Package 'xaringanExtra'

July 16, 2022

Title Extras and Extensions for 'xaringan' Slides

Version 0.7.0

Description Extras and extensions for 'xaringan' slides. Navigate your slides with tile view. Make your slides editable, live! Announce slide changes with subtle tones. Animate slide transitions with 'animate.css'. Add tabbed panels to slides with 'panelset'. Use the 'Tachyons CSS' utility toolkit for rapid slide development. Scribble on your slides. Add a copy button to your code chunks with 'clipboard'. Add a logo or top or bottom banner to every slide. Broadcast slides to stay in sync with remote viewers. Include yourself in your slides with 'webcam'. Plus a whole lot more!

License MIT + file LICENSE

URL https://pkg.garrickadenbuie.com/xaringanExtra/, https://github.com/gadenbuie/xaringanExtra

BugReports https://github.com/gadenbuie/xaringanExtra/issues

Imports htmltools, jsonlite, knitr, utils, uuid

Suggests callr, rmarkdown, testthat (>= 2.1.0), xaringan

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.0

Collate 'animate.R' 'banner.R' 'broadcast.R' 'clipboard.R' 'editable.R' 'fit-screen.R' 'freezeframe.R' 'panelset.R' 'progress-bar.R' 'scribble.R' 'share_again.R' 'slide-tone.R' 'styles.R' 'search.R' 'tachyons.R' 'tile-view.R' 'use_logo.R' 'utils.R' 'webcam.R' 'xaringanExtra-package.R'

NeedsCompilation no

Author Garrick Aden-Buie [aut, cre] (<https://orcid.org/0000-0002-7111-0077>), Matthew T. Warkentin [aut] (Contributed scribble, <https://orcid.org/0000-0001-8730-3511>), Yotam Mann [cph] (tone.js), Daniel Eden [cph] (animate.css),

Tachyons authors [cph],
Klaus Hartl, Fagner Brack, GitHub Contributors [cph] (js-cookie),
Chris Andrejewski [cph] (himalaya),
Eric Londaits [cph] (text-poster.js),
Zeno Rocha [cph] (clipboard.js),
Nikita Karamov [cph] (shareon.js),
Ross Zurowski [cph] (fitvids.js),
Michelle Bu and Eric Zhang [cph] (peerjs),
Kiril Vatev [cph] (tiny.toast),
André Restive [cph] (remark.search),
Printio (Juriy Zaytsev, Maxim Chernyak) [cph] (fabric.js),
Christpher Antonellis [cph] (freezeframe.js)

Maintainer Garrick Aden-Buie <garrick@adenbuie.com>

Repository CRAN

Date/Publication 2022-07-16 13:00:02 UTC

R topics documented:

`animate_css` *Animate.css*

Description

Animate.css is a popular collection of CSS animations. It contains "a bunch of cool, fun, and cross-browser animations for you to use in your projects. Great for emphasis, home pages, sliders, and general just-add-water-awesomeness."

Usage

```
use_animate_css(minified = FALSE, xaringan = TRUE)

html_dependency_animate_css(minified = FALSE, xaringan = TRUE)

use_animate_all(
  style = c("slide_left", "slide_right", "slide_up", "slide_down", "roll", "fade")
)
```

Arguments

`minified`	Should the minified or full CSS source be used?
`xaringan`	When `TRUE`, the `.animated` selector is modified so that the animation is only applied when the slide is visible. Without this, presentations with many animated slides may have poor performance, especially on page load. Set to `FALSE` to use animate.css in other HTML-based documents.
`style`	Animation style to be used for all slides. • `slide_left`: Slide in from the right and out to the left • `slide_right`: Slide in from the left and out to the right • `slide_up`: Slide in from the bottom and out to the top • `slide_down`: Slide in from the top and out to the bottom • `roll`: Roll in from the left and roll out to the right • `fade`: Fade in

Value

An `htmltools::tagList()` with the tile view dependencies, or an `htmltools::htmlDependency()`.

Functions

- `use_animate_css`: Use animate.css in your xaringan slides.
- `html_dependency_animate_css`: Returns an `htmltools::htmlDependency()` with the tile view dependencies. Most users will want to use `use_animate_css()`.
- `use_animate_all`: Use a default animation for all slides. Sets coupled in an out animations for all slides that can be disabled on individual slides by adding the class `.no-animation`.

Usage

To add animate.css to your xaringan presentation, add the following code chunk to your slides' R Markdown file.

````
```{r xaringan-animate, echo=FALSE}
xaringanExtra::use_animate_css()
```

---
class: animated fadeInLeft slideOutRight

This slide fades in from the left and slides out to the right!
````

Note that when `xaringan = TRUE`, as is the default, out animations are only applied to slides that are exiting so that you can specify both in and out behavior of each slide.

Or use `use_animate_all()` to set default in and out animations for all slides. Animations can be disabled for individual slides by adding the class `.no-animation` to the slide.

````
```{r xaringan-animate, echo=FALSE}
xaringanExtra::use_animate_all("slide_left")
```
````

References

See animate.css for a full list of animations.

Examples

```
use_animate_css()
html_dependency_animate_css()
```

---

clipboard *Clipboard*

Description

Add a "Copy Code" button for one-click code chunk copying.

Usage

```
use_clipboard(
  button_text = "Copy Code",
  success_text = "Copied!",
  error_text = "Press Ctrl+C to Copy",
  selector = NULL,
```

```
  minified = TRUE
)

html_dependency_clipboardjs(minified = TRUE)

html_dependency_clipboard()
```

Arguments

`button_text, success_text, error_text`	Text (or HTML) shown in the copy button by default (*button*), on copy *success*, or in the event of an *error*.
`selector`	The CSS selector used to identify the elements that will receive the copy code button. If `NULL`, the extension will automatically choose the selector for **xaringan** slides or general R Markdown. The CSS selector should identify the parent container that holds the content to be copied. The copy button will be added as the last element in this container, and then the text of every element inside the container identified by the selector, minus the copy button text, is copied to the clipboard.
`minified`	Should the minified clipboardjs dependency be used?

