronoun.
`behavior`	A character string specifying the content overflow behavior. One of, • `"auto"` • `"hidden"` • `FALSE` • `"visible"` • `"scroll"` • `TRUE`

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

div(
  .style %>%
    width_relative(25) %>%
    overflow_horizontal("hidden"),
  "We've really got to drag on this sentence because if we don't",
  "then the example does not demonstrat the utility of the overflow",
  "function"
)
```

`padding_all` *Padding*

Description

The `padding_*()` functions adjust a tag element's padding, the space between the element's border and its content or child elements.

Usage

```
padding_all(x, ...)

padding_top(x, ...)

padding_right(x, ...)

padding_bottom(x, ...)

padding_left(x, ...)

padding_horizontal(x, ...)

padding_vertical(x, ...)
```

Arguments

`x`	A tag element or .style pronoun.
`...`	A number specifying the amount of padding. One of, • `0` • `1` • `2` • `3` • `4` • `5` Use name-value pairs to specify breakpoints.

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

div(
  .style %>%
    border_color(theme_primary()) %>%
    padding_all(2),
  "A padded element"
)
```

`position` *Positioning elements*

Description

The `position_*()` functions adjust set the position of an element.

Usage

```
position(x, method)
```

Arguments

`x`	A tag element or .style pronoun.
`method`	A character string specifying the positioning method. One of,

- `"static"`
- `"relative"`
- `"absolute"`
- `"fixed"`
- `"sticky"`

Value

An object of the same type as `x`.

See Also

Other position utilities: `position_centered()`, `position_sticky()`

Examples

```
library(htmltools)

div(
  .style %>%
    position("absolute") %>%
    position_top(50)
)
```

`position_centered` *Centering positioned elements*

Description

Use `position_centered()` to position an element by its center instead of its edge.

Usage

```
position_centered(x)
```

Arguments

`x`	A tag element or .style pronoun.

Value

An object of the same type as `x`.

See Also

Other position utilities: `position_sticky()`, `position()`

Examples

```
library(htmltools)

div(
  .style %>%
    position("absolute") %>%
    position_right(0) %>%
    position_centered()
)

tags$button(
  type = "button",
  .style %>%
    position("relative") %>%
    background_color(theme_primary()),
```

```
  "Mail",
  span(
    .style %>%
      position("absolute") %>%
      position_top(0) %>%
      position_right(0) %>%
      position_centered() %>%
      rounded_all("pill") %>%
      background_color(theme_secondary()),
    "+99"
  )
)
```

`position_sticky` *Sticky positioning*

Description

Use `position_sticky()` to position an element at the top or bottom of the viewport after scrolling past the element.

Usage

```
position_sticky(x, ...)
```

Arguments

`x`	A tag element or .style pronoun.
`...`	A character string specifying an edge. One of, • `"bottom"` • `"top"` Use name-value pairs to specify breakpoints.

Value

An object of the same type as `x`.

See Also

Other position utilities: `position_centered()`, `position()`

Examples

```
library(htmltools)

div(
  .style %>%
    position_sticky(sm = "top"),
  "Sticks to the top of the viewport on small screens"
)
```

`position_top` *Position offset*

Description

Use `position_<side>()` to adjust the position offset of a tag element.

Usage

```
position_top(x, offset)

position_left(x, offset)

position_bottom(x, offset)

position_right(x, offset)
```

Arguments

`x`	A tag element or .style pronoun.
`offset`	A number specifying a percent. One of, `0`, `50`, or `100`

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

div(
  .style %>%
    position("absolute") %>%
    position_right(0)
)
```

`rounded_all` *Element corners*

Description

The `rounded_*()` functions adjust the corners of a tag element.

Usage

```
rounded_all(x, size)

rounded_top(x, size)

rounded_right(x, size)

rounded_bottom(x, size)

rounded_left(x, size)
```

Arguments

`x`	A tag element or .style pronoun.
`size`	A number or character string specifying a corner size. One of, • `0` • `1` • `2` • `3` • `4` • `5` • `"circle"` • `"pill"` Use `0` to remove rounded corners.

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

div(
  .style %>%
    rounded_all(3)
)
```

```
div(
  .style %>%
    rounded_left("pill")
)
```

---

shadow *Visual depth*

Description

The shadow() function adjusts the box shadow of a tag element.

