Package: quartools (via r-universe)

August 23, 2024

Title Programmatic Element Creation For Quarto Documents

Version 0.0.0.9000

Description Programatically generate quarto-compliant markdown elements.

License GPL (>= 3)

URL https://github.com/ElianHugh/quartools

BugReports https://github.com/ElianHugh/quartools/issues

Imports rlang (>= 1.1.0)

Suggests htmltools, knitr, quarto, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3.9000

Repository https://elianhugh.r-universe.dev

RemoteUrl https://github.com/ElianHugh/quartools

RemoteRef HEAD

RemoteSha 0c7b29a96a8107ff0c7db0a718db1df3e49d7e60

Contents

`map_qto` *Apply a function to each element of a vector and return Quarto block vector*

Description

`map_qto()` loops a list over a package function defined by .type or a custom function that returns a quarto block output. This function always returns a list of quarto block objects.

Usage

```
map_qto(
  .x,
  .f = NULL,
  ...,
  .type = c("block", "div", "callout", "heading"),
  .sep = "",
  .collapse = "",
  call = caller_env()
)
```

Arguments

`.x` An input vector.

`.f` Optional function to apply to each element. If function does not return a "quarto_block" class object, the output is passed to `qto_block()`

`...` Additional parameters passed to function defined by `.f`.

`.type` If .f is `NULL`, type is used to define the function applied to each element of the vector. Options include "block", "div", "callout", or "heading".

`.sep, .collapse` Additional parameters passed to `qto_block()` if .f does not return a quarto block class object. Ignored if .f does return a quarto block class object.

`call` The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

See Also

`pmap_qto()`, `purrr::map()`

Examples

```
qto_list <- map_qto(
    list("This is a note.", "And this is a note.", "And this is a note"),
    .type = "callout"
)

qto_block(qto_list)
```

pmap_qto *Map over multiple inputs simultaenously and return Quarto block vector*

Description

`pmap_qto()` loops a list of vectors over a package function defined by .type or a custom function that returns a quarto block output. This function always returns a list of quarto block objects.

Usage

```
pmap_qto(
  .l,
  .f = NULL,
  ...,
  .type = c("block", "div", "callout", "heading"),
  .sep = "",
  .collapse = "",
  call = caller_env()
)
```

Arguments

`.l` An input vector.

`.f` Optional function to apply to each element. If function does not return a "quarto_block" class object, the output is passed to `qto_block()`

`...` Additional parameters passed to function defined by `.f`.

`.type` If .f is `NULL`, type is used to define the function applied to each element of the vector. Options include "block", "div", "callout", or "heading".

`.sep`, `.collapse` Additional parameters passed to `qto_block()` if .f does not return a quarto block class object. Ignored if .f does return a quarto block class object.

`call` The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

See Also

`map_qto()`, `purrr::pmap()`

Examples

```
qto_list <- pmap_qto(
    list(
        list("Answer:", "Answer:", "Answer:"),
        list("Yes", "No", "Yes")
    )
)
qto_block(qto_list)

qto_list <- pmap_qto(
    mtcars[seq(3L), seq(3L)],
    function(mpg, cyl, disp) {
        qto_li(
            .list = list(
                sprintf("mpg is: %s", mpg),
                sprintf("cyl is: %s", cyl),
                sprintf("disp is: %s", disp)
            )
        )
    }
)
qto_block(qto_list)
```

qto_attributes *Create an attribute string for a div or span*

Description

`qto_attributes()` creates an attribute string used by `qto_div()`, `qto_span()`, or `qto_fig_span()`.

Usage

```
qto_attributes(
  id = NULL,
  class = NULL,
  css = NULL,
  ...,
  .attributes = NULL,
  .output = "embrace",
  .drop_empty = TRUE,
  call = caller_env()
)
```

Arguments

`id`	Div or span identifier. If `id` does not start with `"#"`, the hash character is applied as a prefix.

`class`	Div or span class. If `class` does not start with `"."`, the period character is applied as a prefix.
`css`	If `{htmltools}` is installed, a list of css style attributes to pass to `htmltools::css()`.
`...`	Optional named attributes.
`.attributes`	Optional list of attributes. If supplied, any attributes passed to `...` are ignored.
`.output`	Output type. If "embrace", the returned attributes are always enclosed in curly brackets, e.g. "" if no attributes are supplied. If "span", an empty string is returned if no attributes are provided.
`.drop_empty`	If `TRUE`, empty attributes are dropped.
`call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Examples

```
qto_attributes(id = "id", class = "class")

qto_attributes(class = "class", key1 = "val", key2 = "val")

qto_attributes(width = 4)
```

---

qto_basics *Basic formatting and markdown elements*

Description

`qto_heading()` creates headings and `qto_hr()` creates horizontal rules.