Details

To add **clipboard** to your xaringan presentation or R Markdown document, add the following code chunk to your slides' R Markdown file.

````
```{r xaringanExtra-clipboard, echo=FALSE}
xaringanExtra::use_clipboard()
```
````

You can also customize the text that is shown by default when hovering over a code chunk with the `button_text` argument. Use `success_text` to specify the text shown when the copy action works, or `error_text` for the text shown when the copy action fails. If the copy action fails, the text will still be selected, so the user can still manually press `Ctrl+C` to copy the code chunk.

These options accept raw HTML strings, so you can achieve an icon-only appearance using FontAwesome icons:

````
```{r xaringanExtra-clipboard, echo=FALSE}
htmltools::tagList(
  xaringanExtra::use_clipboard(
    button_text = "<i class=\"fa fa-clipboard\"></i>",
    success_text = "<i class=\"fa fa-check\" style=\"color: #90BE6D\"></i>",
    error_text = "<i class=\"fa fa-times-circle\" style=\"color: #F94144\"></i>"
  ),
  rmarkdown::html_dependency_font_awesome()
)
```
````

Value

An `htmltools::tagList()` with the HTML dependencies required for **clipboard**.

Functions

- `use_clipboard`: Adds **clipboard** to your xaringan slides or R Markdown HTML output.
- `html_dependency_clipboardjs`: Returns an `htmltools::htmlDependency()` with the clipboard.js library. For expert use.
- `html_dependency_clipboard`: Returns an `htmltools::htmlDependency()` with the clipboard dependencies for use in xaringan and R Markdown documents. Most users will want to use `use_clipboard()` instead.

References

https://clipboardjs.com/

Examples

```
use_clipboard()
```

---

css_position *Helper to set absolute position of an element.*

Description

Sets position for an absolutely positioned element. Setting one of top or bottom or one of left or right will "unset" the other. It's probably not a good idea to set both top and bottom or right and left.

Usage

```
css_position(top = "1em", right = "1em", left = NULL, bottom = NULL)
```

Arguments

`top`, `right`, `bottom`, `left`
: The position of the element in distance from the top, right, bottom, or left edge of it's container element.

Value

An object of class `css_position` that describes `top`, `right`, `bottom`, and `left` positions.

Examples

```
css_position(top = "1em", right = "1em") # top right corner
css_position(top = "1em", left = "1em") # top left corner
css_position(bottom = 0, right = 0) # bottom right corner
```

editable *Editable*

Description

Editable gives you a way to write directly inside your slides. Make any element of your slides editable by using the `.can-edit[...]` class. Editable fields are reset when the slides are reloaded, but it is possible for edits to persist across sessions (in the same browser) by giving the editable element a `.key-<NAME>` class, where `<NAME>` is a unique identifier (and valid CSS class).

Usage

```
use_editable(id = NULL, expires = 14)

html_dependency_editable(expires = 14, id = NULL)
```

Arguments

`id`	Optional. By default, when `id` is `NULL`, each re-generation of your slides creates a new document ID. This way, values that were previously stored in the browser for an older version of your slides will not be loaded into a new version. If you are confident that the editable fields in your slides are not changing between versions, you can set the document ID so that newer versions of your slides will continue to load edited values from previous versions in the browser.
`expires`	Editable values that also have a `.key-KEYNAME` class are stored in the browser and automatically loaded when the slides are reloaded. These values are stored using cookies so that they can eventually expire and `expires` provides the number of days that those values should be stored before being released.

Value

An `htmltools::tagList()` with the editable dependencies, or an htmltools::htmlDependency.

Functions

- `use_editable`: Adds editable to your xaringan slides.
- `html_dependency_editable`: Returns an htmltools::htmlDependency with the tile view dependencies. Most users will want to use `use_editable()`.

Usage

To make your xaringan presentations *editable*, add the following code chunk to your slides' R Markdown file.

````
```{r xaringan-editable, echo=FALSE}
# Setup editable fields and only store values in the browser for one day
# (by default values expire in 2 weeks).
```
````

```
xaringanExtra::use_editable(expires = 1)
```

Then, to make a component of your slides editable, use the `.can-edit[]` class.

```
## .can-edit[You can edit this slide title]
```

Editable fields that only have the `.can-edit` class are reset whenever the slides are re-loaded in your browser. If you want to store the edited values and have them persist across browser sessions, give each editable field a `.key-<NAME>` class. Be sure to make each key unique and note that the key name must be a valid CSS class, i.e. it cannot contain spaces.

```
## .can-edit.key-firstSlideTitle[Change this title and then reload the page]
```

Warning Editable fields may not work well with slide continuations. If your full slide builds up over several slides, you can only edit the currently visible slide. If the field has a key, however, all editable elements with the same key class are updated when the slides are loaded. In other words, you can edit the title on the first slide of a multi-part slide and reload the page to have the title applied to subsequent slides.

Examples

```
use_editable()
```

`embed_xaringan` *Embed a xaringan presentation in a web page*

Description

Embed xaringan slides in any HTML web page, such as a blogdown page or an R Markdown website. The presentation is embedded in a responsive aspect ratio container for seamless integration with your web page. This feature works best when combined with `use_share_again()`, but `embed_xaringan()` can be used for any xaringan presentation.

