Customization and branding

Your turn

To keep everyone on the same page, we'll play with `quarto-websites-exercise-site_2025-10`

andhs.co/quarto-websites-2025

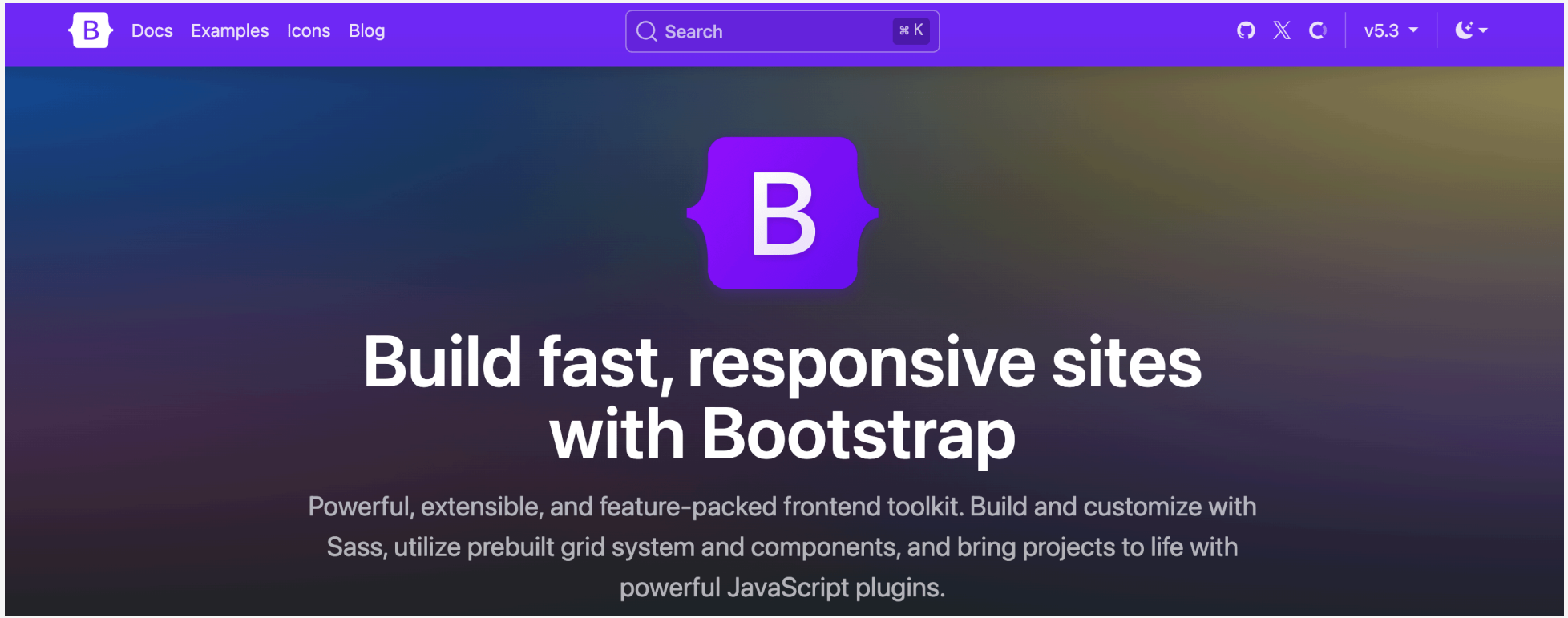
But keep your playground site from yesterday handy!

05:00

Themes

Bootstrap

Quarto uses the popular **Bootstrap** library for HTML structure and CSS styles.



Bootstrap

You can use **Bootstrap components and classes** for special styling

```
[Here's a button](thing.pdf){.btn .btn-primary role="button"}

[Here's another button](thing.pdf){.btn .btn-warning role="button"}
```

↓

Here's a button

Here's another button

Bootswatch themes

Quarto includes 25 themes from **Bootswatch**:

- default
- cerulean
- cosmo
- cyborg
- darkly
- flatly
- journal
- litera
- lumen
- lux
- materia
- minty
- morph
- pulse
- quartz
- sandstone
- simplex
- sketchy
- slate
- solar
- spacelab
- superhero
- united
- vapor
- yeti
- zephyr

Changing themes

Specify the custom theme under `theme` in the YAML settings:

_quarto.yml

```yaml
format:
  html:
    theme:
      - zephyr
```

Your turn

1. Go to **bootswatch.com** and explore the different themes there (use the top navigation bar).
2. Preview your site, then try changing different Bootswatch themes in `_quarto.yml`.

07:00

CSS and Sass

Theme options

Sometimes we want to change theme settings:

- Fonts
- Colors
- Aligment
- Spacing

Basic theme options

Quarto automatically supports **some common Bootstrap theme options**

```yaml
format:
  html:
    theme: zephyr
    fontsize: 1.2em
    linestretch: 1.2
    linkcolor: "#32a852"
    backgroundcolor: "#f8f8c4"
    mainfont: Comic Sans
```

Total control with CSS

Crash course in CSS

HTML elements can have IDs and classes:

```
<h2 id="my-section" class="special">A heading</h2>
```

- IDs are unique on the page
  - Shorthand: `#my-section`
- Classes aren't unique and can be repeated
  - Shorthand: `.special`

```
/* All H2s */
h2 {
  color: red;
}

/* All elements with id my-section */
/* This includes non H2s */
#my-section {
  color: red;
}

/* All H2s with id my-section */
h2#mysection {
  color: red;
}

/* All elements with class .special */
.special {
  font-style: italic;
}

/* All H2s with class .special */
h2.special {
  font-style: italic;
}
```

