香港中文大學
The Chinese University of Hong Kong

MORE ON CSS AND BOOTSTRAP

ESTR2106 2022-23 Term 2

Building Web Applications

Dr. Chuck-jee Chau
chuckjee@cse.cuhk.edu.hk

OUTLINE

- CSS Transforms
- CSS Transitions
- CSS Animations
- Flexbox/Grid Layout
- CSS Preprocessor: Sass
- Customizing Bootstrap
- Multiple color schemes
- Using Chrome DevTools
- Supporting dark mode with plain CSS

CSS TRANSFORMS

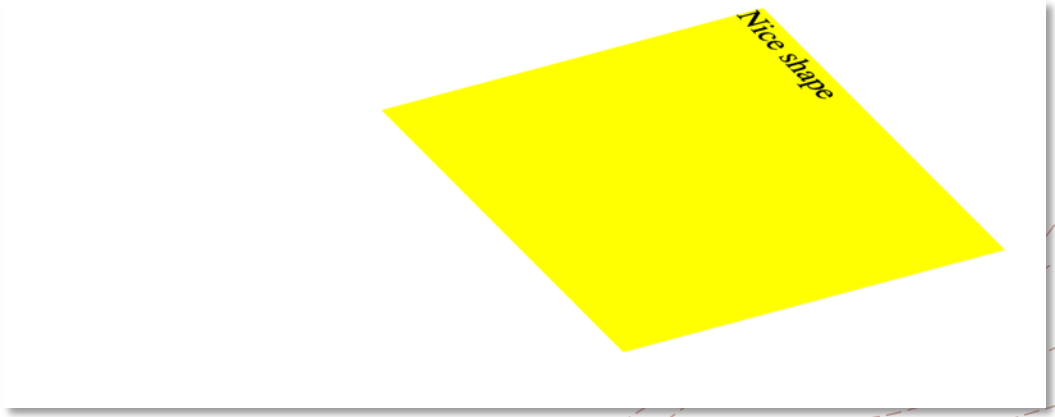
- 2D or 3D transformation in x-, y-, or z- axis
  - Translate: a simple disposition in pixels or other length units, e.g. `5px`
    - `translate(x,y)` / `translate3d(x,y,z)` / `translateX(x)` / `translateY(y)` / `translateZ(z)`
  - Scale: resizing the element with certain ratio, e.g., `1.5`
  - Rotate: rotating the element with an angle in degrees, e.g., `10deg`
    - Similar transformations as `translate*()`
  - Skew: distorting the element in 2D with angles in degrees
    - `skew(x-angle,y-angle)` / `skewX(angle)` / `skewY(angle)`

CSS TRANSFORMS

https://codepen.io/chuckjee/pen/VwMVoYZ

```
#square {
  width: 200px;
  height: 200px;
  background: yellow;
}
#square {
  transform: translateX(300px) rotate(45deg) skew(-30deg);
}
```

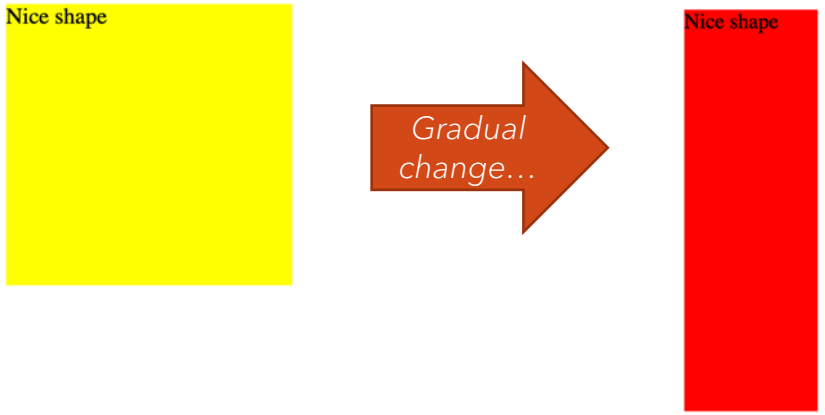
```
<div id="square">Nice shape</div>
```



