Theme colors

We use a subset of all colors to create a smaller color palette for generating color schemes, also available as Sass variables and a Sass map in Bootstrap's `scss/_variables.scss` file.

{{< theme-colors.inline >}} {{- range (index $.Site.Data "theme-colors") }}
{{ .name | title }}
{{ end -}} {{< /theme-colors.inline >}}

All these colors are available as a Sass map, `$theme-colors`.

{{< scss-docs name="theme-colors-map" file="scss/_variables.scss" >}}

Check out [our Sass maps and loops docs]({{< docsref "/customize/sass#maps-and-loops" >}}) for how to modify these colors.

All colors

All Bootstrap colors are available as Sass variables and a Sass map in `scss/_variables.scss` file. To avoid increased file sizes, we don't create text or background color classes for each of these variables. Instead, we choose a subset of these colors for a theme palette.

Be sure to monitor contrast ratios as you customize colors. As shown below, we've added three contrast ratios to each of the main colors—one for the swatch's current colors, one for against white, and one for against black.

{{< theme-colors.inline >}} {{- range $color := $.Site.Data.colors }} {{- if (and (not (eq $color.name "white")) (not (eq $color.name "gray")) (not (eq $color.name "gray-dark"))) }}
${{ $color.name }} {{ $color.hex }}
{{ range (seq 100 100 900) }}
${{ $color.name }}-{{ . }}
{{ end }}
{{ end -}} {{ end -}}
$gray-500 #adb5bd
{{- range $.Site.Data.grays }}
$gray-{{ .name }}
{{ end -}}
{{< /theme-colors.inline >}}
$black #000
$white #fff

Notes on Sass

Sass cannot programmatically generate variables, so we manually created variables for every tint and shade ourselves. We specify the midpoint value (e.g., `$blue-500`) and use custom color functions to tint (lighten) or shade (darken) our colors via Sass's `mix()` color function.

Using `mix()` is not the same as `lighten()` and `darken()`—the former blends the specified color with white or black, while the latter only adjusts the lightness value of each color. The result is a much more complete suite of colors, as shown in this CodePen demo.

Our `tint-color()` and `shade-color()` functions use `mix()` alongside our `$theme-color-interval` variable, which specifies a stepped percentage value for each mixed color we produce. See the `scss/_functions.scss` and `scss/_variables.scss` files for the full source code.

Color Sass maps

Bootstrap's source Sass files include three maps to help you quickly and easily loop over a list of colors and their hex values.

- `$colors` lists all our available base (`500`) colors
- `$theme-colors` lists all semantically named theme colors (shown below)
- `$grays` lists all tints and shades of gray

Within `scss/_variables.scss`, you'll find Bootstrap's color variables and Sass map. Here's an example of the `$colors` Sass map:

{{< scss-docs name="colors-map" file="scss/_variables.scss" >}}

Add, remove, or modify values within the map to update how they're used in many other components. Unfortunately at this time, not *every* component utilizes this Sass map. Future updates will strive to improve upon this. Until then, plan on making use of the `${color}` variables and this Sass map.

Example

Here's how you can use these in your Sass:

```scss
.alpha { color: $purple; }
.beta {
  color: $yellow-300;
  background-color: $indigo-900;
}
```

[Color]({{< docsref "/utilities/colors" >}}) and [background]({{< docsref "/utilities/background" >}}) utility classes are also available for setting `color` and `background-color` using the `500` color values.

Generating utilities

Added in v5.1.0

Bootstrap doesn't include `color` and `background-color` utilities for every color variable, but you can generate these yourself with our [utility API]({{< docsref "/utilities/api" >}}) and our extended Sass maps added in v5.1.0.

1. To start, make sure you've imported our functions, variables, mixins, and utilities.
2. Use our `map-merge-multiple()` function to quickly merge multiple Sass maps together in a new map.
3. Merge this new combined map to extend any utility with a `{color}-{level}` class name.

Here's an example that generates text color utilities (e.g., `.text-purple-500`) using the above steps.

```scss
@import "bootstrap/scss/functions";
@import "bootstrap/scss/variables";
@import "bootstrap/scss/mixins";
@import "bootstrap/scss/utilities";

$all-colors: map-merge-multiple($blues, $indigos, $purples, $pinks, $reds, $oranges,
$yellows, $greens, $teals, $cyans);
```

```
$utilities: map-merge(
  $utilities,
  (
    "color": map-merge(
      map-get($utilities, "color"),
      (
        values: map-merge(
          map-get(map-get($utilities, "color"), "values"),
          (
            $all-colors
          ),
        ),
      ),
    ),
  )
);

@import "bootstrap/scss/utilities/api";
```

This will generate new `.text-{color}-{level}` utilities for every color and level. You can do the same for any other utility and property as well.