Usage

```
qto_heading(..., level = 1L)

qto_hr(rule = "-", length = 72L, before = "\n\n", after = before)
```

Arguments

`...`	Heading text passed sto `qto_block()`.
`level`	Heading level. Defaults to 1.
`rule, length`	Horizontal rule character and length of rule.
`before, after`	Text to insert before and after a horizontal rule.

qto_block *Create a block of text to render as Markdown text in Quarto*

Description

qto_block() passes the input to paste() and assigns the classes "knit_asis" and "quarto_block".

Usage

```
qto_block(..., sep = "", collapse = "", call = caller_env())
```

Arguments

...	dots to convert to character vector
sep	a character string to separate the terms. Not NA_character_.
collapse	an optional character string to separate the results. Not NA_character_. When collapse is a string, the result is always a string (character of length 1).
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Value

character vector of length 1

See Also

- knitr::asis_output()

Examples

```
qto_block("Hello world!")

qto_block("Hello", "world!", sep = " ")

qto_block("- ", LETTERS[1:4], collapse = "\n")
```

qto_callout *Create a Quarto callout block*

Description

Callouts are an excellent way to draw extra attention to certain concepts, or to more clearly indicate that certain content is supplemental or applicable to only some scenarios. https://quarto.org/docs/authoring/callouts.html

Usage

```
qto_callout(
  ...,
  type = c("note", "tip", "warning", "caution", "important"),
  collapse = NULL,
  appearance = NULL,
  icon = NULL,
  title = NULL,
  id = NULL,
  class = NULL,
  .attributes = NULL,
  call = caller_env()
)
```

Arguments

`...` Arguments passed on to `qto_div`

- `.content` If `.content` is supplied, any values passed to `...` are ignored. If `.content` is `NULL`, it is set as all values passed to `...`.
- `drop_empty` If `TRUE`, drop empty values from `.content` or `...`
- `drop_na` If `TRUE`, drop `NA` values from `.content` or `...`
- `css` If `{htmltools}` is installed, a list of css style attributes to pass to `htmltools::css()`.

`type` Callout type. One of "note", "tip", "warning", "caution", or "important". https://quarto.org/docs/authoring/callouts.html#callout-types

`collapse` If `TRUE`, create a folded callout. https://quarto.org/docs/authoring/callouts.html#collapse

`appearance` Callout appearance. One of "default", "simple", "minimal". https://quarto.org/docs/authoring/callouts.html#appearance

`icon` If `FALSE`, create a callout with an icon. https://quarto.org/docs/authoring/callouts.html#icons

`title` Title attribute. Works the same as included a level 2 heading before the body of the callout contents.

`id` Div or span identifier. If `id` does not start with `"#"`, the hash character is applied as a prefix.

`class` Div or span class. If `class` does not start with `"."`, the period character is applied as a prefix.

`.attributes` Optional list of attributes. If supplied, any attributes passed to `...` are ignored.

`call` The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Examples

```
qto_callout(
  "Callouts provide a simple way to attract attention, for example, to this warning.",
  type = "warning"
)

qto_callout(
  "This is an example of a callout with a title.",
  type = "tip",
  title = "Tip with a title"
)
```

qto_div *Create a Quarto div with optional classes, attributes, and other identifiers*

Description

https://quarto.org/docs/authoring/markdown-basics.html#divs-and-spans

Usage

```
qto_div(
  ...,
  id = NULL,
  class = NULL,
  css = NULL,
  .attributes = NULL,
  .content = NULL,
  collapse = "",
  drop_empty = TRUE,
  drop_na = TRUE,
  call = caller_env()
)
```

Arguments

`...` Optional named attributes.

`id` Div or span identifier. If `id` does not start with `"#"`, the hash character is applied as a prefix.

`class` Div or span class. If `class` does not start with `"."`, the period character is applied as a prefix.

`css` If `{htmltools}` is installed, a list of css style attributes to pass to `htmltools::css()`.

`.attributes` Optional list of attributes. If supplied, any attributes passed to `...` are ignored.

`.content` If `.content` is supplied, any values passed to `...` are ignored. If `.content` is `NULL`, it is set as all values passed to `...`.

`collapse` Passed to `base::paste0()` with `.content`.