CSS TRANSITIONS

- The transition property specifies:
  - *`property duration timing-function delay`*
  - In this order, with last 3 items optional
- *Property*: can be any property of an element
- *Duration*: the time to complete the transition (default=**`0s`**)
- *Timing-function*: **`ease`** (slow-fast-slow), **`linear`** (same speed throughout), **`ease-in`**, **`ease-out`**, **`ease-in-out`**, and more
- *Duration*: the time to wait before starting transition (default=**`0s`**)

CSS TRANSITIONS



https://codepen.io/chuckjee/pen/OJxaKMb

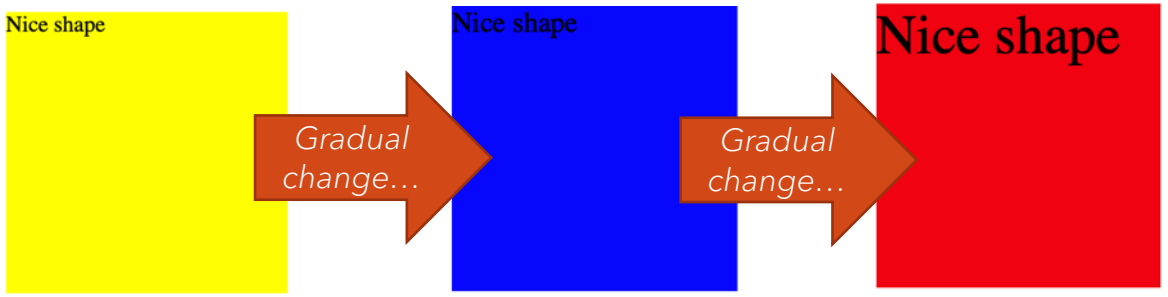
```
#square {
  width: 200px;
  height: 200px;
  background: yellow;
}
#square:hover {
  width: 100px;
  height: 300px;
  background: red;
  transition: width 2s ease-in, height 1s ease-out, background 3s linear;
}
```

CSS ANIMATIONS

- More elaborated animations can be created using keyframes
  - Specifying the behaviours "**`from`**" (0%) what and "**`to`**" (100%) what
  - Percentages can be used for more timepoints

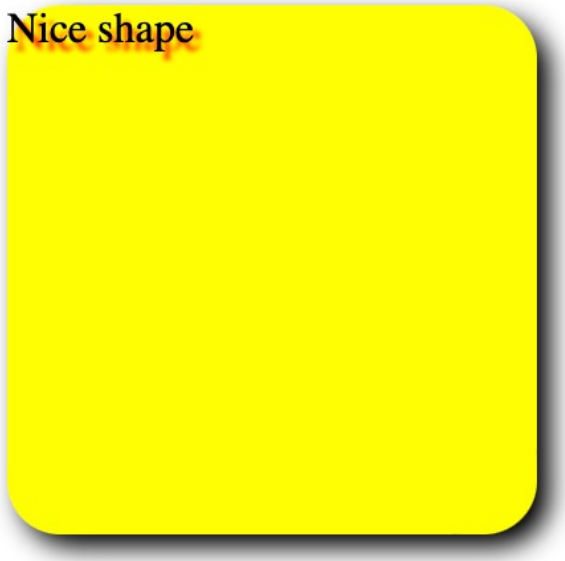
```
@keyframes my-animation {
  0% { font-size: 10px; background: yellow; }
  50% { font-size: 20px; background: blue; }
  100% { font-size: 40px; background: red; }
}
#square {
  width: 200px;
  height: 200px;
  background: yellow;
  animation-name: my-animation;
  animation-duration: 5s;
}
```

https://codepen.io/chuckjee/pen/poWQMya



OTHER CSS EFFECTS

- Rounded corners
  - `border-radius`
- Shadows
  - `text-shadow`, `box-shadow`
- CSS columns
  - `column-count`, etc.



https://codepen.io/chuckjee/pen/OJxaKpp

```
#square {
  width: 200px;
  height: 200px;
  background: yellow;

  border-radius: 20px;
  /* x, y, blur, color */
  text-shadow: 3px 3px 2px red;
  box-shadow: 5px 5px 10px;
}
```

