Bootstrap 5.2

Documentação oficial com 430 páginas

Get started with Bootstrap

Bootstrap is a powerful, feature-packed frontend toolkit. Build anything—from prototype to production—in minutes.

On this page

Quick start

Get started by including Bootstrap’s production-ready CSS and JavaScript via CDN without the need for any build steps. See it in practice with this Bootstrap CodePen demo.

1. **Create a new `index.html` file in your project root.** Include the `<meta name="viewport>` tag as well for proper responsive behavior in mobile devices.

-
  ```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

- **Include Bootstrap’s CSS and JS.** Place the `<link>` tag in the `<head>` for our CSS, and the `<script>` tag for our JavaScript bundle (including Popper for positioning dropdowns, poppers, and tooltips) before the closing `</body>`. Learn more about our CDN links.

```
<!doctype html>
<html lang="en">
  <head>
```

```
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/
bootstrap.min.css" rel="stylesheet"
integrity="sha384-0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmV
or" crossorigin="anonymous">
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.bundle
.min.js"
integrity="sha384-pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52n
D2" crossorigin="anonymous"></script>
  </body>
</html>
```

You can also include Popper and our JS separately. If you don’t plan to use dropdowns, popovers, or tooltips, save some kilobytes by not including Popper.

2.
```
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.5/dist/umd/popper.mi
n.js"
integrity="sha384-Xe+8cL9oJa6tN/veChSP7q+mnSPaj5Bcu9mPX5F5xIGE0DVittaqT5lo
rf0EI7Vk" crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.
min.js" integrity="sha384-
kjU+l4N0Yf4ZOJErLsIcvOU2qSb74wXpOhqTvwVx3OElZRweTnQ6d31fXEoRD1Jy"
crossorigin="anonymous"></script>
```

3. **Hello, world!** Open the page in your browser of choice to see your Bootstrapped page. Now you can start building with Bootstrap by creating your own layout, adding dozens of components, and utilizing our official examples.

CDN links

As reference, here are our primary CDN links.

Description	URL
CSS	`https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css`
JS	`https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.bundle.min.js`

You can also use the CDN to fetch any of our additional builds listed in the Contents page.

Next steps

- Read a bit more about some important global environment settings that Bootstrap utilizes.
- Read about what’s included in Bootstrap in our contents section and the list of components that require JavaScript below.
- Need a little more power? Consider building with Bootstrap by including the source files via package manager.

- Looking to use Bootstrap as a module with `<script type="module">`? Please refer to our using Bootstrap as a module section.

JS components

Curious which components explicitly require our JavaScript and Popper? Click the show components link below. If you're at all unsure about the general page structure, keep reading for an example page template.

Important globals

Bootstrap employs a handful of important global styles and settings, all of which are almost exclusively geared towards the *normalization* of cross browser styles. Let's dive in.

HTML5 doctype

Bootstrap requires the use of the HTML5 doctype. Without it, you'll see some funky and incomplete styling.

```
<!doctype html>
<html lang="en">
  ...
</html>
```

Responsive meta tag

Bootstrap is developed *mobile first*, a strategy in which we optimize code for mobile devices first and then scale up components as necessary using CSS media queries. To ensure proper rendering and touch zooming for all devices, add the responsive viewport meta tag to your `<head>`.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

You can see an example of this in action in the quick start.

Box-sizing

For more straightforward sizing in CSS, we switch the global `box-sizing` value from `content-box` to `border-box`. This ensures `padding` does not affect the final computed width of an element, but it can cause problems with some third-party software like Google Maps and Google Custom Search Engine.

On the rare occasion you need to override it, use something like the following:

```
.selector-for-some-widget {
  box-sizing: content-box;
}
```

With the above snippet, nested elements—including generated content via `::before` and `::after`—will all inherit the specified `box-sizing` for that `.selector-for-some-widget`.

Learn more about box model and sizing at CSS Tricks.

Reboot

For improved cross-browser rendering, we use Reboot to correct inconsistencies across browsers and devices while providing slightly more opinionated resets to common HTML elements.

Community

Stay up to date on the development of Bootstrap and reach out to the community with these helpful resources.

- Read and subscribe to The Official Bootstrap Blog.
- Join the official Slack room.
- Chat with fellow Bootstrappers in IRC. On the `irc.libera.chat` server, in the `#bootstrap` channel.
- Implementation help may be found at Stack Overflow (tagged `bootstrap-5`).
- Developers should use the keyword `bootstrap` on packages that modify or add to the functionality of Bootstrap when distributing through npm or similar delivery mechanisms for maximum discoverability.

You can also follow @getbootstrap on Twitter for the latest gossip and awesome music videos.

https://getbootstrap.com/docs/5.2/getting-started/introduction/

Browsers and devices

Learn about the browsers and devices, from modern to old, that are supported by Bootstrap, including known quirks and bugs for each.

On this page

Supported browsers

Bootstrap supports the **latest, stable releases** of all major browsers and platforms.

Alternative browsers which use the latest version of WebKit, Blink, or Gecko, whether directly or via the platform's web view API, are not explicitly supported. However, Bootstrap should (in most cases) display and function correctly in these browsers as well. More specific support information is provided below.

You can find our supported range of browsers and their versions in our `.browserslistrc` file:

```
# https://github.com/browserslist/browserslist#readme

>= 0.5%
last 2 major versions
not dead
Chrome >= 60
Firefox >= 60
Firefox ESR
iOS >= 12
Safari >= 12
not Explorer <= 11
```

We use Autoprefixer to handle intended browser support via CSS prefixes, which uses Browserslist to manage these browser versions. Consult their documentation for how to integrate these tools into your projects.

Mobile devices

Generally speaking, Bootstrap supports the latest versions of each major platform's default browsers. Note that proxy browsers (such as Opera Mini, Opera Mobile's Turbo mode, UC Browser Mini, Amazon Silk) are not supported.

	Chrome	Firefox	Safari	Android Browser & WebView
Android	Supported	Supported	—	v6.0+
Windows	Supported	Supported	Supported	—

Desktop browsers

Similarly, the latest versions of most desktop browsers are supported.

	Chrome	Firefox	Microsoft Edge	Opera	Safari
Mac	Supported	Supported	Supported	Supported	Supported
Windows	Supported	Supported	Supported	Supported	—

For Firefox, in addition to the latest normal stable release, we also support the latest Extended Support Release (ESR) version of Firefox.

Unofficially, Bootstrap should look and behave well enough in Chromium and Chrome for Linux, and Firefox for Linux, though they are not officially supported.

Internet Explorer

Internet Explorer is not supported. **If you require Internet Explorer support, please use Bootstrap v4.**

Modals and dropdowns on mobile

Overflow and scrolling

Support for `overflow: hidden;` on the `<body>` element is quite limited in iOS and Android. To that end, when you scroll past the top or bottom of a modal in either of those devices' browsers, the `<body>` content will begin to scroll. See Chrome bug #175502 (fixed in Chrome v40) and WebKit bug #153852.

iOS text fields and scrolling

As of iOS 9.2, while a modal is open, if the initial touch of a scroll gesture is within the boundary of a textual `<input>` or a `<textarea>`, the `<body>` content underneath the modal will be scrolled instead of the modal itself. See WebKit bug #153856.

Navbar Dropdowns

The `.dropdown-backdrop` element isn't used on iOS in the nav because of the complexity of z-indexing. Thus, to close dropdowns in navbars, you must directly click the dropdown element (or any other element which will fire a click event in iOS).

Browser zooming

Page zooming inevitably presents rendering artifacts in some components, both in Bootstrap and the rest of the web. Depending on the issue, we may be able to fix it (search first and then open an issue if need be). However, we tend to ignore these as they often have no direct solution other than hacky workarounds.

Validators

In order to provide the best possible experience to old and buggy browsers, Bootstrap uses CSS browser hacks in several places to target special CSS to certain browser versions in order to work around bugs in the browsers themselves. These hacks understandably cause CSS validators to complain that they are invalid. In a couple places, we also use bleeding-edge CSS features that aren’t yet fully standardized, but these are used purely for progressive enhancement.

These validation warnings don’t matter in practice since the non-hacky portion of our CSS does fully validate and the hacky portions don’t interfere with the proper functioning of the non-hacky portion, hence why we deliberately ignore these particular warnings.

Our HTML docs likewise have some trivial and inconsequential HTML validation warnings due to our inclusion of a workaround for a certain Firefox bug.

JavaScript

Bring Bootstrap to life with our optional JavaScript plugins. Learn about each plugin, our data and programmatic API options, and more.

On this page

Individual or compiled

Plugins can be included individually (using Bootstrap's individual `js/dist/*.js`), or all at once using `bootstrap.js` or the minified `bootstrap.min.js` (don't include both).

If you use a bundler (Webpack, Rollup…), you can use `/js/dist/*.js` files which are UMD ready.

Usage with JavaScript frameworks

While the Bootstrap CSS can be used with any framework, **the Bootstrap JavaScript is not fully compatible with JavaScript frameworks like React, Vue, and Angular** which assume full knowledge of the DOM. Both Bootstrap and the framework may attempt to mutate the same DOM element, resulting in bugs like dropdowns that are stuck in the "open" position.

A better alternative for those using this type of frameworks is to use a framework-specific package **instead of** the Bootstrap JavaScript. Here are some of the most popular options:

- React: React Bootstrap
- Vue: BootstrapVue
- Angular: ng-bootstrap

Using Bootstrap as a module

We provide a version of Bootstrap built as `ESM` (`bootstrap.esm.js` and `bootstrap.esm.min.js`) which allows you to use Bootstrap as a module in your browser, if your targeted browsers support it.

```
<script type="module">
  import { Toast } from 'bootstrap.esm.min.js'

  Array.from(document.querySelectorAll('.toast'))
    .forEach(toastNode => new Toast(toastNode))
</script>
```

Incompatible plugins

Due to browser limitations, some of our plugins, namely Dropdown, Tooltip and Popover plugins, cannot be used in a `<script>` tag with `module` type because they depend on Popper. For more information about the issue see here.

Dependencies

Some plugins and CSS components depend on other plugins. If you include plugins individually, make sure to check for these dependencies in the docs.

Our dropdowns, popovers and tooltips also depend on Popper.

Still want to use jQuery? It’s possible!

Bootstrap 5 is designed to be used without jQuery, but it’s still possible to use our components with jQuery. **If Bootstrap detects `jQuery` in the `window` object** it’ll add all of our components in jQuery’s plugin system; this means you’ll be able to do `$('[data-bs-toggle="tooltip"]').tooltip()` to enable tooltips. The same goes for our other components.

Data attributes

Nearly all Bootstrap plugins can be enabled and configured through HTML alone with data attributes (our preferred way of using JavaScript functionality). Be sure to **only use one set of data attributes on a single element** (e.g., you cannot trigger a tooltip and modal from the same button.)

Selectors

Currently to query DOM elements we use the native methods `querySelector` and `querySelectorAll` for performance reasons, so you have to use valid selectors. If you use special selectors, for example: `collapse:Example` be sure to escape them.

Events

Bootstrap provides custom events for most plugins' unique actions. Generally, these come in an infinitive and past participle form - where the infinitive (ex. `show`) is triggered at the start of an event, and its past participle form (ex. `shown`) is triggered on the completion of an action.

All infinitive events provide `preventDefault()` functionality. This provides the ability to stop the execution of an action before it starts. Returning false from an event handler will also automatically call `preventDefault()`.