Usage

```
shadow(x, size)
```

Arguments

x	A tag element or .style pronoun.
size	A character string specifying the shadow size. One of, • "small" • "sm" • "medium" • "md" • "large" • "lg" • "none"

Value

An object of the same type as x.

Examples

```
library(htmltools)

div(
  .style %>%
    shadow("small")
)

div(
  .style %>%
    border_color(theme_warning()) %>%
    background_color(theme_warning()) %>%
    shadow("medium")
)
```

stack_vertical *Quick flex layouts*

Description

The `stack_vertical()` and `stack_horizontal()` functions are shortcuts for creating vertical and horizontal flex layouts.

Usage

```
stack_vertical(x)

stack_horizontal(x)
```

Arguments

x	A tag element or .style pronoun.

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

div(
  .style %>%
    stack_vertical() %>%
    gap_all(3),
  p(
    .style %>%
      border_all() %>%
      padding_all(2),
    "First item"
  ),
  p(
    .style %>%
      border_all() %>%
      padding_all(2),
    "Second item"
  ),
  p(
    .style %>%
      border_all() %>%
      padding_all(2),
    "Third item"
  )
)
```

`text_alignment` *Align text*

Description

The `text_alignment()` function adjusts how the text within a tag element is aligned.

Usage

```
text_alignment(x, ...)
```

Arguments

`x`	A tag element or .style pronoun.
`...`	A character string specifying an alignment. One of, `"left"`, `"right"`, or `"center"`. Use name-value pairs to specify breakpoints.

Value

An object of the same type as `x`.

See Also

Other text utilities: `text_break()`, `text_color()`, `text_decoration()`, `text_height()`, `text_selection()`, `text_style()`, `text_transform()`, `text_weight()`, `text_wrap()`

Examples

```
library(htmltools)

div(
  .style %>%
    text_alignment("left")
)

div(
  .style %>%
    text_alignment("center")
)
```

`text_break` *Break long text*

Description

The `text_break()` function is used to break long strings of text within a tag element.

Usage

```
text_break(x)
```

Arguments

`x`	A tag element or .style pronoun.

Value

An object of the same type as `x`.

See Also

Other text utilities: `text_alignment()`, `text_color()`, `text_decoration()`, `text_height()`, `text_selection()`, `text_style()`, `text_transform()`, `text_weight()`, `text_wrap()`

Examples

```
library(htmltools)

div(
  .style %>%
    text_break(),
  "wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww"
)
```

`text_color` *Text color*

Description

The `text_color()` function adjusts the color of text in a tag element.

Usage

```
text_color(x, color)

text_emphasis(x, color)
```

Arguments

x	A tag element or .style pronoun.
color	A character string specifying a color. One of,

- "primary"
- "secondary"
- "success"
- "danger"
- "warning"
- "info"
- "light"
- "dark"
- "body"
- "black"
- "white"

Value

An object of the same type as x.

See Also

Other text utilities: text_alignment(), text_break(), text_decoration(), text_height(), text_selection(), text_style(), text_transform(), text_weight(), text_wrap()

Examples

```
library(htmltools)

div(
  .style %>%
    text_color(theme_primary())
)

div(
  .style %>%
    text_color(theme_dark()) %>%
    border_color(theme_dark())
)
```

`text_decoration` *Text decoration*

Description

The `text_dectoration()` function adjusts how text is decorated within a tag element.

Usage

```
text_decoration(x, decoration)
```

Arguments

`x`	A tag element or .style pronoun.
`decoration`	A character string specifying a decoration. One of, • `"underline"` • `"strike"` • `"none"`

Value

An object of the same type as `x`.

See Also

Other text utilities: `text_alignment()`, `text_break()`, `text_color()`, `text_height()`, `text_selection()`, `text_style()`, `text_transform()`, `text_weight()`, `text_wrap()`

Examples