Nesting

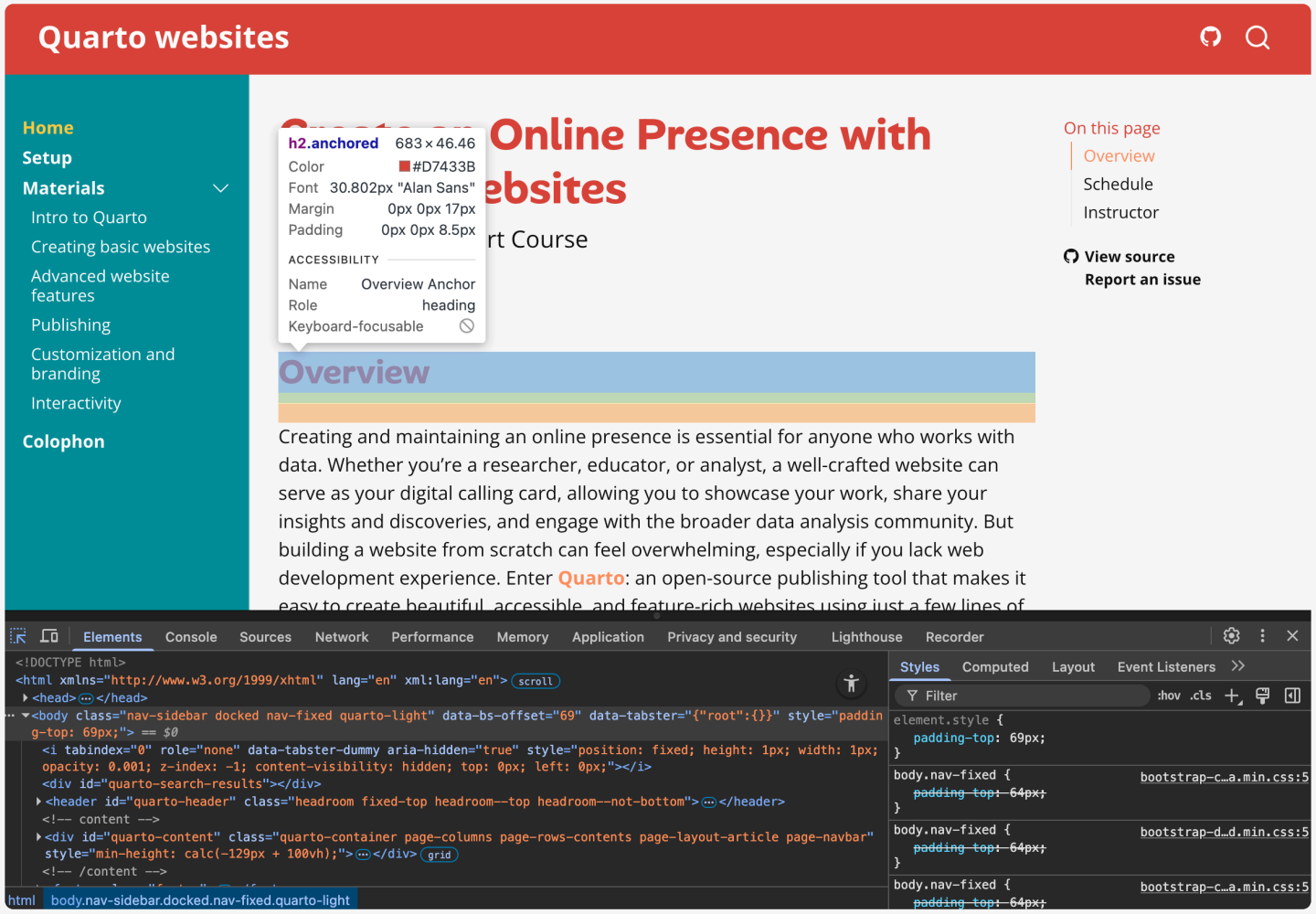
HTML elements can be nested inside each other

```html
<div id="part1">
  <h2 id="my-section" class="special">A heading</h2>
</div>

<div id="part2">
  <h2 id="blah" class="special">Another heading</h2>
</div>
```

```css
/* All H2s in #part1 */
#part1 h2 {
  color: red;
}

/* All H2s in #part2 */
#part2 h2 {
  color: blue;
}
```

Browser inpsector

Explore HTML and CSS right from your browser!



