Colors

Colorize text with color utilities. If you want to colorize links, you can use the [`.link-*` helper classes]({{< docsref "/helpers/colored-links" >}}) which have `:hover` and `:focus` states.

{{< example >}} {{< colors.inline >}} {{- range (index $.Site.Data "theme-colors") }}

.text-{{ .name }}

{{- end -}} {{< /colors.inline >}}
.text-body

.text-muted

.text-white

.text-black-50

.text-white-50

{{< /example >}}
{{< callout warning >}} **Deprecation:** With the addition of `.text-opacity-*` utilities and CSS variables for text utilities, `.text-black-50` and `.text-white-50` are deprecated as of v5.1.0. They'll be removed in v6.0.0. {{< /callout >}}

{{< callout info >}} {{< partial "callout-warning-color-assistive-technologies.md" >}} {{< /callout >}}

Opacity

Added in v5.1.0

As of v5.1.0, text color utilities are generated with Sass using CSS variables. This allows for real-time color changes without compilation and dynamic alpha transparency changes.

How it works

Consider our default `.text-primary` utility.

```
.text-primary {
  --bs-text-opacity: 1;
  color: rgba(var(--bs-primary-rgb), var(--bs-text-opacity)) !important;
}
```

We use an RGB version of our `--bs-primary` (with the value of `13, 110, 253`) CSS variable and attached a second CSS variable, `--bs-text-opacity`, for the alpha transparency (with a default value `1` thanks to a local CSS variable). That means anytime you use `.text-primary` now, your computed `color` value is `rgba(13, 110, 253, 1)`. The local CSS variable inside each `.text-*` class avoids inheritance issues so nested instances of the utilities don't automatically have a modified alpha transparency.

Example

To change that opacity, override `--bs-text-opacity` via custom styles or inline styles.

{{< example >}}

This is default primary text
This is 50% opacity primary text
{{< /example >}}

Or, choose from any of the `.text-opacity` utilities:

{{< example >}}

This is default primary text
This is 75% opacity primary text
This is 50% opacity primary text
This is 25% opacity primary text
{{< /example >}}

Specificity

Sometimes contextual classes cannot be applied due to the specificity of another selector. In some cases, a sufficient workaround is to wrap your element's content in a `<div>` or more semantic element with the desired class.

Sass

In addition to the following Sass functionality, consider reading about our included [CSS custom properties]({{< docsref "/customize/css-variables" >}}) (aka CSS variables) for colors and more.

Variables

Most `color` utilities are generated by our theme colors, reassigned from our generic color palette variables.

{{< scss-docs name="color-variables" file="scss/_variables.scss" >}}

{{< scss-docs name="theme-color-variables" file="scss/_variables.scss" >}}

Grayscale colors are also available, but only a subset are used to generate any utilities.

{{< scss-docs name="gray-color-variables" file="scss/_variables.scss" >}}

Map

Theme colors are then put into a Sass map so we can loop over them to generate our utilities, component modifiers, and more.

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

Grayscale colors are also available as a Sass map. **This map is not used to generate any utilities.**

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

RGB colors are generated from a separate Sass map:

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

And color opacities build on that with their own map that's consumed by the utilities API:

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

Utilities API

Color utilities are declared in our utilities API in `scss/_utilities.scss`. [Learn how to use the utilities API.]({{< docsref "/utilities/api#using-the-api" >}})

{{< scss-docs name="utils-color" file="scss/_utilities.scss" >}}