```
const myModal = document.getElementById('myModal')

myModal.addEventListener('show.bs.modal', event => {
  if (!data) {
    return event.preventDefault() // stops modal from being shown
  }
})
```

jQuery events

Bootstrap will detect jQuery if `jQuery` is present in the `window` object and there is no `data-bs-no-jquery` attribute set on `<body>`. If jQuery is found, Bootstrap will emit events thanks to jQuery's event system. So if you want to listen to Bootstrap's events, you'll have to use the jQuery methods (`.on`, `.one`) instead of `addEventListener`.

```
$('#myTab a').on('shown.bs.tab', () => {
  // do something...
})
```

Programmatic API

All constructors accept an optional options object or nothing (which initiates a plugin with its default behavior):

```
const myModalEl = document.getElementById('myModal')

const modal = new bootstrap.Modal(myModalEl) // initialized with defaults
const modal1 = new bootstrap.Modal(myModalEl, { keyboard: false }) //
initialized with no keyboard
```

If you'd like to get a particular plugin instance, each plugin exposes a `getInstance` method. In order to retrieve it directly from an element, do this: `bootstrap.Popover.getInstance(myPopoverEl)`.

CSS selectors in constructors

You can also use a CSS selector as the first argument instead of a DOM element to initialize the plugin. Currently the element for the plugin is found by the `querySelector` method since our plugins support a single element only.

```
const modal = new bootstrap.Modal('#myModal')
const dropdown = new bootstrap.Dropdown('[data-bs-toggle="dropdown"]')
```

Asynchronous functions and transitions

All programmatic API methods are **asynchronous** and return to the caller once the transition is started but **before it ends**.

In order to execute an action once the transition is complete, you can listen to the corresponding event.

```
const myCollapseEl = document.getElementById('myCollapse')

myCollapseEl.addEventListener('shown.bs.collapse', event => {
  // Action to execute once the collapsible area is expanded
})
```

In addition a method call on a **transitioning component will be ignored**.

```
const myCarouselEl = document.getElementById('myCarousel')
const carousel = bootstrap.Carousel.getInstance(myCarouselEl) // Retrieve a
Carousel instance

myCarouselEl.addEventListener('slid.bs.carousel', event => {
  carousel.to('2') // Will slide to the slide 2 as soon as the transition to
slide 1 is finished
})

carousel.to('1') // Will start sliding to the slide 1 and returns to the caller
carousel.to('2') // !! Will be ignored, as the transition to the slide 1 is not
finished !!
```

Default settings

You can change the default settings for a plugin by modifying the plugin’s `Constructor.Default` object:

```
// changes default for the modal plugin's `keyboard` option to false
bootstrap.Modal.Default.keyboard = false
```

No conflict (only if you use jQuery)

Sometimes it is necessary to use Bootstrap plugins with other UI frameworks. In these circumstances, namespace collisions can occasionally occur. If this happens, you may call `.noConflict` on the plugin you wish to revert the value of.

```
const bootstrapButton = $.fn.button.noConflict() // return $.fn.button to
previously assigned value
$.fn.bootstrapBtn = bootstrapButton // give $().bootstrapBtn the Bootstrap
functionality
```

Version numbers

The version of each of Bootstrap’s plugins can be accessed via the `VERSION` property of the plugin’s constructor. For example, for the tooltip plugin:

```
bootstrap.Tooltip.VERSION // => "5.2.0-beta1"
```

No special fallbacks when JavaScript is disabled

Bootstrap's plugins don't fall back particularly gracefully when JavaScript is disabled. If you care about the user experience in this case, use `<noscript>` to explain the situation (and how to re-enable JavaScript) to your users, and/or add your own custom fallbacks.

Third-party libraries

Bootstrap does not officially support third-party JavaScript libraries like Prototype or jQuery UI. Despite `.noConflict` and namespaced events, there may be compatibility problems that you need to fix on your own.

Sanitizer

Tooltips and Popovers use our built-in sanitizer to sanitize options which accept HTML.

The default `allowList` value is the following:

```
const ARIA_ATTRIBUTE_PATTERN = /^aria-[\w-]*$/i
const DefaultAllowlist = {
  // Global attributes allowed on any supplied element below.
  '*': ['class', 'dir', 'id', 'lang', 'role', ARIA_ATTRIBUTE_PATTERN],
  a: ['target', 'href', 'title', 'rel'],
  area: [],
  b: [],
  br: [],
  col: [],
  code: [],
  div: [],
  em: [],
  hr: [],
  h1: [],
  h2: [],
  h3: [],
  h4: [],
  h5: [],
  h6: [],
  i: [],
  img: ['src', 'srcset', 'alt', 'title', 'width', 'height'],
  li: [],
  ol: [],
  p: [],
  pre: [],
  s: [],
  small: [],
  span: [],
  sub: [],
  sup: [],
  strong: [],
  u: [],
  ul: []
}
```

If you want to add new values to this default `allowList` you can do the following:

```
const myDefaultAllowList = bootstrap.Tooltip.Default.allowList

// To allow table elements
myDefaultAllowList.table = []
```

```
// To allow td elements and data-bs-option attributes on td elements
myDefaultAllowList.td = ['data-bs-option']

// You can push your custom regex to validate your attributes.
// Be careful about your regular expressions being too lax
const myCustomRegex = /^data-my-app-[\w-]+/
myDefaultAllowList['*'].push(myCustomRegex)
```

If you want to bypass our sanitizer because you prefer to use a dedicated library, for example DOMPurify, you should do the following:

```
const yourTooltipEl = document.getElementById('yourTooltip')
const tooltip = new bootstrap.Tooltip(yourTooltipEl, {
  sanitizeFn(content) {
    return DOMPurify.sanitize(content)
  }
})
```

RFS

Bootstrap’s resizing engine responsively scales common CSS properties to better utilize available space across viewports and devices.

On this page

What is RFS?

Bootstrap’s side project RFS is a unit resizing engine which was initially developed to resize font sizes (hence its abbreviation for Responsive Font Sizes). Nowadays RFS is capable of rescaling most CSS properties with unit values like `margin`, `padding`, `border-radius`, or even `box-shadow`.

The mechanism automatically calculates the appropriate values based on the dimensions of the browser viewport. It will be compiled into `calc()` functions with a mix of `rem` and viewport units to enable the responsive scaling behavior.

Using RFS

The mixins are included in Bootstrap and are available once you include Bootstrap’s `scss`. RFS can also be installed standalone if needed.

Using the mixins

The `rfs()` mixin has shorthands for `font-size`, `margin`, `margin-top`, `margin-right`, `margin-bottom`, `margin-left`, `padding`, `padding-top`, `padding-right`, `padding-bottom`, and `padding-left`. See the example below for source Sass and compiled CSS.

```
.title {
  @include font-size(4rem);
}
```

```
.title {
  font-size: calc(1.525rem + 3.3vw);
}

@media (min-width: 1200px) {
  .title {
    font-size: 4rem;
  }
}
```

Any other property can be passed to the `rfs()` mixin like this:

```
.selector {
  @include rfs(4rem, border-radius);
}
```

`!important` can also just be added to whatever value you want:

```
.selector {
  @include padding(2.5rem !important);
}
```

Using the functions

When you don't want to use the includes, there are also two functions:

- `rfs-value()` converts a value into a `rem` value if a `px` value is passed, in other cases it returns the same result.
- `rfs-fluid-value()` returns the fluid version of a value if the property needs rescaling.

In this example, we use one of Bootstrap's built-in responsive breakpoint mixins to only apply styling below the `lg` breakpoint.

```
.selector {
  @include media-breakpoint-down(lg) {
    padding: rfs-fluid-value(2rem);
    font-size: rfs-fluid-value(1.125rem);
  }
}

@media (max-width: 991.98px) {
  .selector {
    padding: calc(1.325rem + 0.9vw);
    font-size: 1.125rem; /* 1.125rem is small enough, so RFS won't rescale this
*/
  }
}
```

Extended documentation

RFS is a separate project under the Bootstrap organization. More about RFS and its configuration can be found on its GitHub repository.

RTL

Learn how to enable support for right-to-left text in Bootstrap across our layout, components, and utilities.

On this page

Get familiar

We recommend getting familiar with Bootstrap first by reading through our Getting Started Introduction page. Once you've run through it, continue reading here for how to enable RTL.

You may also want to read up on the RTLCSS project, as it powers our approach to RTL.

Experimental feature

The RTL feature is still **experimental** and will probably evolve according to user feedback. Spotted something or have an improvement to suggest? Open an issue, we'd love to get your insights.

Required HTML

There are two strict requirements for enabling RTL in Bootstrap-powered pages.

1. Set `dir="rtl"` on the `<html>` element.
2. Add an appropriate `lang` attribute, like `lang="ar"`, on the `<html>` element.

From there, you'll need to include an RTL version of our CSS. For example, here's the stylesheet for our compiled and minified CSS with RTL enabled:

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.rtl.min.css" integrity="sha384-
dc2NSrAXbAkjrdm9IYrX10fQq9SDG6Vjz7nQVKdKcJl3pC+k37e7qJR5MVSCS+wR"
crossorigin="anonymous">
```

Starter template

You can see the above requirements reflected in this modified RTL starter template.

```
<!doctype html>
<html lang="ar" dir="rtl">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.rtl.min.css" integrity="sha384-
dc2NSrAXbAkjrdm9IYrX10fQq9SDG6Vjz7nQVKdKcJl3pC+k37e7qJR5MVSCS+wR"
crossorigin="anonymous">

    <title>مرحبًا بالعالم!</title>
  </head>
  <body>
    <h1>مرحبًا بالعالم!</h1>

    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.bundle
.min.js"
integrity="sha384-pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52n
D2" crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.5/dist/umd/popper.min.js"
integrity="sha384-Xe+8cL9oJa6tN/veChSP7q+mnSPaj5Bcu9mPX5F5xIGE0DVittaqT5lorf0EI7
Vk" crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.min.js
" integrity="sha384-
kjU+l4N0Yf4ZOJErLsIcvOU2qSb74wXpOhqTvwVx3OElZRweTnQ6d31fXEoRD1Jy"
crossorigin="anonymous"></script>
    -->
  </body>
</html>
```

RTL examples

Get started with one of our several RTL examples.