Browser inpsector

Edit and tinker with styles

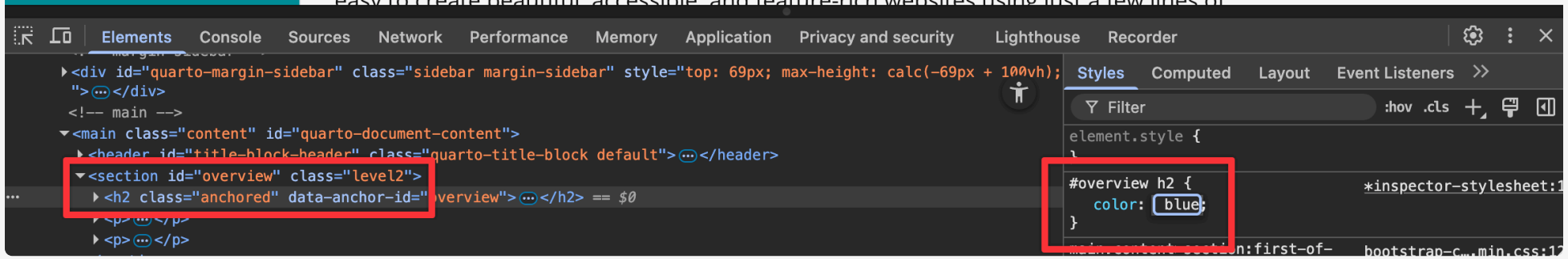
Customization and branding

Interactivity

Colophon

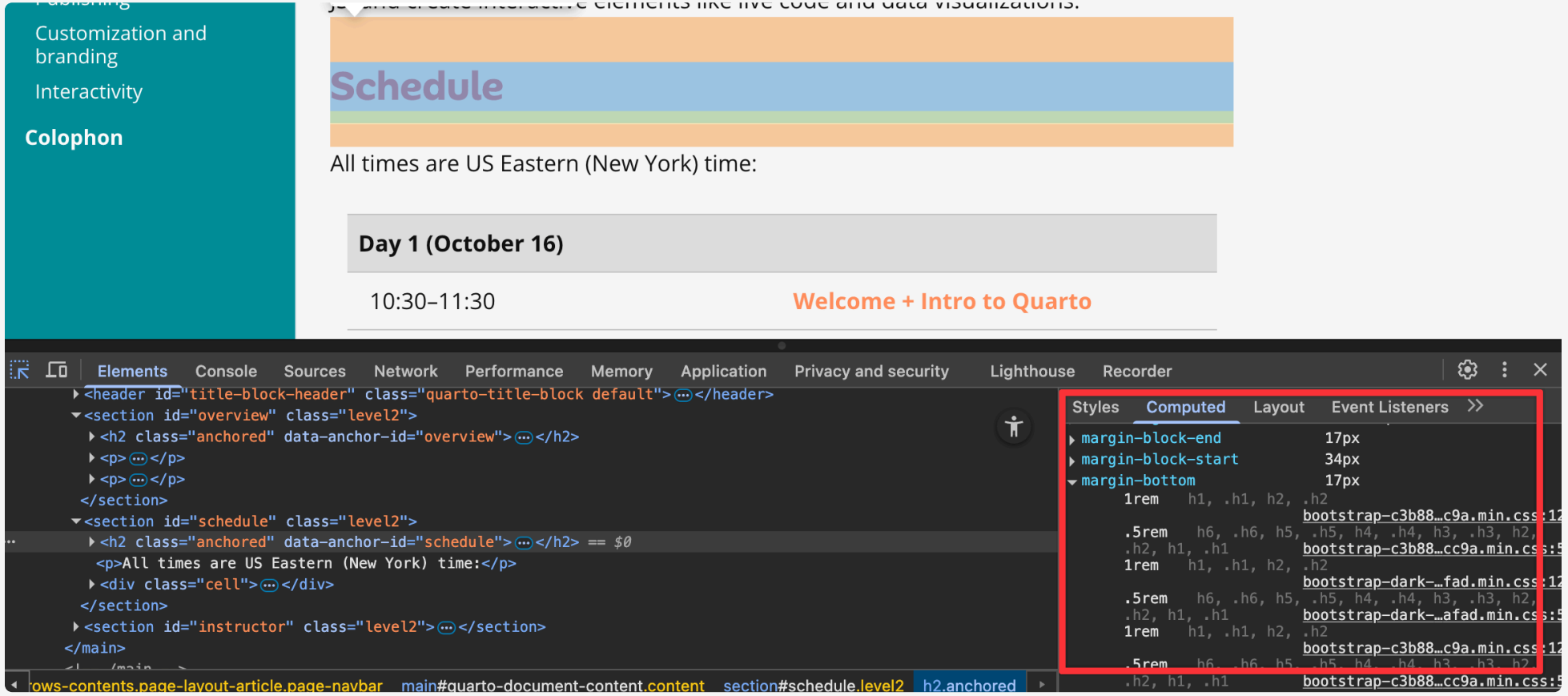
Overview

Creating and maintaining an online presence is essential for anyone who works with data. Whether you're a researcher, educator, or analyst, a well-crafted website can serve as your digital calling card, allowing you to showcase your work, share your insights and discoveries, and engage with the broader data analysis community. But building a website from scratch can feel overwhelming, especially if you lack web development experience. Enter Quarto: an open-source publishing tool that makes it easy to create beautiful, accessible, and feature-rich websites using just a few lines of



Browser inpsector

See which rule applies



Your turn

1. Preview your site
2. Use the browser inspector to select a heading
3. Create a CSS style that targets that heading and makes it
    - red (`color`) and
    - italic (`font-style`) and
    - 30px (`font-size`)
4. **Bonus**: Those in-browser edits will disappear if you refresh the page. See if you can figure out how to put them in your website. (**Hint!**)

05:00

Sass: CSS, but fancier

```
@import url('https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100..900;1,100..900&display=swap');


/*-- scss:defaults --*/

/* Built-in Bootstrap variables */
$h2-font-size:          1.6rem !default;
$headings-font-weight:  500 !default;
$font-family-sans-serif:  Roboto;

/* Your own variables */
$my-neat-red: #e21818;


/*-- scss:rules --*/
#quarto-sidebar h2 {
  font-family: Roboto;
  color: $my-neat-red;

  /* Magically compiles to #quarto-sidebar h2 a:hover */
  a:hover {
    color: #d1c81d;
  }
}
```

Bootstrap Sass variables

There are ***so many***!