`drop_empty` If `TRUE`, drop empty values from `.content` or `...`

`drop_na` If `TRUE`, drop `NA` values from `.content` or `...`

`call` The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

See Also

- `qto_callout()`

Examples

```
# div with an class
qto_div(
  "This content can be styled with a border",
  class = "border"
)

# Nested div
qto_div(
  qto_div("Here is a warning.", class = "warning"),
  "More content.",
  id = "special",
  class = "sidebar"
)
```

`qto_dl` *Create a definition list*

Description

See the Quarto documentation for more information on lists: `https://quarto.org/docs/authoring/markdown-basics.html#lists`

Usage

```
qto_dl(
  ...,
  .term = NULL,
  .definition = NULL,
  .list = NULL,
  .drop_na = FALSE,
  .sep = "\n",
  .replacement = ""
)
```

Arguments

`...` Optional. Named argument where the argument name is a term and the value name is the text to display as a definition for the term. Ignored if `.list` or `.term` and `.definition` are supplied.

`.term`, `.definition` If supplied, `.list` is set to a named list using terms as names and definitions as values. Ignored if `.list` is supplied.

`.list` If `.list` is supplied, any values passed to `...` are ignored. One of `.list`, `.term` and `.definition`, or `...` must be supplied.

`.drop_na` If `TRUE`, drop `NA` values from the supplied list.

`.sep` Added separator string between terms and definitions. Defaults to `"\n"` for wider list spacing. If set to `""`, the definition list uses a tighter spacing.

`.replacement` If not `NULL`, replace NA values from the supplied list with the value of replacement.

See Also

- `qto_ol()`
- `qto_li()`

Examples

```
qto_dl(
  "Quarto" = "An open-source scientific and technical publishing system"
)

qto_dl(
  .term = "Quarto",
  .definition = "An open-source scientific and technical publishing system"
)

qto_dl(
  .list = list("Quarto" = "An open-source scientific and technical publishing system")
)
```

qto_fig *Insert an figure or image*

Description

qto_fig() creates Markdown formatting to insert a figure or image. Learn more about figure formatting options in the Quarto documentation: https://quarto.org/docs/authoring/figures.html Note that LaTeX output formats may not support external images: https://tex.stackexchange.com/questions/5433/can-i-use-an-image-located-on-the-web-in-a-latex-document

Usage

```
qto_fig(
  src,
  caption = NULL,
  alt = NULL,
  reference = NULL,
  align = NULL,
  pos = NULL,
  width = NULL,
  height = NULL,
  title = NULL,
  ...,
  allow_missing = TRUE
)
```

Arguments

src	Figure file path or URL.
caption	Figure caption.
alt	Figure alt text.
reference	Figure reference. "fig-" prefix is optional.
align, pos, width, height	Figure alignment, position, width, and height.
title	Figure title.
...	Additional attributes passed to attributes argument of qto_fig_span().
allow_missing	If FALSE, error if src is not an existing file or a valid URL. URLs are not checked if they work.

See Also

knitr::include_graphics()

Other span: qto_link(), qto_span()

Examples

```
qto_fig("image.jpeg", "Image caption", alt = "Image alt text")

qto_fig("graphic.pdf", title = "Title of the graphic")
```

qto_link *Create a Markdown link*

Description

Create a Markdown link

Usage

```
qto_link(src, text = NULL, ..., allow_empty = FALSE, call = caller_env())
```

Arguments

src
: Path or URL for link. Required.

text
: Optional link text. If link text is not provided, a bare link, e.g. `<https://quarto.org>` is returned.

...
: Arguments passed on to `qto_attributes`

  `id` Div or span identifier. If `id` does not start with `"#"`, the hash character is applied as a prefix.

  `class` Div or span class. If `class` does not start with `"."`, the period character is applied as a prefix.

  `css` If `{htmltools}` is installed, a list of css style attributes to pass to `htmltools::css()`.

  `.attributes` Optional list of attributes. If supplied, any attributes passed to `...` are ignored.

allow_empty
: If `FALSE` (default), error if string is empty. If `TRUE`, allow input to be an empty string.

call
: The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

See Also

Other span: `qto_fig()`, `qto_span()`

Examples

```
qto_link("https://quarto.org")

qto_link("https://quarto.org", "Quarto")

qto_link("https://quarto.org", "Quarto", class = "smaller")
```

qto_list *Create ordered or unordered lists for Quarto*

Description

qto_ol() created ordered lists and qto_li() creates unordered lists. See the Quarto documentation for more information on lists: https://quarto.org/docs/authoring/markdown-basics.html#lists

Usage

```
qto_ol(.list = NULL, ..., level = 1L, symbol = "1.")

qto_li(.list = NULL, ..., level = 1L, symbol = "*", sep = "")
```

Arguments

`.list` A vector of list items. Optional if items are passed to `...`.

