bslib

Lecture 20

Dr. Colin Rundel

Shiny & bootstrap

The interface provided by Shiny is based on the html elements, styling, and javascript provided by the Bootstrap library.

As we've seen so far, knowing the specifics of Bootstrap are not needed for working with Shiny - but understanding some of its conventions goes a long way to helping you customize the elements of your app (via custom CSS and other components).

This is not the only place that Bootstrap shows up in the R ecosystem - e.g. both RMarkdown and Quarto html documents use Bootstrap for styling as well.

bslib

The bslib R package provides a modern UI toolkit for Shiny, R Markdown, and Quarto based on Bootstrap.

It facilitates:

- Custom theming of Shiny apps and R Markdown documents.
  - Apps can even be themed interactively in real-time.
- Use of modern versions of Bootstrap and Bootswatch
  - Shiny and R Markdown currently default to Bootstrap 3 and may continue to do so to maintain backwards compatibility.
- **Creation of delightful and customizable Shiny dashboards**
  - The underlying UI components (e.g., cards, value boxes, sidebars, etc) are also designed to work in other contexts (e.g., in R Markdown).

bslib components

Cards

> Cards are a common organizing unit for modern user interfaces (UI). At their core, they're just rectangular containers with borders and padding. However, when utilized properly to group related information, they help users better digest, engage, and navigate through content. This is why most successful dashboard/UI frameworks make cards a core feature of their component library.

```
card(
  card_header(
    "A header"
  ),
  card_body(
    shiny::markdown(
      "Some text with a [link](https://github.co
    )
  )
)
```

More options

```
card(
  max_height = 225,
  card_header(
    "A long, scrolling, description",
    class = "bg-dark"
  ),
  card_body(
    lorem::ipsum(paragraphs = 3, sentences = 5)
  )
)
```

```
card(
  max_height = 225,
  card_header(
    "A leaflet map",
    class = "bg-success"
  ),
  card_body(
    class = "p-0",
    leaflet::leaflet() |>
      leaflet::addTiles()
  )
)
```

Multiple card bodies

```
card(
  max_height = 500,
  card_header(
    "A long, scrolling, description",
    class = "bg-dark"
  ),
  card_body(
    leaflet::leaflet() |>
      leaflet::addTiles()
  ),
  card_body(
    lorem::ipsum(paragraphs = 1, sentences = 3)
  )
)
```

Value boxes

These are simple cards that are designed to show simple numeric or text values.

```
library(bsicons)
library(htmltools)

value_box(
  title = "I got",
  value = "99 problems",
  showcase = bs_icon("music-note-beamed"),
  theme = "cyan",
  p("bslib ain't one", bs_icon("emoji-smile")),
  p("hit me", bs_icon("suit-spade"))
)
```

Multiple value boxes

```
library(bsicons)
library(htmltools)

page_fillable(
  value_box(
    title = "1st value",
    value = "123",
    theme = "",
    showcase = bs_icon("bar-chart"),
    p("The 1st detail")
  ),
  value_box(
    title = "2nd value",
    value = "456",
    showcase = bs_icon("graph-up"),
    theme = "danger",
    p("The 2nd detail"),
    p("The 3rd detail")
  )
)
```

Layouts

Fixed layout

```
library(leaflet)
page_fillable(
  card(
    max_height = 200,
    card_header("Card 1"),
    lorem::ipsum(1,3)
  ),
  card(
    max_height = 100,
    card_header("Card 2"),
    "This is it."
  ),
  card(
    max_height = 200,
    card_header("Card 3"),
    leaflet() |> addTiles()
  )
)
```

Column layout

```
library(leaflet)
page_fillable(
  layout_columns(
    height = 200,
    card(
      card_header("Card 1"),
      lorem::ipsum(1,3)
    ),
    card(
      card_header("Card 2"),
      "This is it."
    )
  ),
  layout_columns(
    height = 300,
    card(
      card_header("Card 3"),
      leaflet() |> addTiles()
    )
  )
)
```

Column layout

Column widths layout

```
library(leaflet)
page_fillable(
  layout_columns(
    col_widths = c(8, 4, -1, 10, -1),
    row_heights = c("200px", "300px"),
    card(
      card_header("Card 1"),
      lorem::ipsum(1,3)
    ),
    card(
      card_header("Card 2"),
      "This is it."
    ),
    card(
      card_header("Card 3"),
      leaflet() |> addTiles()
    )
  )
)
```

Column widths layout

Dynamic layouts

```
library(leaflet)
layout_column_wrap(
  width = 1/2,
  card(
    max_height = 250,
    card_header("Card 1"),
    lorem::ipsum(1,3)
  ),
  card(
    max_height = 250,
    card_header("Card 2"),
    "This is it."
  ),
  card(
    max_height = 250,
    card_header("Card 3"),
    leaflet() |> addTiles()
  )
) |>
  anim_width("100%", "33%")
```

Dynamic layouts

Dynamic layouts - responsive columns

```
library(leaflet)
layout_column_wrap(
  width = "200px",
  card(
    max_height = 250,
    card_header("Card 1"),
    lorem::ipsum(1,3)
  ),
  card(
    max_height = 250, fill=FALSE,
    card_header("Card 2"),
    "This is it."
  ),
  card(
    max_height = 250,
    card_header("Card 3"),
    leaflet() |> addTiles()
  )
) |>
  anim_width("100%", "33%")
```

Dynamic layouts - responsive columns

Nested Layouts

```
library(leaflet)
layout_column_wrap(
  width = 1/2,
  card(
    card_header("Card 1"),
    lorem::ipsum(1,3)
  ),
  layout_column_wrap(
    width = 1,
    heights_equal = "row",
    card(
      card_header("Card 2"),
      "This is it."
    ),
    card(
      card_header("Card 3"),
      leaflet() |> addTiles()
    )
  )
)
```

Nested Layouts

Theming

Bootswatch

Due to the ubiquity of Bootstrap a large amount of community effort has gone into developing custom themes - a large free collection of these are available at bootswatch.com/.

`bs_theme()`

Provides a high level interface to adjusting the theme for an entire Shiny app,

- Change bootstrap version via `version` argument
- Pick a bootswatch theme via `bootswatch` argument
- Adjust basic color palette (`bg`, `fg`, `primary`, `secondary`, etc.)
- Adjust fonts (`base_font`, `code_font`, `heading_font`, `font_scale`)
- and more

The object returned by `bs_theme()` can be passed to the `theme` argument of `fluidPage()` and similar page UI elements.

In a Shiny app dynamic theming can be enabled by including `bs_themer()` in the server function of your app.

thematic

> Simplified theming of ggplot2, lattice, and {base} R graphics. In addition to providing a centralized approach to styling R graphics, thematic also enables automatic styling of R plots in Shiny, R Markdown, and RStudio.

In the case of our flexdashboard (or other shiny app), all we need to do is to include a call to `thematic_shiny()` before the app is loaded.

- Using the value `"auto"` will attempt to resolve the `bg`, `fg`, `accent`, or `font` values at plot time.