- Quarto **lists these as the most typical**
- Bootstrap technically has ***all these***
- **Here's a filterable table** of everything

Using CSS/Sass with Quarto

Use vanilla CSS

```yaml
format:
  html:
    css: custom.css
```

Use Sass

```yaml
format:
  html:
    theme:
      - zephyr
      - custom.scss
```

Some examples:

- **This simple one**
- **This slightly more complex one**
- **This even more complex one**
- **heck yeah this one is wild**

Our turn

Create a SCSS file and make a bunch of rules!

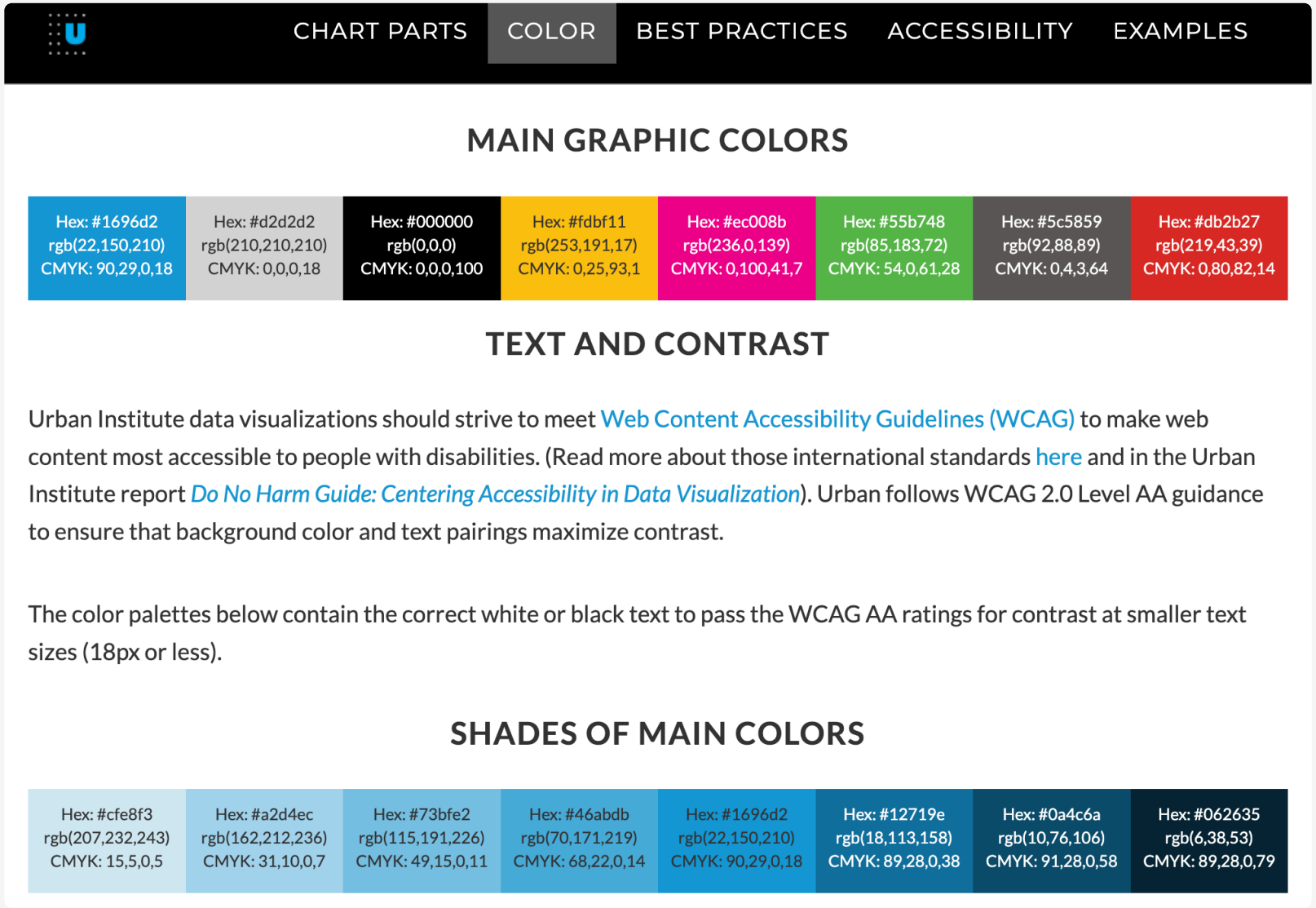
05:00

Branding

CSS is a little inconvenient

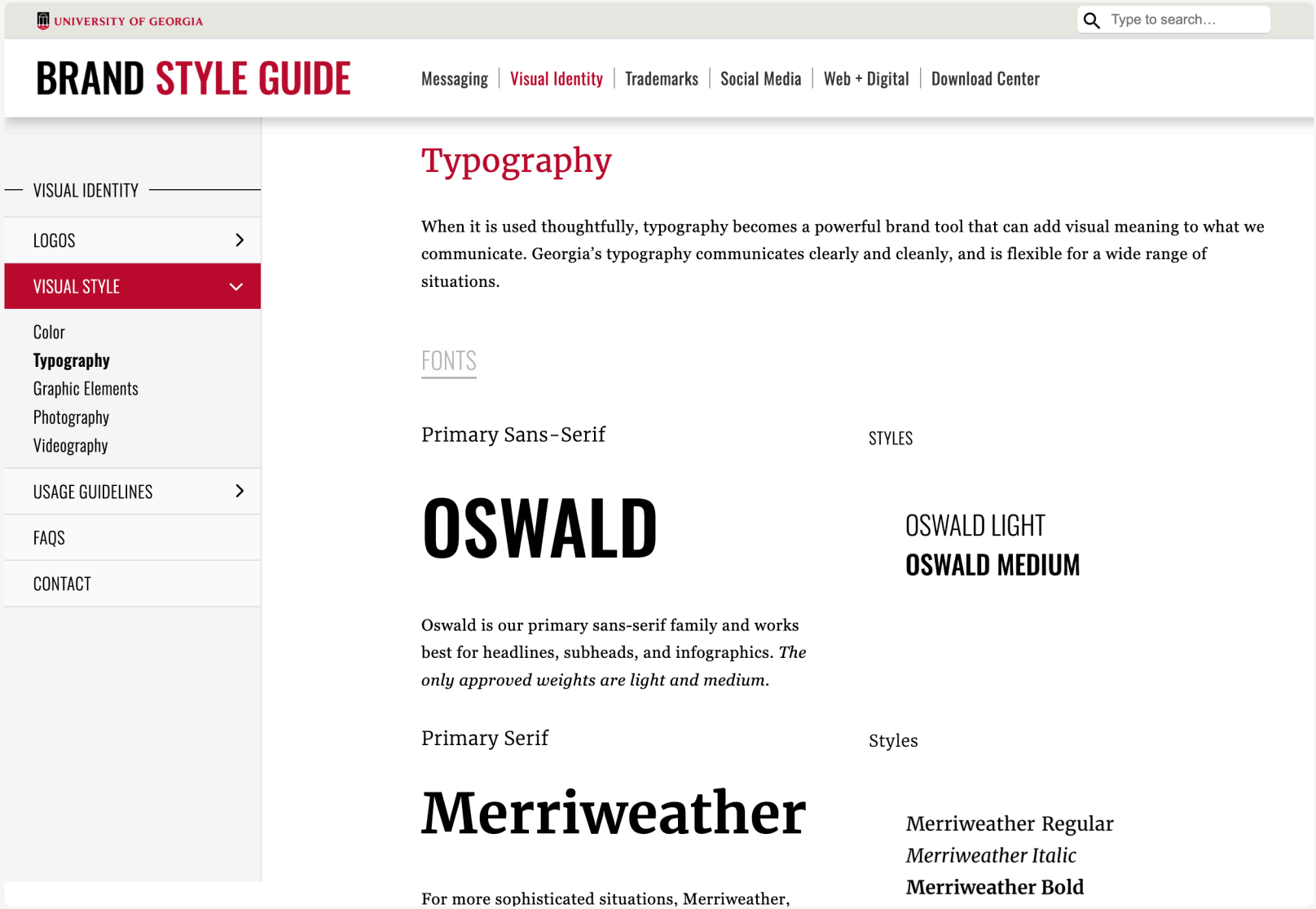
- HTML, slides, and dashboards all use slighlty different underlying HTML