CSS LOGICAL FUNCTIONS

- `min(…)`
  - Pick the smallest among parameters
- `max(…)`
  - Pick the largest among parameters
- `clamp(`*`min, ideal, max`*`)`
  - To ensure value is between *min* and *max*, based on *ideal*
- *See: https://web.dev/min-max-clamp/*

https://codepen.io/chuckjee/pen/vYeQoZG

```
#square {
  background: yellow;
  /* which is min? 500px vs 80% of screen */
  height: min(300px, 80vh);
  /* which is max? 500px or 50% of parent */
  width: max(500px, 50%);
}
```

CSS LAYOUT

- The display property is a nice tool for elements layout
  - `display: block;` → block element, taking up full width of parent
  - `display: inline;` → inline element, width depends on content
  - `display: inline-block;` → an inline element with adjustable width
  - `display: none;` → element is not shown, with width 0
- Other than these, there are more new options

FLEXBOX LAYOUT

- If you use **`display:flex`** for a box, the "block" contents inside will be flowed according to the flex-direction
  - Vertically centering contents in parent
  - Equalizing height/width for children
- This fluid layout is rather popular
  - e.g., Gallery of photos
- *Learn more: https://css-tricks.com/snippets/css/a-guide-to-flexbox*

```
<div id="box">
  <div>Hello</div>
  <div>There</div>
  <div>Longer box</div>
  <div>123</div>
  <h1>Tall box</h1>
</div>
```

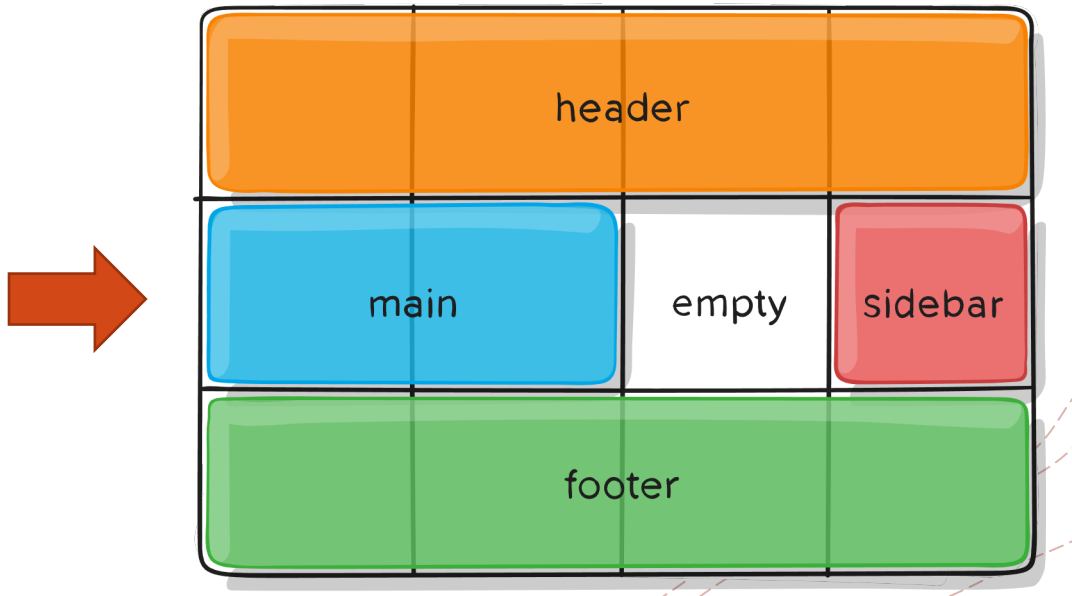
```
#box {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}
#box * {
  background: yellow;
  padding: 5px;
  margin: 5px;
}
```

https://codepen.io/chuckjee/pen/MWjLbJb

GRID LAYOUT

- Another option is to use **`display:grid`** for a box
- *See: https://css-tricks.com/snippets/css/complete-guide-grid/*

```
.item-a { grid-area: header; }
.item-b { grid-area: main; }
.item-c { grid-area: sidebar; }
.item-d { grid-area: footer; }
.container {
  display: grid;
  grid-template-columns: 50px 50px 50px 50px;
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
```



CSS PREPROCESSOR: SASS

- SASS (*Syntactically Awesome Style Sheets*) can be compiled (***transpiled***) to CSS
  - `.sass` files: original SASS syntax, with tab indentation and no `{ }`
  - `.scss` files: "Sassy" CSS syntax, superset of CSS syntax + Sass features
- Reduce redundancy in CSS code → save time!
- Install `sass` tool to convert into CSS
  - *See: https://sass-lang.com/install*
  - Now as a start, you can try online Sass compilers first