```
library(htmltools)

div(
  .style %>%
    text_decoration("strike"),
  "TODO: buy milk"
)

div(
  .style %>%
    text_emphasis(theme_danger()) %>%
    text_decoration("underline"),
  "Red AND underlined!"
)
```

text_height *Text height*

Description

The `text_height()` function adjusts the line height of text within a tag element.

Usage

```
text_height(x, height)
```

Arguments

`x`	A tag element or .style pronoun.
`height`	A character string specifying a line height. One of, • `"sm"` • `"small"` • `"base"` • `"lg"` • `"large"`

Value

An object of the same type as `x`.

See Also

Other text utilities: `text_alignment()`, `text_break()`, `text_color()`, `text_decoration()`, `text_selection()`, `text_style()`, `text_transform()`, `text_weight()`, `text_wrap()`

Examples

```
library(htmltools)

p(
  .style %>%
    text_height("base"),
  "This is the browser's default line height spacing.",
  "Make sure there are multiple lines of text in a paragraph.",
  "Otherwise, the changes applied by these utilities are not visible."
)
```

`text_selection` *Text selection*

Description

The `text_selection()` function adjusts how text is selected within a tag element when the user clicks on the element.

Usage

```
text_selection(x, select)
```

Arguments

`x`	A tag element or .style pronoun.
`select`	A character string specifying how text is selected. One of, • `"all"` • `"auto"` • `"none"`

Value

An object of the same type as `x`.

See Also

Other text utilities: `text_alignment()`, `text_break()`, `text_color()`, `text_decoration()`, `text_height()`, `text_style()`, `text_transform()`, `text_weight()`, `text_wrap()`

Examples

```
library(htmltools)

div(
  .style %>%
    text_selection("all"),
  "Click to select all the text"
)
```

`text_size` *Text size*

Description

The `text_size()` function adjusts the font size of a tag element.

Usage

```
text_size(x, size)
```

Arguments

`x`	A tag element or .style pronoun.
`size`	A number specifying a font size for the text. One of, • 1 • 2 • 3 • 4 • 5 • 6 The sizes follow the conventions of HTML heading tags, so 1 is the largest font and 6 the smallest.

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

p(
  .style %>%
    text_size(1),
  "Largest size"
)

p(
  .style %>%
    text_size(6),
  "Smallest size"
)
```

`text_style` *Text style*

Description

The `text_style()` function adjusts the style of text in a tag element.

Usage

```
text_style(x, style)
```

Arguments

`x`	A tag element or .style pronoun.
`style`	A character string specifying the text style. One of, • `"italic"` • `"normal"`

Value

An object of the same type as `x`.

See Also

Other text utilities: `text_alignment()`, `text_break()`, `text_color()`, `text_decoration()`, `text_height()`, `text_selection()`, `text_transform()`, `text_weight()`, `text_wrap()`

Examples

```
library(htmltools)

p(
  .style %>%
    text_style("italic"),
  "Italic text"
)

p(
  .style %>%
    text_style("normal"),
  "Normal text"
)
```

`text_transform` *Text transformation*

Description

The `text_transform()` function adjusts the case of text in a tag element.

Usage

```
text_transform(x, transform)
```

Arguments

`x`	A tag element or .style pronoun.
`transform`	A character string specifying the transform. One of, • `"uppercase"` • `"lowercase"` • `"capitalize"`

Value

An object of the same type as `x`.

See Also

Other text utilities: `text_alignment()`, `text_break()`, `text_color()`, `text_decoration()`, `text_height()`, `text_selection()`, `text_style()`, `text_weight()`, `text_wrap()`

Examples

```
library(htmltools)

div(
  .style %>%
    text_transform("lowercase"),
  "TRANSFORMED TO LOWERCASE"
)

div(
  .style %>%
    text_color(theme_warning()) %>%
    text_transform("uppercase"),
  "transformed to uppercase"
)
```

text_weight *Text weight*

Description

The `text_weight()` function adjusts the font weight of a tag element.