- No easy way to reuse the colors and typography from your CSS customizations in R and Python plots or in PDF documents
- Hard to share consistent, resusable themes with others in your organization (or with the world)

Style guides



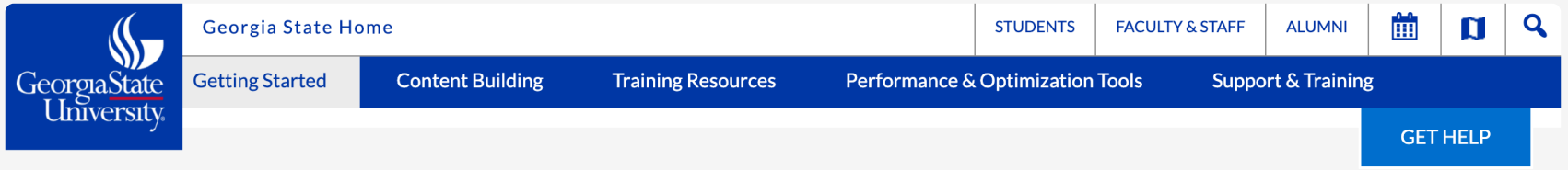
Urban Institute Data Visualization Style Guide

Style guides



University of Georgia Brand Style Guide

Style guides



Primary Colors

The university primary colors are blue and white. These two colors should be the strongest palette on layouts created for Georgia State.

Georgia State Blue	rgb **0 57 166** hex **0039A6**	**White**	rgb **255 255 255** hex **FFFFFF**