Usage

```
embed_xaringan(
  url,
  ratio = "16:9",
  border = "2px solid currentColor",
  max_width = NULL,
  margin = "1em auto",
  style = NULL
)
```

Arguments

`url`	The URL or path to the presentation to embed.
`ratio`	The ratio of the presentation, either as `"width:height"` or `width/height`, e.g. `"16:9"` or `1.7777`.
`border`	The border style of the embedded `<iframe>`. For no border, use `"none"`.
`max_width`	The max width of the `<iframe>`, in a valid CSS units.
`margin`	The margin placed around the embedded `<iframe>`.
`style`	Additional CSS `style` property value pairs, e.g. `c("padding-left: 1em", "padding-right: 1em")`.

Value

An `htmltools::tagList()` with the HTML dependencies required for **share again**.

See Also

`use_share_again()`

Examples

```
# In your slides call
use_share_again()
```

`extra_styles` *Add Extra CSS Styles*

Description

Adds CSS extras to your slides. You can select which extras you wish to add to your slides.

Usage

```
use_extra_styles(
  hover_code_line = TRUE,
  mute_unhighlighted_code = TRUE,
  bundle_id = NULL
)

html_dependency_extra_styles(
  hover_code_line = TRUE,
  mute_unhighlighted_code = TRUE,
  bundle_id = NULL
)
```

Arguments

`hover_code_line`
: Adds a hover effect for code chunks in your slides. Adds a floating pointer to the hovered line and makes the line bold.

`mute_unhighlighted_code`
: On code chunks with highlights (added with line-ending #<< comments or starting with *), non-highlighted lines are muted and the highlighted line is full opacity.

`bundle_id`
: Make the CSS bundle unique. Use this if your slides share a common resource directory and you want to include different CSS extras in different slides.

Value

An `htmltools::htmlDependency()` with the selected additional styles.

Functions

- `use_extra_styles`: Add the extra CSS styles to your slides
- `html_dependency_extra_styles`: Returns an `htmltools::htmlDependency()` with the extra styles dependencies. Most users will want to use `use_extra_styles()`.

Examples

```
use_extra_styles()
```

`fit_screen` *Fit Slides to the Screen*

Description

This extension resizes the slides to match the browser window height and width. In other words, the slides are maximized to match the screen size. The primary use case for this extension is for when you want to show your slides in split screen, for example when demonstrating code in RStudio or another window. To enable fit-to-screen, press **Alt/Option + F** during the slideshow. To disable, reload the slides.

Usage

```
use_fit_screen()

html_dependency_fit_screen()
```

Value

An`htmltools::tagList()` with the fit-to-screen dependency, or an `htmltools::htmlDependency()`.

Functions

- `use_fit_screen`: Use the fit-to-screen extension in your xaringan slides.
- `html_dependency_fit_screen`: Returns an `htmltools::htmlDependency()` with the fit screen dependencies. Most users will want to use `use_fit_screen()`.

Usage

To enable fit-to-screen, add the following code chunk to your slides:

````
```{r xaringan-fit-screen, echo=FALSE}
xaringanExtra::use_fit_screen()
```
````

And then press **Alt/Option + F** at any point during your slide show to enable the extension.

Examples

```
use_fit_screen()
```

---

freezeframe *FreezeFrame*

Description

FreezeFrame starts any gifs on a slide when you turn to that slide. This helps This helps alleviate the awkward pause that can happen when you turn to a slide with a gif that has already started and you have to wait until it loops back around. You can also directly click on the gif to stop or start it.

Usage

```
use_freezeframe(
  selector = "img[src$=\"gif\"]",
  trigger = c("click", "hover", "none"),
  overlay = FALSE,
  responsive = TRUE,
  warnings = TRUE
)

html_dependency_freezeframe()
```

Arguments

selector	The selector used to search for .gifs to freeze.
trigger	The trigger event to start animation for non-touch devices. One of "click" (default), "hover" or "none".
overlay	Whether or not to display a play icon on top of the paused image, default: FALSE.
responsive	Whether or not to make the image responsive (100% width), default: TRUE.
warnings	Whether or not to issue warnings in the browser console if an image doesn't appear to be a gif.

Value

An htmltools::tagList() with the FreezeFrame dependencies, or an htmltools::htmlDependency().

Functions

- use_freezeframe: Adds FreezeFrame to your xaringan slides.
- html_dependency_freezeframe: Returns an htmltools::htmlDependency() with the FreezeFrame dependencies for use in xaringan and R Markdown documents. Most users will want to use use_freezeframe() instead.

Usage

To add FreezeFrame to your xaringan presentation, add the following code chunk to your slides' R Markdown file.

```
```{r xaringanExtra-freezeframe, echo=FALSE}
xaringanExtra::use_freezeframe()
```
```

References

http://ctrl-freaks.github.io/freezeframe.js/, https://github.com/ctrl-freaks/freezeframe.js/

Examples

```
use_freezeframe()
```

logo *Add Logo*

Description

use_logo() adds a logo to all of your slides. You can make the logo a clickable link and choose where on the page it is placed. You can also set which types of slides will not get the logo by default.

Usage

```
use_logo(
  image_url,
  width = "110px",
  height = "128px",
  position = css_position(top = "1em", right = "1em"),
  link_url = NULL,
  exclude_class = c("title-slide", "inverse", "hide_logo")
)

html_dependency_logo(
  link_url = NULL,
  exclude_class = c("title-slide", "inverse", "hide_logo"),
  inline = NULL
)
```

Arguments

image_url	The URL to the image file of your logo. In general, either a full URL or the path to the image relative to your slides file.
width	Width in CSS units of the logo
height	Height in CSS units of the logo
position	The position of the logo from the sides of the slide. Use css_position() to specify.
link_url	Optional. If provided, your logo becomes a clickable link.
exclude_class	The slide classes that should not receive the logo. By default, the title slide, inverse slides and slides with the hide_logo class are excluded.
inline	In html_dependency_logo(), should the JS and CSS code to create the logo be returned inline (inside the rendered slides) or in separate CSS and JS documents attached as an html dependency? The default is to use inline for **xaringan** version 0.16 or later.