Usage

```
text_weight(x, weight)
```

Arguments

`x`	A tag element or .style pronoun.
`weight`	A character string specifying the font weight. One of, • `"bold"` • `"bolder"` • `"semibold"` • `"medium"` • `"normal"` • `"light"` • `"lighter"` `"bolder"` and `"lighter"` change the font weight relative to the current font weight.

Value

An object of the same type as `x`.

See Also

Other text utilities: `text_alignment()`, `text_break()`, `text_color()`, `text_decoration()`, `text_height()`, `text_selection()`, `text_style()`, `text_transform()`, `text_wrap()`

Examples

```
library(htmltools)

p(
  .style %>%
    text_weight("bold"),
  "Bold"
)

p(
  .style %>%
    text_weight("light"),
```

```
  "Light"
)
```

`text_wrap` *Text wrap*

Description

The `text_wrap()` adjusts how text is wrapped, or not wrapped, in a tag element.

Usage

```
text_wrap(x, wrap)
```

Arguments

`x`	A tag element or .style pronoun.
`wrap`	A boolean. One of, • `TRUE` • `FALSE`

Value

An object of the same type as `x`.

See Also

Other text utilities: `text_alignment()`, `text_break()`, `text_color()`, `text_decoration()`, `text_height()`, `text_selection()`, `text_style()`, `text_transform()`, `text_weight()`

Examples

```
library(htmltools)

div(
  style = "width: 5rem;",
  .style %>%
    background_color(theme_primary()) %>%
    text_wrap(TRUE),
  "Wrap text to fit the element"
)

div(
  style = "width: rem;",
  .style %>%
    background_color(theme_secondary()) %>%
    text_wrap(FALSE),
  "This text won't wrap onto a new line."
```

```
)
```

`theme_primary` *Theme colors*

Description

Theme color functions.

Usage

```
theme_primary()

theme_secondary()

theme_success()

theme_danger()

theme_warning()

theme_info()

theme_light()

theme_dark()
```

Value

A character string.

Examples

```
library(htmltools)

div(
  .style %>%
    background_color("primary")
)

div(
  .style %>%
    background_color(theme_primary())
)
```

`vertical_alignment` *Vertical alignment*

Description

The `vertical_alignment()` function adjusts the inline position of inline, inline-block, and table cell elements. The utility may be used to adjust the vertical alignment of an image in a line of text or the contents of a table cell.

Usage

```
vertical_alignment(x, alignment)
```

Arguments

`x`	A tag element or .style pronoun.
`alignment`	A character string specifying an alignment. One of,

- `"baseline"`
- `"top"`
- `"middle"`
- `"bottom"`
- `"text-top"`
- `"text-bottom"`

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

div(
  "Text",
  span("Above") %>%
    vertical_alignment("top"),
  span("Below") %>%
    vertical_alignment("bottom")
)
```

visible *Visibility*

Description

The `visible()` function adjusts the visibility of a tag element. An invisible element is both visually hidden and is also hidden from screen readers.

Usage

```
visible(x, show)
```

Arguments

`x`	A tag element or .style pronoun.
`show`	A boolean specifying the visibility. One of, • `TRUE` • `FALSE`

Value

An object of the same type as `x`.

Examples

```
library(htmltools)

div(
  .style %>%
    visible(FALSE),
  "Not visible"
)
```

width_relative *Relative width*

Description

The `width_relative()` function adjusts a tag element's width relative to its parent element's width.

Usage

```
width_relative(x, percent)
```

Arguments

x	A tag element or .style pronoun.
percent	A number specifying a percent. One of, • 25 • 50 • 75 • 100

Value

An object of the same type as x.

Examples

```
library(htmltools)

div(
  .style %>%
    width_relative(25)
)

div(
  .style %>%
    width_relative(100)
)
```

Index