There's a newer version of Bootstrap!

Search docs... Ctrl + /

v5.1

View on GitHub

Colors

Convey meaning through `color` with a handful of color utility classes. Includes support for styling links with hover states, too.

Get $250 in credit when you ditch Big Tech baggage and try Vultr today!

ads via Carbon

On this page

Colors
Opacity
- How it works
- Example

Specificity
Sass
- Variables
- Map
- Utilities API

Colors

Colorize text with color utilities. If you want to colorize links, you can use the `.link-*` helper classes which have `:hover` and `:focus` states.

.text-primary

.text-secondary

.text-success

.text-danger

.text-warning

.text-info

.text-light

.text-dark

.text-body

.text-muted

.text-white

.text-black-50

.text-white-50

```
<p class="text-primary">.text-primary</p>
<p class="text-secondary">.text-secondary</p>
<p class="text-success">.text-success</p>
<p class="text-danger">.text-danger</p>
<p class="text-warning bg-dark">.text-warning</p>
<p class="text-info bg-dark">.text-info</p>
<p class="text-light bg-dark">.text-light</p>
<p class="text-dark">.text-dark</p>
<p class="text-body">.text-body</p>
<p class="text-muted">.text-muted</p>
<p class="text-white bg-dark">.text-white</p>
<p class="text-black-50">.text-black-50</p>
<p class="text-white-50 bg-dark">.text-white-50</p>
```

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.

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by

> the color is either obvious from the content itself (e.g. the visible text), or is included through alternative means, such as additional text hidden with the `.visually-hidden` class.

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.

This is default primary text
This is 50% opacity primary text

```
<div class="text-primary">This is default primary text</div>
<div class="text-primary" style="--bs-text-opacity: .5;">This is 50% opacity primary
```

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

This is default primary text
This is 75% opacity primary text
This is 50% opacity primary text
This is 25% opacity primary text

```
<div class="text-primary">This is default primary text</div>
<div class="text-primary text-opacity-75">This is 75% opacity primary text</div>
<div class="text-primary text-opacity-50">This is 50% opacity primary text</div>
<div class="text-primary text-opacity-25">This is 25% opacity primary text</div>
```

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 (aka CSS variables) for colors and more.

Variables

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

```
$blue:    #0d6efd;
$indigo:  #6610f2;
$purple:  #6f42c1;
$pink:    #d63384;
$red:     #dc3545;
$orange:  #fd7e14;
$yellow:  #ffc107;
$green:   #198754;
$teal:    #20c997;
$cyan:    #0dcaf0;
```

```
$primary:       $blue;
$secondary:     $gray-600;
$success:       $green;
$info:          $cyan;
$warning:       $yellow;
$danger:        $red;
$light:         $gray-100;
$dark:          $gray-900;
```

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

```
$white:    #fff;
$gray-100: #f8f9fa;
$gray-200: #e9ecef;
$gray-300: #dee2e6;
$gray-400: #ced4da;
$gray-500: #adb5bd;
$gray-600: #6c757d;
$gray-700: #495057;
$gray-800: #343a40;
$gray-900: #212529;
$black:    #000;
```

Map

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

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

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

```
$grays: (
  "100": $gray-100,
  "200": $gray-200,
  "300": $gray-300,
  "400": $gray-400,
  "500": $gray-500,
  "600": $gray-600,
  "700": $gray-700,
  "800": $gray-800,
  "900": $gray-900
);
```

RGB colors are generated from a separate Sass map:

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

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

```
$utilities-text: map-merge(
  $utilities-colors,
  (
    "black": to-rgb($black),
    "white": to-rgb($white),
    "body": to-rgb($body-color)
  )
);
$utilities-text-colors: map-loop($utilities-text, rgba-css-var, "$key", "text");
```

Utilities API

Color utilities are declared in our utilities API in `scss/_utilities.scss`. Learn how to use the utilities API.

```
    "color": (
      property: color,
      class: text,
      local-vars: (
        "text-opacity": 1
      ),
      values: map-merge(
        $utilities-text-colors,
```

```
      (
        "muted": $text-muted,
        "black-50": rgba($black, .5), // deprecated
        "white-50": rgba($white, .5), // deprecated
        "reset": inherit,
      )
    )
  ),
  "text-opacity": (
    css-var: true,
    class: text-opacity,
    values: (
      25: .25,
      50: .5,
      75: .75,
      100: 1
    )
  ),
```

Bootstrap

Designed and built with all the love in the world by the Bootstrap team with the help of our contributors.

Code licensed MIT, docs CC BY 3.0.

Currently v5.1.3.

Analytics by Fathom.

Links

Home

Docs

Examples

Themes

Blog

Swag Store

Guides

Getting started

Starter template

Webpack

Parcel

Projects

Community

Bootstrap 5
Bootstrap 4
Icons
RFS
npm starter
Issues
Discussions
Corporate sponsors
Open Collective
Stack Overflow