Value

An htmltools::tagList() with the Add Logo dependencies, or an htmltools::htmlDependency().

Functions

- use_logo: Adds logo to your xaringan slides.
- html_dependency_logo: Returns an htmltools::htmlDependency() with the tile view dependencies. Most users will want to use use_logo().

Usage

To add a logo to your xaringan presentation, add the following code chunk to your slides' R Markdown file.

```
```{r xaringan-logo, echo=FALSE}
xaringanExtra::use_logo(
  image_url = "https://raw.githubusercontent.com/rstudio/hex-stickers/master/PNG/xaringan.png"
)
```
```

See the documentation for ?use_logo for more options regarding sizing and positioning. You can also make the logo a link using link_url and you can hide the logo for a particular slide by using the hide_logo slide class.

Examples

```
xaringan_logo <- file.path(
  "https://raw.githubusercontent.com/rstudio/hex-stickers/master",
  "PNG/xaringan.png"
)
use_logo(xaringan_logo)
```

---

panelset *Panelset*

Description

A panelset designed for showing off code, but useful for anything really.

Usage

```
use_panelset(in_xaringan = NULL)

style_panelset_tabs(
  foreground = NULL,
  background = NULL,
  ...,
  active_foreground = NULL,
  active_background = NULL,
  active_border_color = NULL,
```

```
  hover_background = NULL,
  hover_foreground = NULL,
  hover_border_color = NULL,
  tabs_border_bottom = NULL,
  tabs_sideways_max_width = NULL,
  inactive_opacity = NULL,
  font_family = NULL,
  selector = ".panelset"
)

style_panelset(...)

html_dependency_panelset()
```

Arguments

`in_xaringan`	Set to `TRUE` if rendering in xaringan slides or `FALSE` if using panelset elsewhere. This determines the style of knitr hook that is registered to enable the `panelset` chunk option.
`foreground`	The text color of a non-active panel tab, default is `currentColor`.
`background, active_background, hover_background`	Background colors for panel tabs; in-active tabs, active tab, hovered tab. The default values are all `unset`.
`...`	Ignored or passed from `style_panelset()` to `style_panelset_tabs()`.
`active_foreground`	The text color of an active, as in selected, panel tab. Default is `currentColor`.
`active_border_color, hover_border_color`	The color of the top border of a tab when it is active or the color of the bottom border of a tab when it is hovered or focused. Defaults are `currentColor`.
`hover_foreground`	The text color of a hovered panel tab. Default is `currentColor`.
`tabs_border_bottom`	The border color between the tabs and content. Default is `#ddd`.
`tabs_sideways_max_width`	The maximum width of the tabs in sideways mode. The default value is `25%`. A value between 25% and 33% is recommended. The tabs can only ever be at most 50% of the container width.
`inactive_opacity`	The opacity of inactive panel tabs, default is `0.5`.
`font_family`	The font family to be used for the panel tabs text. Default is a monospace system font stack.
`selector`	The CSS selector used to choose which panelset is being styled. In most cases, you can use the default selector and style all panelsets on the page.

Value

An `htmltools::tagList()` with the panelset dependencies, or an `htmltools::htmlDependency()`.

Functions

- `use_panelset`: Adds panelset to your xaringan slides.
- `style_panelset_tabs`: Style the panelset. Returns an **htmltools** `<style>` tag.
- `style_panelset`: Deprecated, renamed `style_panelset_tabs()`.
- `html_dependency_panelset`: Returns an `htmltools::htmlDependency()` with the tile view dependencies. Most users will want to use `use_panelset()`.

Usage

To add panelset to your xaringan presentation, add the following code chunk to your slides' R Markdown file.

````
```{r xaringan-panelset, echo=FALSE}
xaringanExtra::use_panelset()
```

.panelset[
.panel[.panel-name[app.R]

```r
hist(runif(100))
```
]

.panel[.panel-name[About]

Take a look at the R code in that other panel.
]
]
````

Sideways Panelsets

As an alternative to the "tabs above content" view, you can also use *sideways* panelsets where the tabs appear beside the tabbed content.

To choose this effect, add the `.sideways` class to `.panelset` in your slides or R Markdown text.

````
.panelset.sideways[
.panel[.panel-name[ui.R]
```r
# shiny ui code here...
```
]

.panel[.panel-name[server.R]
```r
function(input, output, session) {
  # shiny server code here...
````
````

```
}
```
]
]

By default in sideways-mode, the tabs will appear on the left side. You can choose to place the tabs on the right side by including both `.sideways` and `.right` with `.panelset`.

.panelset.sideways.right[
.panel[.panel-name[ui.R]
```r
# shiny ui code here...
```
]

.panel[.panel-name[server.R]
```r
function(input, output, session) {
  # shiny server code here...
}
```
]
]

Style Panelset

To customize the appearance of your panels, you can use `style_panelset_tabs()` called directly in an R chunk in your slides.

```{r echo=FALSE}
style_panelset_tabs(foreground = "honeydew", background = "seagreen")
```

The panelset uses opacity to soften the in-active tabs to increase the chances that the tabs will work with your slide theme. If you decide to change your tab colors or to use solid colored tabs, you'll likely want to set `inactive_opacity = 1` in `style_panelset()` (or the corresponding `--panel-tab-inactive-opacity` CSS variable).