SASS VARIABLES

- Colors, length, fonts, are often reused across elements in a stylesheet
- Variable starts with **$**, and will be replaced in compilation
  - Suggesting using letters for variable name
  - **-** and **_** are considered the same!
- Scope is observed → global vs. local variables
  - Scope can be overridden by **`!global`**

SASS VARIABLES

```
$main-color: lightgreen;
$second-color: white;

body { background: $main-color; }
#box {
  background: $second-color;
  color: $main-color;
}
```

https://codepen.io/chuckjee/pen/dyVQxZY

```
<div id="box">A box here</div>
```

SASS NESTING

- Nested blocks in CSS makes more sense with HTML hierarchy

SCSS

```scss
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
  li {
    display: inline-block;
  }
  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

CSS

```css
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

SASS MODULES

- Just like other programming languages, you can load external code files with `@use`
- This is useful for a huge site, so stylesheets can have a hierarchical relationship

SASS MIXINS

- A mixin is a block of CSS properties grouped together for reusing under a name
- A value can be passed to the mixin for flexible CSS properties

SOME MORE ABOUT SASS

- Both `/* */` and `//` comments are supported in Sass
  - Only `/* */` comments will be compiled to CSS
- Most arithmetic and logical operators are available, with usual order of precedence
  - Except `/`, which is deprecated and should be replaced by math.div()
- Some powerful built-in modules
  - `sass:color`, `sass:list`, `sass:map`, `sass:math`, `sass:meta`, `sass:selector`, `sass:string`
- Automation is usually done for deploying the website, so all scss code are transpiled for publication with scripts

MORE ON BOOTSTRAP

CUSTOMIZING BOOTSTRAP

- Directly using Bootstrap from CDN
  → convenient and fast
- Download Bootstrap and use it together with your files
  → flexible and customizable
- Customize Bootstrap with ***Sass***, e.g., changing variable defaults
  - `$enable-shadows` and `$enable-gradients` are **`false`** by default
  - Learn here: *https://getbootstrap.com/docs/5.2/customize/overview/*

CUSTOMIZING BOOTSTRAP

- The Bootstrap source files are required:
  - *https://getbootstrap.com/docs/5.2/getting-started/download/#source-files*
- Such a file structure is recommended:
  - **`custom.scss`** would store your own variables
  - Inside, import either all of Bootstrap of only partially
    - *See: https://getbootstrap.com/docs/5.2/customize/sass/#importing*

```
your-project/
├── scss
│   └── custom.scss
└── bootstrap/
    ├── js
    └── scss
```

- An example of changing color schemes is shown here
  - Similar techniques can be used for further customization, available in Bootstrap docs!

MULTIPLE COLOR SCHEMES

- In **`custom.scss`**, load all default variables first

```
@import "../bootstrap/scss/bootstrap";
```

- And load the *mixins* (e.g., **`color-scheme`**)

```
@import "../bootstrap/scss/mixins";
```

- Then, a list of customized styles can be applied for, e.g., **`body`**

```
body {
    /* redefine theme colors */
    /* redefine theme color variables */
}
```

custom.scss

```
@import "../bootstrap/scss/bootstrap";
@import "../bootstrap/scss/mixins";

body {
  @include color-scheme(dark) {

    /* redefine theme colors for dark theme */
    $primary: #668cff;
    $secondary: #ffffff;
    $success: #66ff66;
    $danger: #ff6666;
    $warning: #ffd966;
    $info: #66ffff;
    $light: #e6e6e6;
    $dark: #b3b3b3;

    $theme-colors: (
        "primary": $primary,
        "secondary": $secondary,
        "success": $success,
        "danger": $danger,
        "warning": $warning,
        "info": $info,
        "light": $light,
        "dark": $dark,
    );

    /* redefine theme color variables */

    @each $color, $value in $theme-colors {
        --#{$variable-prefix}#{$color}: #{$value};
    }

    $theme-colors-rgb: map-loop($theme-colors, to-
rgb, "$value");

    @each $color, $value in $theme-colors-rgb {
        --#{$variable-prefix}#{$color}-rgb:
#{$value};
    }

    $body-color: #f2f2f2;
    $body-bg: #4d4d4d;

    --#{$variable-prefix}body-color: #{$body-color};
    --#{$variable-prefix}body-bg: #{$body-bg};

  }
}
```