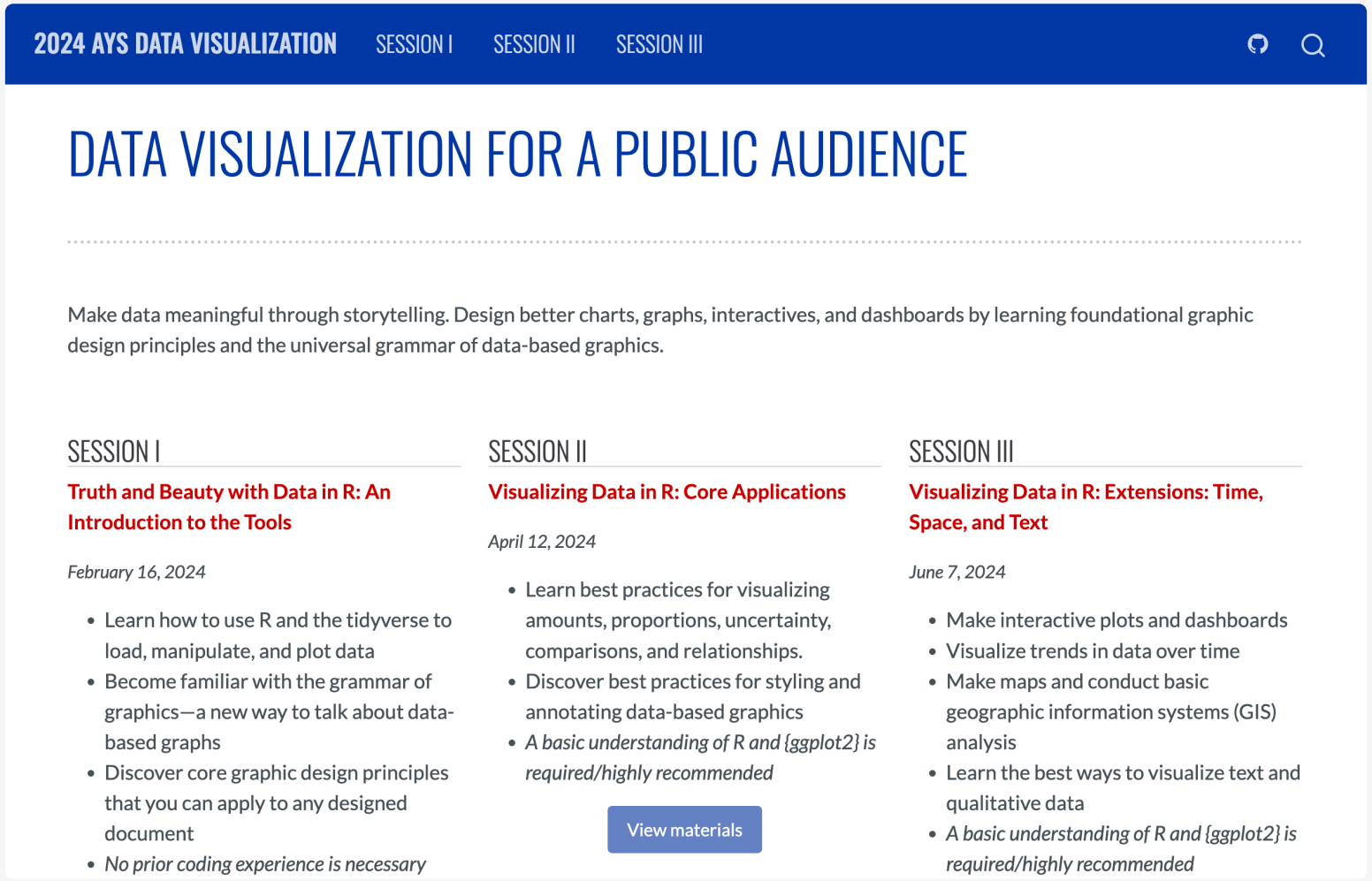
Secondary Color Chart

Red Accent	rgb **198 12 48** hex **CC0000**	**Light Blue**	rgb **151 202 235** hex **97CAEB**
Blue Steel	rgb **55 64 87** hex **374057**	**Light Gray**	rgb **238 238 238** hex **EEEEEE**
Cool Blue	rgb **0 113 206** hex **0071CE**	**Medium Gray**	rgb **204 204 204** hex **CCCCCC**

Georgia State University Web Color Guidelines

Matching style with CSS

You can recreate styles with CSS (site; custom.scss)



BRAND YML

Unified branding with a simple YAML file

Create reports, apps, dashboards, plots and more that match your organization's brand guidelines with a single `_brand.yml` file.

Learn more: **https://posit-dev.github.io/brand-yml/**

1. Define branding in a single `_brand.yml` file.
2. Apply that branding across almost all Quarto formats.

`_brand.yml` elements

- `meta`: Identifying information, name of the company, URLs, etc.
- `logo`: Files or links to the brand's logos
- `color`: Colors in the brand's color palette
- `typography`: Fonts for different elements
- `defaults`: Additional context-specific settings

_brand.yml structure

_brand.yml

```yaml
meta:
  name: Urban Institute
  link:
    home: https://urbaninstitute.github.io/graphics-styleguide/

logo:
  images:
    icon:
      path: img/urban.png
      alt: Urban Institute logo
  small: img/urban-sm.png

color:
  palette:
    blue: "#1696d2"
    gray-light: "#d2d2d2"
    black: "#000000"
    yellow: "#fdbf11"
```

Enabling `_brand.yml` in Quarto

1. Define branding in `_brand.yml`.
2. Save in the root directory of your Quarto project.

Quarto will detect the presence of `_brand.yml` and automatically apply the brand to all documents of the supported formats in the project.

If your brand file has a different name or lives in a subdirectory, use the `brand` key.

my-document.qmd

```yaml
---
title: "My neat report"
format: html
brand: org_theme.yml
---
```

Disable `_brand.yml`

To turn off `_brand.yml` for a document, use `brand: false`.

my-document.qmd

```yaml
---
title: "My neat report"
format: html
brand: false
---
```

Your turn

1. There is a file named `urban_institute.yml`. Make a copy of it, rename it to `_brand.yml`, and rerender your site. What changes?
2. Modify some of the variables in the `_brand.yml` file and rerender to see how your site changes. Explore **the brand.yml documentation** and see what other settings you can adjust.

   You can reset the file by copying and pasting it from **here**.