Behind the scenes, `style_panelset_tabs()` updates the values of custom CSS properties that define the panelset appearance. If you'd rather work with CSS, the default values of these properties are shown in the CSS code below. You can copy the whole CSS block to your slides and modify the values to customize the style to fit your presentation.

```{css echo=FALSE}
.panelset {
   --panel-tab-foreground: currentColor;
   --panel-tab-background: unset;
   --panel-tab-active-foreground: currentColor;
   --panel-tab-active-background: unset;
```

```
  --panel-tab-active-border-color: currentColor;
  --panel-tab-hover-foreground: currentColor;
  --panel-tab-hover-background: unset;
  --panel-tab-hover-border-color: currentColor;
  --panel-tab-inactive-opacity: 0.5;
  --panel-tabs-border-bottom: #ddd;
 --panel-tab-font-family: Menlo, Consolas, Monaco, Liberation Mono, Lucida Console, monospace;
}
```

Usage in R Markdown

Panelset works in all R Markdown HTML outputs like HTML reports and blogdown webpages!

Panelset works in the same way as `rmarkdown`'s tabset feature, albeit with fewer style options, but the trade-off is that it works in a wider range of document types. Generally, as long as the output is HTML, panelset should work.

Another advantage of panelset is that it enables deeplinking: the currently shown tab is encoded in the URL automatically, allowing users to link to open tabs. Users can also right click on a panel's tab and select *Copy Link* to link directly to a specific panel's tab, which will appear in view when visiting the copied link.

With standard R Markdown, i.e. `rmarkdown::html_document()`, you can use the following template.

```
# Panelset In R Markdown! {.panelset}

## Tab One

Amet enim aptent molestie vulputate pharetra
vulputate primis et vivamus semper.

## Tab Two

### Sub heading one

Sit etiam malesuada arcu fusce ullamcorper
interdum proin tincidunt curabitur felis?

## Tab Three

Adipiscing mauris egestas vitae pretium
ad dignissim dictumst platea!

# Another section

This content won't appear in a panel.
```

In other, less-standard R Markdown HTML formats, you can use pandoc's fenced divs.

```
::::: {.panelset}

::: {.panel}
[First Tab]{.panel-name}

Lorem sed non habitasse nulla donec egestas magna
enim posuere fames ante diam!
:::

::: {.panel}
[Second Tab]{.panel-name}

Consectetur ligula taciti neque scelerisque gravida
class pharetra netus lobortis quisque mollis iaculis.
:::

:::::
```

Alternatively, you can also use raw HTML.

```
<div class="panelset">
  <div class="panel">
    <div class="panel-name">First Tab</div>
    <!-- Panel content -->
    <p>Lorem ipsum, etc, etc</p>
  </div>
  <div class="panel">
    <div class="panel-name">Second Tab</div>
    <!-- Panel content -->
    <p>Lorem ipsum, etc, etc</p>
  </div>
</div>
```

Panelset knitr Chunks

A common use-case for **panelset** is to show the code and its output in separate tabs. For example, you might want to first show the code to create a plot in the first tab, with the plot itself in a second tab. On slides where space is constrained, this approach can be useful.

To help facilitate this process, **panelset** provides a `panelset` chunk option. When set to `TRUE`, the code is included in a panel tab named *Code* and the output is included in a panel tab named *Output*. Note that you still need to wrap this chunk in a panelset-creating container.

```
.panelset[
```{r panelset = TRUE}
list(
  normal = rnorm(10),
  uniform = runif(10),
  cauchy = rcauchy(10)
```
```

````
)
```
]
````

You can also set the `panelset` chunk option to a named vector, where the `source` item is the tab name for the source code and the `output` item is the tab name for the code output.

````
```{r panelset = c(source = "ggplot2", output = "Plot")}
ggplot(Orange) +
  aes(x = age, y = circumference, colour = Tree) +
  geom_point() +
  geom_line() +
  guides(colour = FALSE) +
  theme_bw()
```
````

When your code contains multiple expressions and outputs, you may also want to set the `results = "hold"` chunk option. Currently, knitr uses `results = "markup"` as the default, in which case each code expression and output pair will generate a pair of tabs.

````
```{r panelset = TRUE, results="hold"}
print("Oak is strong and also gives shade.")
print("The lake sparkled in the red hot sun.")
```
````

Finally, panelset chunks also work in R Markdown documents, but they must be encapsulated in `<div class="panelset">` and `</div>`

````
<div class="panelset">

```{r panelset = TRUE}
print("Oak is strong and also gives shade.")
```

</div>
````

or appear inside a section with the panelset class.

````
### A Random Sentence {.panelset}

```{r panelset = TRUE}
print("Oak is strong and also gives shade.")
```
````

[
````

```
list(
  normal = rnorm(10),
  uniform = runif(10),
  cauchy = rcauchy(10)
)
```

]: R:%0A%60%60%60%7Br%20panelset%20=%20TRUE%7D%0Alist(%0A%20%20normal%20=%20rnorm(10),%0A%2

Examples

```
use_panelset()
```

`scribble` *Scribble*

Description

Scribble lets you draw on your **xaringan** slides. Click the pencil icon to begin drawing. Use the eraser to remove lines from your drawing, or the trash to clear the entire canvas. Note that in order to minimize confusion, you will not be able to navigate slides while in draw or erase mode.

You may toggle the visibility of the scribble toolbox by pressing `S` at any time. Your drawings will persist when changing slides. You may save a permanent copy of the slides with the markup by printing your presentation (e.g. using Chrome > File > Print).

Usage

```
use_scribble(
  pen_color = "#FF0000",
  pen_size = 3,
  eraser_size = pen_size * 10,
  palette = NULL
)

html_dependency_fabricjs(minimized = TRUE)

html_dependency_scribble(pen_color, pen_size, eraser_size, palette = NULL)
```

Arguments

`pen_color`	Initial pen color (default is `"#FF0000` (red)). Must be a hexadecimal color, e.g. `#000` or `#4232ea`.
`pen_size`	Pen size (default is 3).
`eraser_size`	Eraser size (default is `pen_size * 10`).
`palette`	A selection of up to 10 colors that become available when drawing is active via the keys `0` through `9`. Press the number keys of 0-9 to quickly active each of the palette colors.
`minimized`	Use the minimized `fabric.js` dependency?