Approach

Our approach to building RTL support into Bootstrap comes with two important decisions that impact how we write and use our CSS:

1. **First, we decided to build it with the RTLCSS project.** This gives us some powerful features for managing changes and overrides when moving from LTR to RTL. It also allows us to build two versions of Bootstrap from one codebase.
2. **Second, we've renamed a handful of directional classes to adopt a logical properties approach.** Most of you have already interacted with logical properties thanks to our flex utilities—they replace direction properties like `left` and `right` in favor `start` and `end`. That makes the class names and values appropriate for LTR and RTL without any overhead.

For example, instead of `.ml-3` for `margin-left`, use `.ms-3`.

Working with RTL, through our source Sass or compiled CSS, shouldn't be much different from our default LTR though.

Customize from source

When it comes to customization, the preferred way is to take advantage of variables, maps, and mixins. This approach works the same for RTL, even if it's post-processed from the compiled files, thanks to how RTLCSS works.

Custom RTL values

Using RTLCSS value directives, you can make a variable output a different value for RTL. For example, to decrease the weight for `$font-weight-bold` throughout the codebase, you may use the `/*rtl: {value}*/` syntax:

```
$font-weight-bold: 700 #{/* rtl:600 */} !default;
```

Which would output to the following for our default CSS and RTL CSS:

```
/* bootstrap.css */
dt {
  font-weight: 700 /* rtl:600 */;
}

/* bootstrap.rtl.css */
dt {
  font-weight: 600;
}
```

Alternative font stack

In the case you're using a custom font, be aware that not all fonts support the non-Latin alphabet. To switch from Pan-European to Arabic family, you may need to use `/*rtl:insert: {value}*/` in your font stack to modify the names of font families.

For example, to switch from `Helvetica Neue` font for LTR to `Helvetica Neue Arabic` for RTL, your Sass code could look like this:

```
$font-family-sans-serif:
  Helvetica Neue #{"/* rtl:insert:Arabic */"},
  // Cross-platform generic font family (default user interface font)
  system-ui,
  // Safari for macOS and iOS (San Francisco)
  -apple-system,
  // Chrome < 56 for macOS (San Francisco)
  BlinkMacSystemFont,
  // Windows
  "Segoe UI",
  // Android
  Roboto,
  // Basic web fallback
  Arial,
  // Linux
  "Noto Sans",
  // Sans serif fallback
  sans-serif,
  // Emoji fonts
  "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji" !
default;
```

LTR and RTL at the same time

Need both LTR and RTL on the same page? Thanks to RTLCSS String Maps, this is pretty straightforward. Wrap your `@import`s with a class, and set a custom rename rule for RTLCSS:

```
/* rtl:begin:options: {
  "autoRename": true,
  "stringMap":[ {
    "name": "ltr-rtl",
    "priority": 100,
    "search": ["ltr"],
    "replace": ["rtl"],
    "options": {
      "scope": "*",
      "ignoreCase": false
    }
  } ]
} */
.ltr {
  @import "../node_modules/bootstrap/scss/bootstrap";
}
/*rtl:end:options*/
```

After running Sass then RTLCSS, each selector in your CSS files will be prepended by `.ltr`, and `.rtl` for RTL files. Now you're able to use both files on the same page, and simply use `.ltr` or `.rtl` on your components wrappers to use one or the other direction.

Edge cases and known limitations

While this approach is understandable, please pay attention to the following:

1. When switching `.ltr` and `.rtl`, make sure you add `dir` and `lang` attributes accordingly.
2. Loading both files can be a real performance bottleneck: consider some optimization, and maybe try to load one of those files asynchronously.
3. Nesting styles this way will prevent our `form-validation-state()` mixin from working as intended, thus require you tweak it a bit by yourself. See #31223.

The breadcrumb case

The breadcrumb separator is the only case requiring its own brand new variable— namely `$breadcrumb-divider-flipped` —defaulting to `$breadcrumb-divider`.

Additional resources

- RTLCSS
- RTL Styling 101

Color

Bootstrap is supported by an extensive color system that themes our styles and components. This enables more comprehensive customization and extension for any project.

On this page

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.

Primary
Secondary
Success
Danger
Warning
Info
Light
Dark

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

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

Check out our Sass maps and loops docs 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.