05:00

Use `_brand.yml` in Markdown

Access some `_brand.yml` values in Quarto documents with a shortcode

my_document.qmd

```
1 The site's main color is {{< brand color primary >}}
```

↓

The site’s main color is #d7433b

Learn more: **https://quarto.org/docs/authoring/brand.html#using-brand-values**

Use `_brand.yml` in Sass

Access some `_brand.yml` values with the `$brand-*` prefix

custom-styles.scss

```scss
/*-- scss:rules --*/

h3 {
  color: $brand-magenta;
}
```

Learn more: **https://quarto.org/docs/authoring/brand.html#in-scss**

Use `_brand.yml` in R and Python

Access and apply specific brand elements

R | Python

```r
library(brand.yml)

brand <- read_brand_yml("_brand.yml")

brand$color$primary
#> [1] "#1696d2"

brand$color$palette$yellow
#> [1] "#fdbf11"

brand$typography$fonts[[1]]$family
#> [1] "Lato"
```

Theme helpers

The {quarto} package contains theme helpers that apply branding to plots

R | Python

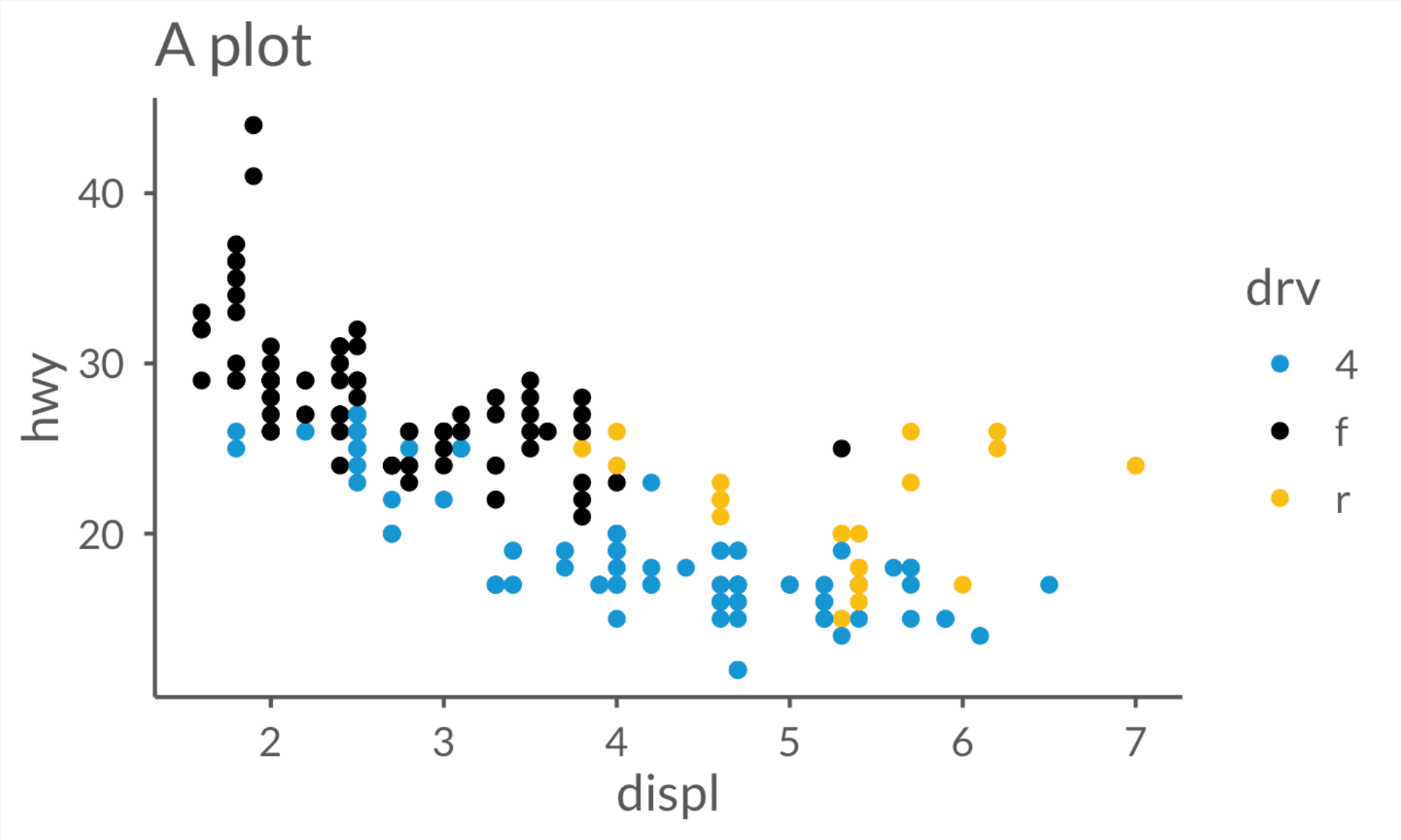
```r
library(quarto)

my_theme <- theme_brand_ggplot2("_brand.yml")
```

Doesn't do much (yet) beyond using the brand's `background` and `foreground` colors in the plot

R + {ggplot2} and Python + plotnine documentation

Example plot

```
library(tidyverse)
library(brand.yml)

brand <- read_brand_yml("_brand.yml")

brand_theme <- quarto::theme_brand_ggplot2("_brand.yml") +
  theme(
    text = element_text(family = brand$typography$fonts[[1]]$family, size =
  )

ggplot(mpg, aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  labs(title = "A plot") +
  scale_color_manual(
    values = c(
      brand$color$palette$blue,
      brand$color$palette$black,
      brand$color$palette$yellow
```


A plot
hwy
40
30
20
2
3
4
5
6
7
displ
drv
4
f
r

Your turn

1. In the first code chunk of `01-exercise.qmd`, replace `theme_minimal()` with `theme_brand_ggplot2()`. You will need to supply a brand file path (`"_brand.yml"`).
2. Re-render and see what changes.
3. Change the `foreground` and `background` colors in `_brand.yml` and rerun the code to create the plot. What changes?