MULTIPLE COLOR SCHEMES

- The customized scss needs to be compiled for actual use
  - Any Sass compiler can be used, for your computer's OS
  - Or, the official command line tools: *https://sass-lang.com/install*

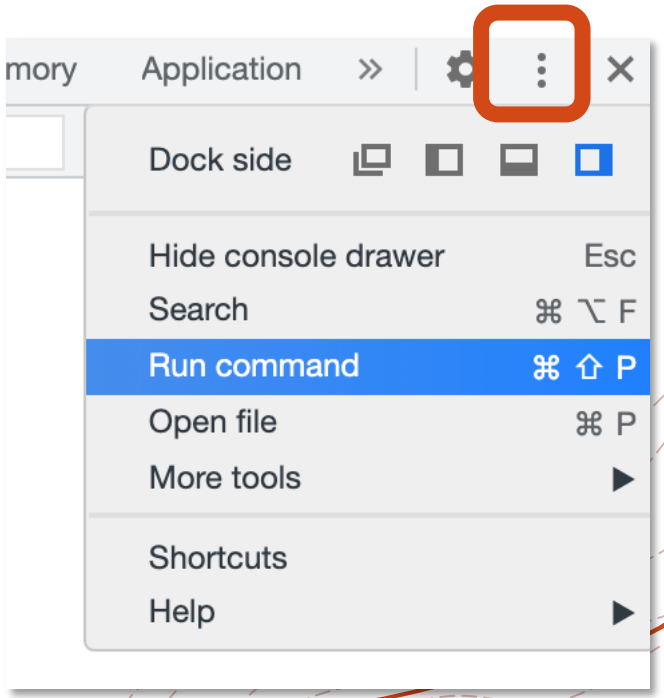
```
sass scss/custom.scss style-compiled.css
```

- Then, the new CSS stylesheet **`style-compiled.css`** can be used in an HTML file

```
<link rel="stylesheet" href="style-compiled.css">
```