$blue #0d6efd
$blue-100
$blue-200
$blue-300
$blue-400
$blue-500
$blue-600
$blue-700
$blue-800
$blue-900
$indigo #6610f2
$indigo-100
$indigo-200
$indigo-300
$indigo-400
$indigo-500
$indigo-600
$indigo-700
$indigo-800
$indigo-900
$purple #6f42c1
$purple-100
$purple-200
$purple-300
$purple-400
$purple-500
$purple-600
$purple-700
$purple-800
$purple-900
$pink #d63384
$pink-100
$pink-200
$pink-300
$pink-400
$pink-500
$pink-600
$pink-700
$pink-800
$pink-900
$red #dc3545
$red-100

$red-200
$red-300
$red-400
$red-500
$red-600
$red-700
$red-800
$red-900
$orange #fd7e14
$orange-100
$orange-200
$orange-300
$orange-400
$orange-500
$orange-600
$orange-700
$orange-800
$orange-900
$yellow #ffc107
$yellow-100
$yellow-200
$yellow-300
$yellow-400
$yellow-500
$yellow-600
$yellow-700
$yellow-800
$yellow-900
$green #198754
$green-100
$green-200
$green-300
$green-400
$green-500
$green-600
$green-700
$green-800
$green-900
$teal #20c997
$teal-100
$teal-200
$teal-300
$teal-400
$teal-500
$teal-600
$teal-700

$teal-800
$teal-900
$cyan #0dcaf0
$cyan-100
$cyan-200
$cyan-300
$cyan-400
$cyan-500
$cyan-600
$cyan-700
$cyan-800
$cyan-900
$gray-500 #adb5bd
$gray-100
$gray-200
$gray-300
$gray-400
$gray-500
$gray-600
$gray-700
$gray-800
$gray-900
$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:

```
$colors: (
  "blue":       $blue,
  "indigo":     $indigo,
  "purple":     $purple,
  "pink":       $pink,
  "red":        $red,
  "orange":     $orange,
  "yellow":     $yellow,
  "green":      $green,
  "teal":       $teal,
  "cyan":       $cyan,
  "black":      $black,
  "white":      $white,
  "gray":       $gray-600,
  "gray-dark":  $gray-800
);
```

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:

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

Color and 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 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.

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

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

Components

Learn how and why we build nearly all our components responsively and with base and modifier classes.

On this page

Base classes

Bootstrap’s components are largely built with a base-modifier nomenclature. We group as many shared properties as possible into a base class, like `.btn`, and then group individual styles for each variant into modifier classes, like `.btn-primary` or `.btn-success`.

To build our modifier classes, we use Sass’s `@each` loops to iterate over a Sass map. This is especially helpful for generating variants of a component by our `$theme-colors` and creating responsive variants for each breakpoint. As you customize these Sass maps and recompile, you’ll automatically see your changes reflected in these loops.

Check out our Sass maps and loops docs for how to customize these loops and extend Bootstrap’s base-modifier approach to your own code.

Modifiers

Many of Bootstrap’s components are built with a base-modifier class approach. This means the bulk of the styling is contained to a base class (e.g., `.btn`) while style variations are confined to modifier classes (e.g., `.btn-danger`). These modifier classes are built from the `$theme-colors` map to make customizing the number and name of our modifier classes.

Here are two examples of how we loop over the `$theme-colors` map to generate modifiers to the `.alert` and `.list-group` components.

```
// Generate contextual modifier classes for colorizing the alert.

@each $state, $value in $theme-colors {
  $alert-background: shift-color($value, $alert-bg-scale);
  $alert-border: shift-color($value, $alert-border-scale);
  $alert-color: shift-color($value, $alert-color-scale);

  @if (contrast-ratio($alert-background, $alert-color) < $min-contrast-ratio) {
    $alert-color: mix($value, color-contrast($alert-background), abs($alert-
color-scale));
  }
  .alert-#{$state} {
    @include alert-variant($alert-background, $alert-border, $alert-color);
  }
}
```

```
// List group contextual variants
//
// Add modifier classes to change text and background color on individual items.
// Organizationally, this must come after the `:hover` states.

@each $state, $value in $theme-colors {
  $list-group-variant-bg: shift-color($value, $list-group-item-bg-scale);
  $list-group-variant-color: shift-color($value, $list-group-item-color-scale);
  @if (contrast-ratio($list-group-variant-bg, $list-group-variant-color) < $min-
contrast-ratio) {
    $list-group-variant-color: mix($value, color-contrast($list-group-variant-
bg), abs($list-group-item-color-scale));
  }

  @include list-group-item-variant($state, $list-group-variant-bg, $list-group-
variant-color);
}
```

Responsive

These Sass loops aren't limited to color maps, either. You can also generate responsive variations of your components. Take for example our responsive alignment of the dropdowns where we mix an `@each` loop for the `$grid-breakpoints` Sass map with a media query include.

```
// We deliberately hardcode the `bs-` prefix because we check
// this custom property in JS to determine Popper's positioning

@each $breakpoint in map-keys($grid-breakpoints) {
  @include media-breakpoint-up($breakpoint) {
    $infix: breakpoint-infix($breakpoint, $grid-breakpoints);

    .dropdown-menu#{$infix}-start {
      --bs-position: start;

      &[data-bs-popper] {
        right: auto;
        left: 0;
      }
    }

    .dropdown-menu#{$infix}-end {
      --bs-position: end;

      &[data-bs-popper] {
        right: 0;
        left: auto;
      }
    }
  }
}
```

Should you modify your `$grid-breakpoints`, your changes will apply to all the loops iterating over that map.

```
$grid-breakpoints: (
  xs: 0,
  sm: 576px,
  md: 768px,
  lg: 992px,
  xl: 1200px,
  xxl: 1400px
);
```

For more information and examples on how to modify our Sass maps and variables, please refer to the Sass section of the Grid documentation.

Creating your own

We encourage you to adopt these guidelines when building with Bootstrap to create your own components. We've extended this approach ourselves to the custom components in our documentation and examples. Components like our callouts are built just like our provided components with base and modifier classes.