`...` Items to use as list items. Ignored if .list is supplied.

`level` Indent level of list. Nested levels in a single function call are not yet supported.

`symbol` Symbol to use for list bullet. Use "(@)" to create a list where numbering continues after interruption.

`sep` Separator character between list items passed to qto_block(). Defaults to "". Set to "\n" for wider list spacing.

See Also

qto_dl()

Examples

```
qto_ol(LETTERS[1:2])

qto_li(LETTERS[3:4])

qto_li(LETTERS[5:6], level = 2)

qto_ol(LETTERS[7:8], sep = "\n")

qto_li(LETTERS[5:6], symbol = c("*", "    +"))
```

qto_shortcode *Create a Quarto shortcode*

Description

qto_shortcode() creates a shortcode based on an input string.

Usage

```
qto_shortcode(x, before = "{{< ", after = " >}}", ...)

qto_video(src)

qto_pagebreak()

qto_kbd(...)
```

Arguments

x — An input string.

before, after — Text to insert before and after the supplied string.

... — For qto_kbd(), a set of strings to combine with "-" or a named set of attributes where names are one of "mac", "win", or "linux".

src — For qto_video(), a URL with the video source.

Examples

```
qto_video("https://www.youtube.com/embed/wo9vZccmqwc")

qto_pagebreak()

qto_kbd("Shift", "Ctrl", "P")

qto_kbd(
  mac = "Shift-Command-O",
  win = "Shift-Control-O",
  linux = "Shift-Ctrl-L"
)
```

qto_span *Create a Quarto span*

Description

See the Quarto documentation for more on divs and spans: https://quarto.org/docs/authoring/markdown-basics.html#divs-and-spans

Usage

```
qto_span(text, ..., .attributes = NULL)
```

Arguments

`text` Text to wrap in span bracket.

`...` Arguments passed on to `qto_attributes`

- `id` Div or span identifier. If `id` does not start with "#", the hash character is applied as a prefix.
- `class` Div or span class. If `class` does not start with ".", the period character is applied as a prefix.
- `css` If `{htmltools}` is installed, a list of css style attributes to pass to `htmltools::css()`.
- `.output` Output type. If "embrace", the returned attributes are always enclosed in curly brackets, e.g. "" if no attributes are supplied. If "span", an empty string is returned if no attributes are provided.
- `.drop_empty` If `TRUE`, empty attributes are dropped.
- `call` The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

`.attributes` Optional list of attributes. If supplied, any attributes passed to `...` are ignored.

See Also

Other span: `qto_fig()`, `qto_link()`

Examples

```
qto_span("This is *some text*", class = "class", key = "value")

qto_span("This is good", id = "id", class = "class", key1 = "val1", key2 = "val2")
```

`with_body_column` *Temporarily modify page layout*

Description

Create a div block that modifies the current quarto layout column temporarily.

See the Quarto documentation for a full list of available extension options.

Usage

```
with_body_column(
  ...,
  outset = FALSE,
  id = NULL,
  class = NULL,
  extension = NULL,
  .attributes = NULL,
  call = caller_env()
)

with_page_column(
  ...,
  id = NULL,
  class = NULL,
  extension = NULL,
  .attributes = NULL,
  call = caller_env()
)

with_screen_inset_column(
  ...,
  id = NULL,
  class = NULL,
  extension = NULL,
  .attributes = NULL,
  call = caller_env()
)

with_screen_column(
  ...,
  id = NULL,
  class = NULL,
  extension = NULL,
  .attributes = NULL,
  call = caller_env()
)
```

```
with_margin_column(..., id = NULL, .attributes = NULL, call = caller_env())
```

Arguments

- `...` Arguments passed on to `qto_div`
  - `.content` If `.content` is supplied, any values passed to `...` are ignored. If `.content` is `NULL`, it is set as all values passed to `...`.
  - `collapse` Passed to `base::paste0()` with `.content`.
  - `drop_empty` If `TRUE`, drop empty values from `.content` or `...`
  - `drop_na` If `TRUE`, drop `NA` values from `.content` or `...`
  - `css` If `{htmltools}` is installed, a list of css style attributes to pass to `htmltools::css()`.
- `outset` If `TRUE`, the ouset affix is applied to the column class
- `id` Div or span identifier. If `id` does not start with `"#"`, the hash character is applied as a prefix.
- `class` Div or span class. If `class` does not start with `"."`, the period character is applied as a prefix.
- `extension` affix to apply to column class
- `.attributes` Optional list of attributes. If supplied, any attributes passed to `...` are ignored.
- `call` The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

character vector of length 1

Examples

```
with_body_column("Hello world!")
```

Index