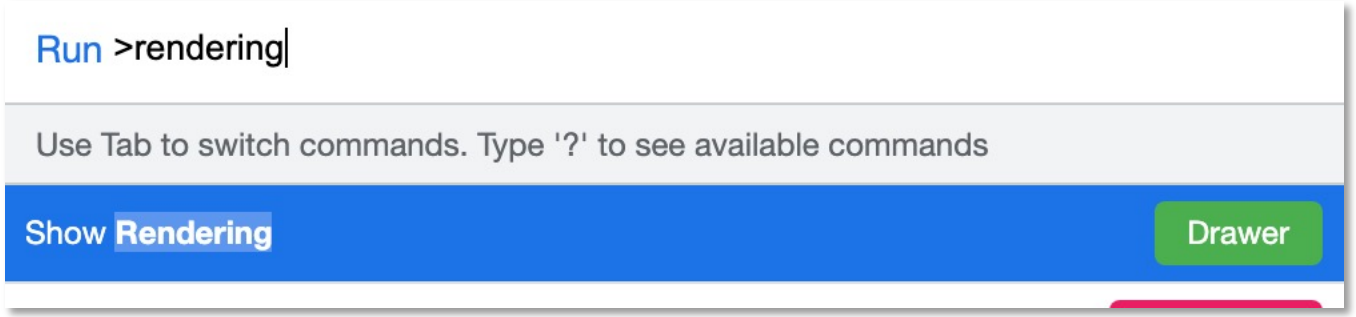
MULTIPLE COLOR SCHEMES

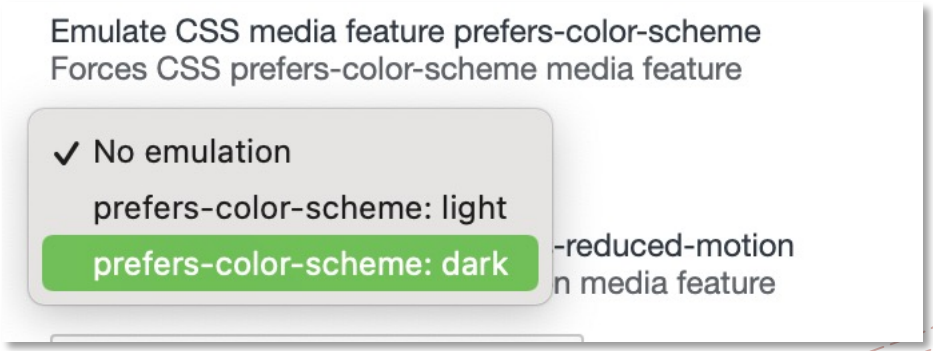
- The color scheme will change basing on the target OS settings
  - e.g., some people prefers dark mode after sunset
- In Google Chrome, you can ***emulate*** the effect for development and testing
  1. Start *Chrome DevTools* by right-clicking on a page and choose ***Inspect***
  2. Under the menu items of 3 dots ("***Customize and control DevTools***"), choose ***Run command***