05:00

Your turn

1. In `01-exercise.qmd`, read in `_brand.yml` with `read_brand_yml()` from the {brand.yml} package. Store it as a variable named `brand`.
2. Change the color of the points to the brand's `blue` color and the line to the brand's `magenta` color. You'll use code that looks something like this:

```
geom_line(color = brand$color$palette$black)
```

3. **Bonus task!** Map a variable from the dataset to the color aesthetic and change the legend to use colors from the brand.

05:00

Make your own `_brand.yml`

- In RStudio, go to File > New File > Text file
- In Positron or VS Code, use the Explorer panel to add a new file.

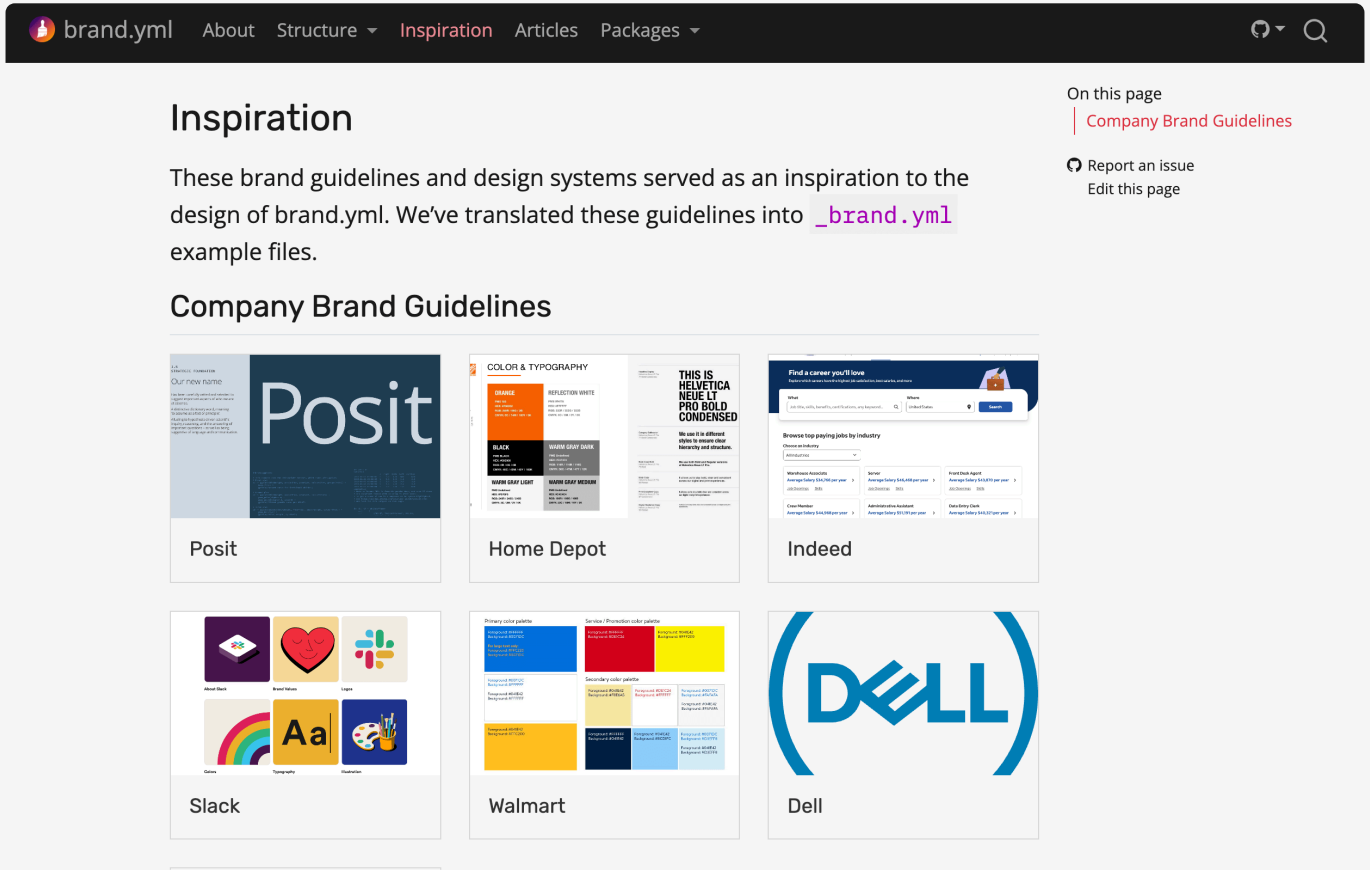
Save the file as `_brand.yml`

Or create a **Brand Extension** for sharing and distributing your brand

Don’t reinvent the wheel!

DO NOT try to write a `_brand.yml` from scratch!

Look at **the “Inspiration” page**!



Learn more: **https://posit-dev.github.io/brand-yml/inspiration/**

Your turn

Create a brand file for yourself, your project, or your organization.

Don’t try to do this from scratch! Use these resources:

- **The brand.yml documentation**
- **The brand.yml inspiration page**
- **This site’s `_brand.yml`**

05:00

What’s next?

Course outline

- ✅ ~~Intro to Quarto~~
- ✅ ~~Creating basic websites~~
- ✅ ~~Advanced website features~~
- ✅ ~~Publishing~~
- ✅ ~~Customization and branding~~
- Interactivity

Break!