This is a callout. We built it custom for our docs so our messages to you stand out. It has three variants via modifier classes.

```
<div class="callout">...</div>
```

In your CSS, you'd have something like the following where the bulk of the styling is done via `.callout`. Then, the unique styles between each variant is controlled via modifier class.

```
// Base class
.callout {}

// Modifier classes
.callout-info {}
.callout-warning {}
.callout-danger {}
```

For the callouts, that unique styling is just a `border-left-color`. When you combine that base class with one of those modifier classes, you get your complete component family:

This is an info callout. Example text to show it in action.
This is a warning callout. Example text to show it in action.
This is a danger callout. Example text to show it in action.

CSS variables

Use Bootstrap’s CSS custom properties for fast and forward-looking design and development.

On this page

Bootstrap includes many CSS custom properties (variables) in its compiled CSS for real-time customization without the need to recompile Sass. These provide easy access to commonly used values like our theme colors, breakpoints, and primary font stacks when working in your browser’s inspector, a code sandbox, or general prototyping.

All our custom properties are prefixed with `bs-` to avoid conflicts with third party CSS.

Root variables

Here are the variables we include (note that the `:root` is required) that can be accessed anywhere Bootstrap’s CSS is loaded. They’re located in our `_root.scss` file and included in our compiled dist files.

```
:root {
  --bs-blue: #0d6efd;
  --bs-indigo: #6610f2;
  --bs-purple: #6f42c1;
  --bs-pink: #d63384;
  --bs-red: #dc3545;
  --bs-orange: #fd7e14;
  --bs-yellow: #ffc107;
  --bs-green: #198754;
  --bs-teal: #20c997;
  --bs-cyan: #0dcaf0;
  --bs-black: #000;
  --bs-white: #fff;
  --bs-gray: #6c757d;
  --bs-gray-dark: #343a40;
  --bs-gray-100: #f8f9fa;
  --bs-gray-200: #e9ecef;
  --bs-gray-300: #dee2e6;
  --bs-gray-400: #ced4da;
  --bs-gray-500: #adb5bd;
  --bs-gray-600: #6c757d;
  --bs-gray-700: #495057;
  --bs-gray-800: #343a40;
  --bs-gray-900: #212529;
  --bs-primary: #0d6efd;
  --bs-secondary: #6c757d;
  --bs-success: #198754;
  --bs-info: #0dcaf0;
  --bs-warning: #ffc107;
  --bs-danger: #dc3545;
  --bs-light: #f8f9fa;
  --bs-dark: #212529;
  --bs-primary-rgb: 13, 110, 253;
  --bs-secondary-rgb: 108, 117, 125;
  --bs-success-rgb: 25, 135, 84;
  --bs-info-rgb: 13, 202, 240;
  --bs-warning-rgb: 255, 193, 7;
  --bs-danger-rgb: 220, 53, 69;
  --bs-light-rgb: 248, 249, 250;
  --bs-dark-rgb: 33, 37, 41;
  --bs-white-rgb: 255, 255, 255;
  --bs-black-rgb: 0, 0, 0;
  --bs-body-color-rgb: 33, 37, 41;
  --bs-body-bg-rgb: 255, 255, 255;
  --bs-font-sans-serif: system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica
Neue", "Noto Sans", "Liberation Sans", Arial, sans-serif, "Apple Color Emoji",
"Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";
  --bs-font-monospace: SFMono-Regular, Menlo, Monaco, Consolas, "Liberation
Mono", "Courier New", monospace;
  --bs-gradient: linear-gradient(180deg, rgba(255, 255, 255, 0.15), rgba(255,
255, 255, 0));
  --bs-body-font-family: var(--bs-font-sans-serif);
  --bs-body-font-size: 1rem;
  --bs-body-font-weight: 400;
  --bs-body-line-height: 1.5;
  --bs-body-color: #212529;
  --bs-body-bg: #fff;
  --bs-border-width: 1px;
  --bs-border-style: solid;
  --bs-border-color: #dee2e6;
  --bs-border-color-translucent: rgba(0, 0, 0, 0.175);
  --bs-border-radius: 0.375rem;
  --bs-border-radius-sm: 0.25rem;
  --bs-border-radius-lg: 0.5rem;
  --bs-border-radius-xl: 1rem;
  --bs-border-radius-2xl: 2rem;
  --bs-border-radius-pill: 50rem;
  --bs-heading-color: ;
  --bs-link-color: #0d6efd;
  --bs-link-hover-color: #0a58ca;
  --bs-code-color: #d63384;
  --bs-highlight-bg: #fff3cd;
}
```

Component variables

Bootstrap 5 is increasingly making use of custom properties as local variables for various components. This way we reduce our compiled CSS, ensure styles aren’t inherited in places like nested tables, and allow some basic restyling and extending of Bootstrap components after Sass compilation.

Have a look at our table documentation for some insight into how we’re using CSS variables. Our navbars also use CSS variables as of v5.2.0. We’re also using CSS variables across our grids—primarily for gutters the new opt-in CSS grid—with more component usage coming in the future.

Whenever possible, we’ll assign CSS variables at the base component level (e.g., `.navbar` for navbar and its sub-components). This reduces guessing on where and how to customize, and allows for easy modifications by our team in future updates.

Prefix

Most CSS variables use a prefix to avoid collisions with your own codebase. This prefix is in addition to the `--` that's required on every CSS variable.