Value

An `htmltools::tagList()` with the scribble dependencies, or an `htmltools::htmlDependency()`.

Functions

- `use_scribble`: Adds scribble to your xaringan slides.
- `html_dependency_fabricjs`: Returns an `htmltools::htmlDependency()` with the `fabric.js` dependencies for use in xaringan and R Markdown documents. Most users will want to use `use_scribble()` instead.
- `html_dependency_scribble`: Returns an `htmltools::htmlDependency()` with the scribble dependencies for use in xaringan and R Markdown documents. Most users will want to use `use_scribble()` instead.

Usage

To add scribble to your `xaringan` presentation, add the following code chunk to your slides' R Markdown file.

````
```{r xaringan-scribble, echo=FALSE}
xaringanExtra::use_scribble()
```
````

Examples

```
use_scribble()
```

---

search *Search*

Description

Search gives you a way to quickly search for text on slides.

Usage

```
use_search(
  position = c("bottom-left", "bottom-right", "top-left", "top-right"),
  case_sensitive = FALSE,
  show_icon = FALSE,
  auto_search = TRUE
)

html_dependency_search(
  position = c("bottom-left", "bottom-right", "top-left", "top-right"),
  case_sensitive = FALSE,
  show_icon = FALSE,
```

```
  auto_search = TRUE
)

style_search(
  icon_fill = "rgba(128, 128, 128, 0.5)",
  input_background = "rgb(204, 204, 204)",
  input_foreground = "black",
  input_border = "1px solid rgb(249, 38, 114)",
  match_background = "rgb(38, 220, 249)",
  match_foreground = "black",
  match_current_background = "rgb(38, 249, 68)",
  match_current_foreground = "black",
  selector = ".search"
)
```

Arguments

position	Where to place the search box.
case_sensitive	If FALSE, ignores case of search and text.
show_icon	Show the icon to open or close the search?
auto_search	Search on each keystroke (TRUE) or on enter (FALSE)?
icon_fill	Color of search icon
input_background	Color of search input box background
input_foreground	Color of text in search input box
input_border	Border style of search input box
match_background	Color of match background (not current)
match_foreground	Color of match text (not current)
match_current_background	Color of current match background
match_current_foreground	Color of current match text
selector	CSS selector specifying which search bar to update (for advanced or unusual uses only)

Value

An `htmltools::tagList()` with the search dependencies, or an `htmltools::htmlDependency()`.

Functions

- `use_search`: Adds search to your xaringan slides.
- `html_dependency_search`: Returns an `htmltools::htmlDependency()` with the search dependencies. Most users will want to use `use_search()`.
- `style_search`: Style the search input.

Usage

To add search to your xaringan presentation, add the following code chunk to your slides' R Markdown file.

```
```{r xaringan-search, echo=FALSE}
xaringanExtra::use_search()
```
```

Author(s)

Original implementation by André Restivo

References

https://github.com/arestivo/remark.search

Examples

```
use_search()
```

---

share_again *Share or Embed xaringan Slides*

Description

The *share again* extension helps you share your xaringan slides. Adding the extension to your slides with `use_share_again()` enables a "share" bar that only appears when your slides are embedded in a web page in an `<iframe>`. To embed your presentation in another page, like a blogdown post or a R Markdown based site, use `embed_xaringan()`. This function adds your slides in a responsive aspect ratio container that seamlessly includes your slides in the page. Finally, for perfect looking slides, use `style_share_again()` to style the share bar to match your slides' aesthetic. You can also use this function to enable or disable specific social media sites and platforms from the share menu.

Usage

```
use_share_again()

style_share_again(
  foreground = "rgb(255, 255, 255)",
  background = "rgba(0, 0, 0, 0.5)",
 share_buttons = c("all", "none", "twitter", "facebook", "linkedin", "pinterest",
    "pocket", "reddit")
)
```

Arguments

foreground	The foreground color of the buttons and text in the share bar
background	The background color of the share bar
share_buttons	A vector of social media platforms to be included in the share menu of the share bar. You can include "all" sites or "none" of buttons, and if either are included in share_buttons, then the rest of the vector is ignored. (And "all" takes precedence over "none".) The *copy link* option is always included.

Value

An htmltools::tagList() with the HTML dependencies required for **share again**.

Functions

- use_share_again: Add the *share again* bar to your slides (only shown when embedded in an <iframe>)
- style_share_again: Style the *share again* bar to match your slides, or choose the social media sites included in the share menu

See Also

embed_xaringan()

Examples

```
# In your slides call
use_share_again()

# In the document where you want to embed the slides call
embed_xaringan("https://slides.yihui.org/xaringan/")
```

---

slide_tone *Slide Tone*

Description

Slide tone plays a subtle sound when you change slides.The tones increase in pitch for each slide from a low C to a high C note. The tone pitch stays the same for incremental slides.

Usage

```
use_slide_tone()

html_dependency_slide_tone()
```

Value

An `htmltools::tagList()` with the slide tone dependencies, or an htmltools::htmlDependency.

Functions

- `use_slide_tone`: Adds slide tone to your xaringan slides.
- `html_dependency_slide_tone`: Returns an htmltools::htmlDependency with the tile view dependencies. Most users will want to use `use_slide_tone()`.

Usage

To add slide tone to your xaringan presentation, add the following code chunk to your slides' R Markdown file.

```
```{r xaringan-slide-tone, echo=FALSE}
xaringanExtra::use_slide_tone()
```
```

References

tone.js

Examples

```
use_slide_tone()
```

`style_banner` *Style Banner*

Description

Change the banner style properties for a specific selector. By default, the changes apply to all banners, but by setting `selector` you can apply style changes to the banners of specific slide styles. For example, to set the styles for inverse slides, use `selector = ".inverse"`.