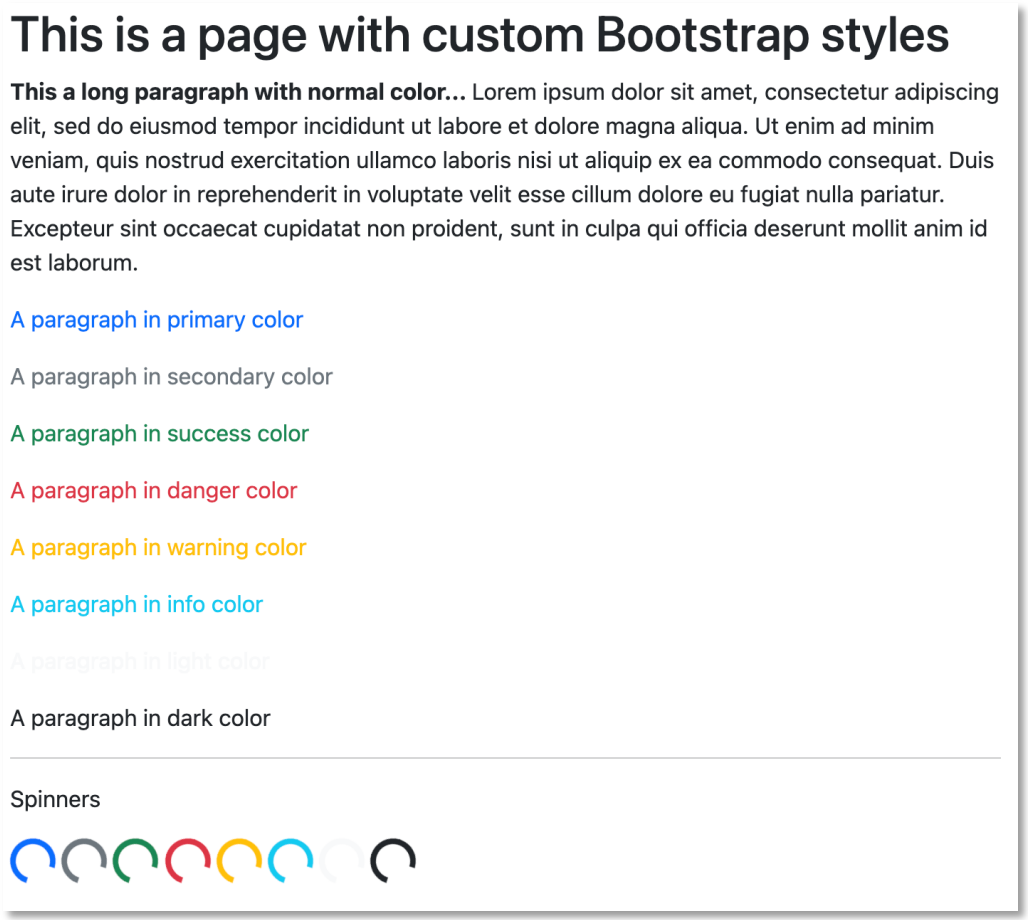
MULTIPLE COLOR SCHEMES

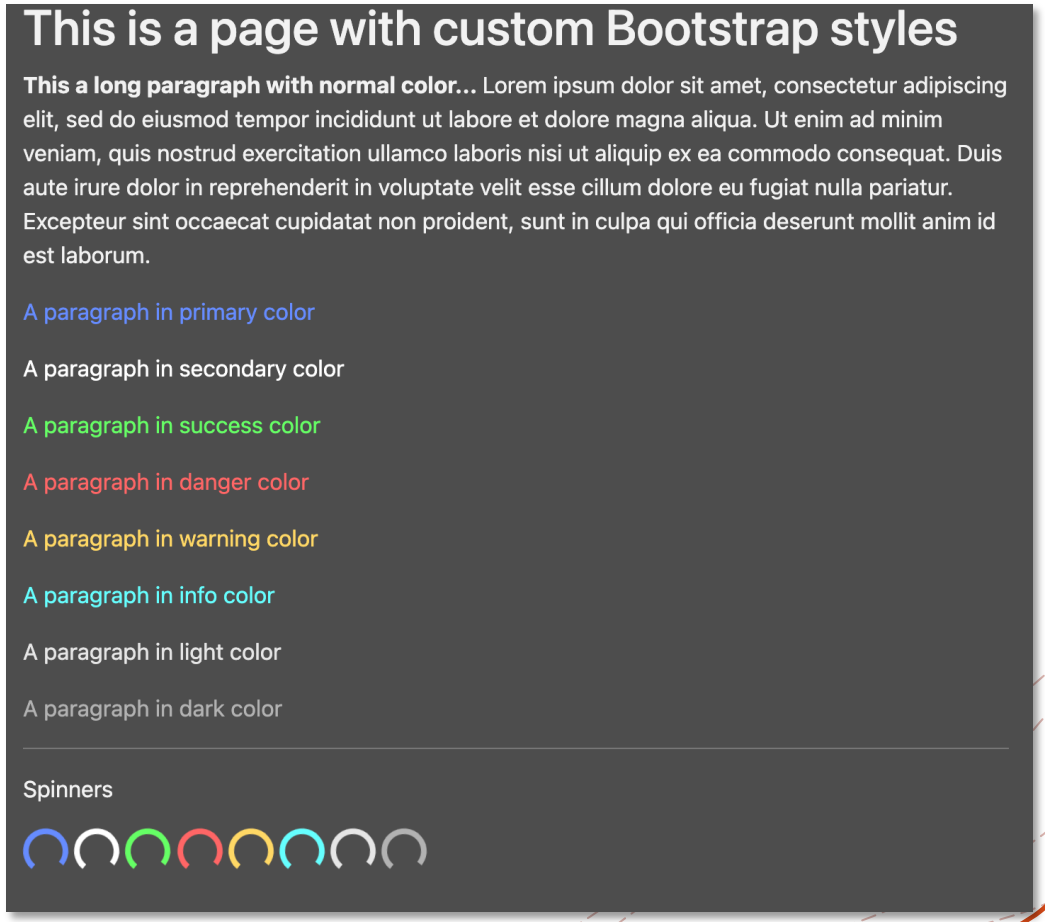
- DevTools (cont'd)
  3. Type *rendering* to choose ***Show Rendering***
  4. Scroll down and there is an option to ***Emulate CSS media feature prefers-color-scheme***
  5. Pick the preferred color scheme





MULTIPLE COLOR SCHEMES





SUPPORTING DARK MODE WITH PLAIN CSS

- If only plain CSS is used without Bootstrap, different external stylesheets can also be applied, depending on the current OS color scheme, indicated by **`prefers-color-scheme`** at the browser

```
<link rel="stylesheet" href="/dark.css"
      media="(prefers-color-scheme: dark)">
<link rel="stylesheet" href="/light.css"
      media="(prefers-color-scheme: light)">
```

- The **`media`** attribute is a powerful tool to pick between stylesheets basing on different ***screen sizes***, or even for ***printer output***
  - *See: https://www.w3schools.com/cssref/css3_pr_mediaquery.asp*
- Read more about dark mode: *https://web.dev/prefers-color-scheme/*

READ FURTHER…

Interesting articles on CSS-Tricks

https://css-tricks.com

Sass Tutorial on w3schools

https://www.w3schools.com/sass

Sass Documentation

https://sass-lang.com/documentation

Customize Bootstrap

https://getbootstrap.com/docs/5.2/customize/overview/