Usage

```
style_banner(
  text_color = NULL,
  background_color = NULL,
  padding_horizontal = NULL,
  padding_vertical = NULL,
  height = NULL,
  width = NULL,
  font_size = NULL,
  font_family = NULL,
```

```
  z_index = NULL,
  selector = ":root"
)
```

Arguments

`text_color`	The color of text in the banners which may be overridden by other styles, e.g. link color, etc. The default value inherits from the primary text color of the slide.
`background_color`	The color of the banner background. By default the background is transparent.
`padding_horizontal, padding_vertical`	The inner padding of the banner. By default no padding is applied in the vertical direction, but `2em` of padding is applied horizontally. If anything, you probably only want to change the value of `padding_horizontal`.
`height`	The height of the banner in a valid CSS unit.
`width`	The maximum width of each column in the banner. You can set the width for all columns with a single valid CSS unit, or you may provide a vector of CSS units, named `"left"`, `"center"`, or `"right"` or provided in that order.
`font_size, font_family`	The font size and family of the text in the banner. The default font size is `0.7em` and the default family inherits from the primary text of the slide.
`z_index`	The z-index of the banner. By default this value is 0 so that all other content appears over the banner. To ensure the banner appears *above slide content*, you can set `z_index` to something greater than 0.
`selector`	A CSS selector, e.g. `".inverse"`, where the styles set in this call should be applied. Typically, you'll either set these styles for all banners using the default `selector`, or you'll want to customize the banner appearance for particular slide classes. Note that you can call `style_banner()` as many times as you want in your slides, but you'll want to change the `selector` for each call.

Value

Returns a `<style>` tag with the banner styles for `selector` as HTML via `htmltools::HTML()`.

See Also

`use_banner()`

Examples

```
style_banner(text_color = "red")
style_banner(text_color = "white", background_color = "red")
```

tachyons *Tachyons*

Description

Tachyons is a collection of CSS utility classes that works beautifully with **xaringan** presentations using the 'remarkjs" class syntax.

Usage

```
use_tachyons(minified = TRUE)

html_dependency_tachyons(minified = TRUE)
```

Arguments

`minified`	Use the minified Tachyons css file? Default is `TRUE`.

Value

An `htmltools::tagList()` with the tachyons dependencies, or an `htmltools::htmlDependency()`.

Functions

- `use_tachyons`: Adds tachyons to your xaringan slides.
- `html_dependency_tachyons`: Returns an `htmltools::htmlDependency()` with the tile view dependencies. Most users will want to use `use_tachyons()`.

Usage

To add tachyons to your xaringan presentation, add the following code chunk to your slides' R Markdown file.

````
```{r xaringan-tachyons, echo=FALSE}
xaringanExtra::use_tachyons()
```
````

Tachyons provides small, single-purpose CSS classes that are easily composed to achieve larger functionality and styles. In the remarkjs content classes syntax, you can compose classes by chaining them together. For example, the following markdown produces a box with a washed green background (`.bg-washed-green`), a dark green border (`.b--dark-green`) on all sides (`.ba`) with line width 2 (`.bw2`) and border radius (`.br3`). The box has a shadow (`.shadow-5`) and medium-large horizontal padding (`.ph4`) with a large top margin (`.mt5`).

```
.bg-washed-green.b--dark-green.ba.bw2.br3.shadow-5.ph4.mt5[
The only way to write good code is to write tons of bad code first.
Feeling shame about bad code stops you from getting to good code

.tr[
 Hadley Wickham
]]
```

References

tachyons, Tachyons Cheat Sheet

Examples

```
use_tachyons()
```

`tile_view` *Tile View*

Description

Tile view gives you a way to quickly jump between slides. Just press **O** at any point in your slideshow and the tile view appears. Click on a slide to jump to the slide, or press **O** to exit tile view.

Usage

```
use_tile_view()

html_dependency_tile_view()
```

Value

An `htmltools::tagList()` with the tile view dependencies, or an `htmltools::htmlDependency()`.

Functions

- `use_tile_view`: Adds tile view to your xaringan slides.
- `html_dependency_tile_view`: Returns an `htmltools::htmlDependency()` with the tile view dependencies. Most users will want to use `use_tile_view()`.

Usage

To add tile view to your xaringan presentation, add the following code chunk to your slides' R Markdown file.

```
```{r xaringan-tile-view, echo=FALSE}
xaringanExtra::use_tile_view()
```
```

Examples

```
use_tile_view()
```

---

use_banner *Add a banner to the top or bottom of your slides.*

Description

Adds a banner to all the slides in your deck at the top or the bottom of the slide. You can place content in the left, center, or right portion of the banner.

Usage

```
use_banner(
  ...,
  bottom_left = NULL,
  bottom_center = NULL,
  bottom_right = NULL,
  top_left = NULL,
  top_center = NULL,
  top_right = NULL,
  exclude = NULL
)
```

Arguments

`...`	Banner styles created with `style_banner()`. Technically, additional arguments are added into the `htmltools::tagList()` with the banner dependencies, but you'll only want to use `style_banner()` calls here.
`bottom_left, top_left`	Text or HTML to place in the left column of the top or bottom of the slide.
`bottom_center, top_center`	Text or HTML to place in the center column at the top or the bottom of the slide.
`bottom_right, top_right`	Text or HTML to place in the right column of the top or bottom of the slide.
`exclude`	A vector of slide classes where the banner should not be applied. By default all slides are included, but you might want to exclude the title and inverse slides with `exclude = c("title-slide", "inverse")`.

Value

An `htmltools::tagList()` with the banner dependencies, or an htmltools::htmlDependency.

See Also

`style_banner()`

Examples

```
use_banner(bottom_left = "bit.ly/my-awesome-slides")

use_banner(
  bottom_left = "bit.ly/my-awesome-slides",
  top_center = "My Presentation",
  exclude = c("title-slide", "inverse"),
  style_banner(text_color = "grey")
)
```

---

use_broadcast *Broadcast Your Slides*

Description

Experimental! **Broadcast** lets others follow along, in real time! Built with PeerJS, **broadcast** give you a unique URL to share with your viewers. Then, when they load your slides, their slides will automatically follow you as you present!

Usage

```
use_broadcast()
```

Details

To equip your slides with broadcast capabilities, add the following chunk to your slides' `.Rmd` file.

````
```{r broadcast, echo=FALSE}
xaringanExtra::use_broadcast()
```
````

Then, host your slides online, either on a personal webpage, or through Netlify, GitHub Pages, GitLab Pages, or another service.

When you want to present, open the version of your slides hosted online in a modern browser. Then press P to enter the presenter view. Click on the **Broadcast** button to start broadcasting.

After a short moment, if everything works, the broadcast button will turn into a broadcast link. Share this link with your audience. When they open the link, their browser will connect with yours and from then on, whenever you advance or change slides, your viewer's slides will move to the current slide.

Note that the broadcast link is unique and, as the presenter, is remembered for 4 hours. After 4 hours of inactivity, a new link will be generated. In general, create and share the broadcast link just before or as your event starts and certainly not more than an hour before the presentation.

How It Works:

PeerJS creates a direct, peer-to-peer connection between your browser and your viewer's browsers. A third party PeerJS server is used to initially facilitate the connection using the broadcast ID to connect with the presenter's browser.

After the connection is made, data is sent directly between browsers and the PeerJS server is no longer involved. Furthermore, at no time is any information about your presentation transmitted over the network. When you move to a slide, say for example slide 11, **broadcast** announces "Slide 11" to any connected viewers and JavaScript in their browser moves their presentation to slide 11.

This has two consequences:

1. Viewers can move around and look at slides other than the one currently active in the presenter's browser. When the presenter changes slides, however, all viewers' slides will move to the new slide.
2. If your slides involve interactivity, such as htmlwidgets or panelset, changes made in the presenter's view aren't replicated for viewers. Viewers will be taken to the same slide as the presenter, but they will need to click on their own to follow interactively.

Extra Details:

It's worth mentioning a few details. First of all, the broadcaster needs to be connected first before viewers connect. If a viewer connects before the broadcaster starts (or restarts), they should reload the link to reconnect.

Similarly, if the broadcaster reloads their slides, viewers will also need to reload to reconnect. But once everyone is connected, a message will appear for the viewer to prompt them to reconnect.

If you are the presenter and you load the broadcast link, the broadcast will automatically reconnect and start broadcasting. If you want to view your slides without broadcasting, just load the plain URL for the slides without the `?broadcast=...` portion. From this view, you can restart the broadcast from the presenter view and if the broadcast ID is still valid that ID will be used. To reset the broadcast ID without waiting 4 hours, load your slides with `?broadcast=1` and new broadcast link will be created at the next broadcast.

Value

An `htmltools::tagList()` with the HTML dependencies required for **broadcast**.

References

https://peerjs.com

Examples

```
use_broadcast()
```

use_progress_bar *Add an animated progress bar*

Description

Adds an animated progress bar to all slides.

Usage

```
use_progress_bar(
  color = "red",
  location = c("top", "bottom"),
  height = "0.25em"
)
```

Arguments

color	A valid CSS color to be used as the color of the progress bar.
location	One of "top" or "bottom".
height	A valid CSS unit specifying the height of the progress bar.

Value

An htmltools::tagList() with the HTML dependencies required for **progress bar**.

Examples

```
xaringanExtra::use_progress_bar("red", "top", "0.25em")
```

use_xaringan_extra *Use xaringanExtra Extensions*

Description

Load multiple **xaringanExtra** extensions at once. All extensions can be loaded with this function.

Usage

```
use_xaringan_extra(
  include = c("tile_view", "animate_css", "tachyons", "panelset", "broadcast",
    "share_again", "scribble")
)
```

Arguments

include	Character vector of extensions to include. One or more of "tile_view", "editable", "share_again", "broadcast", "slide_tone", "animate_css", "panelset" "tachyons", "fit_screen", "webcam", "clipboard", "search", "scribble", "freezeframe".

Value

An htmltools::tagList() with the htmltools::htmlDependency()s for the requested extensions.

Examples

```
use_xaringan_extra(c("tile_view", "panelset"))
use_xaringan_extra(c("tile_view", "scribble", "share_again"))
```

---

webcam *Webcam*

Description

Add a live video of your webcam into your slides (in your own browser only). Useful when you are presenting via video conference to include your video, or when you are recording a class or lecture.

Usage

```
use_webcam(width = 200, height = 200, margin = "1em")

html_dependency_webcam(width = 200, height = 200, margin = "1em")
```

Arguments

width, height	Width and height of the video pane in absolute CSS units, i.e. as 200 or "200px".
margin	Margin around the video pane in CSS units.

Details

To add **webcam** to your xaringan presentation, add the following code chunk to your slides' R Markdown file.

````
```{r}
xaringanExtra::use_webcam()
```
````

Inside your slides, press **w** to turn the webcam on and off, or press **Shift + W** to move the video to the next corner. You can also drag and drop the video within the browser window.

Value

An `htmltools::tagList()` with the HTML dependencies required for **webcam**.

Functions

- `use_webcam`: Add the webcam extension to your slides
- `html_dependency_webcam`: Returns an `htmltools::htmlDependency()` with the webcam dependencies. Most users will want to use `use_webcam()`.

References

The webcam extension is based on the original webcam implementation by Yihui Xie, author of **xaringan**.

Examples

```
use_webcam()
```

Index