Customize the prefix via the `$prefix` Sass variable. By default, it's set to `bs-` (note the trailing dash).

Examples

CSS variables offer similar flexibility to Sass's variables, but without the need for compilation before being served to the browser. For example, here we're resetting our page's font and link styles with CSS variables.

```
body {
  font: 1rem/1.5 var(--bs-font-sans-serif);
}
a {
  color: var(--bs-blue);
}
```

Grid breakpoints

While we include our grid breakpoints as CSS variables (except for `xs`), be aware that **CSS variables do not work in media queries**. This is by design in the CSS spec for variables, but may change in coming years with support for `env()` variables. Check out this Stack Overflow answer for some helpful links. In the mean time, you can use these variables in other CSS situations, as well as in your JavaScript.

Breakpoints

Breakpoints are customizable widths that determine how your responsive layout behaves across device or viewport sizes in Bootstrap.

On this page

Core concepts

- **Breakpoints are the building blocks of responsive design.** Use them to control when your layout can be adapted at a particular viewport or device size.
- **Use media queries to architect your CSS by breakpoint.** Media queries are a feature of CSS that allow you to conditionally apply styles based on a set of browser and operating system parameters. We most commonly use `min-width` in our media queries.
- **Mobile first, responsive design is the goal.** Bootstrap's CSS aims to apply the bare minimum of styles to make a layout work at the smallest breakpoint, and then layers on styles to adjust that design for larger devices. This optimizes your CSS, improves rendering time, and provides a great experience for your visitors.

Available breakpoints

Bootstrap includes six default breakpoints, sometimes referred to as *grid tiers*, for building responsively. These breakpoints can be customized if you're using our source Sass files.

Breakpoint	Class infix	Dimensions
Extra small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

Each breakpoint was chosen to comfortably hold containers whose widths are multiples of 12. Breakpoints are also representative of a subset of common device sizes and viewport dimensions—they don't specifically target every use case or device. Instead, the ranges provide a strong and consistent foundation to build on for nearly any device.

These breakpoints are customizable via Sass—you'll find them in a Sass map in our `_variables.scss` stylesheet.

```
$grid-breakpoints: (
  xs: 0,
  sm: 576px,
  md: 768px,
  lg: 992px,
  xl: 1200px,
  xxl: 1400px
);
```

For more information and examples on how to modify our Sass maps and variables, please refer to the Sass section of the Grid documentation.

Media queries

Since Bootstrap is developed to be mobile first, we use a handful of media queries to create sensible breakpoints for our layouts and interfaces. These breakpoints are mostly based on minimum viewport widths and allow us to scale up elements as the viewport changes.

Min-width

Bootstrap primarily uses the following media query ranges—or breakpoints—in our source Sass files for our layout, grid system, and components.

```
// Source mixins

// No media query necessary for xs breakpoint as it's effectively `@media (min-
width: 0) { ... }`
@include media-breakpoint-up(sm) { ... }
@include media-breakpoint-up(md) { ... }
@include media-breakpoint-up(lg) { ... }
@include media-breakpoint-up(xl) { ... }
@include media-breakpoint-up(xxl) { ... }

// Usage

// Example: Hide starting at `min-width: 0`, and then show at the `sm`
breakpoint
.custom-class {
  display: none;
}
@include media-breakpoint-up(sm) {
  .custom-class {
    display: block;
  }
}
```

These Sass mixins translate in our compiled CSS using the values declared in our Sass variables. For example:

```
// X-Small devices (portrait phones, less than 576px)
// No media query for `xs` since this is the default in Bootstrap

// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) { ... }

// Medium devices (tablets, 768px and up)
```

```
@media (min-width: 768px) { ... }

// Large devices (desktops, 992px and up)
@media (min-width: 992px) { ... }

// X-Large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }

// XX-Large devices (larger desktops, 1400px and up)
@media (min-width: 1400px) { ... }
```

Max-width

We occasionally use media queries that go in the other direction (the given screen size *or smaller*):

```
// No media query necessary for xs breakpoint as it's effectively `@media (max-
width: 0) { ... }`
@include media-breakpoint-down(sm) { ... }
@include media-breakpoint-down(md) { ... }
@include media-breakpoint-down(lg) { ... }
@include media-breakpoint-down(xl) { ... }
@include media-breakpoint-down(xxl) { ... }

// Example: Style from medium breakpoint and down
@include media-breakpoint-down(md) {
  .custom-class {
    display: block;
  }
}
```

These mixins take those declared breakpoints, subtract `.02px` from them, and use them as our `max-width` values. For example:

```
// `xs` returns only a ruleset and no media query
// ... { ... }

// `sm` applies to x-small devices (portrait phones, less than 576px)
@media (max-width: 575.98px) { ... }

// `md` applies to small devices (landscape phones, less than 768px)
@media (max-width: 767.98px) { ... }

// `lg` applies to medium devices (tablets, less than 992px)
@media (max-width: 991.98px) { ... }

// `xl` applies to large devices (desktops, less than 1200px)
@media (max-width: 1199.98px) { ... }

// `xxl` applies to x-large devices (large desktops, less than 1400px)
@media (max-width: 1399.98px) { ... }
```

Why subtract .02px? Browsers don’t currently support range context queries, so we work around the limitations of `min-` and `max-` prefixes and viewports with fractional widths (which can occur under certain conditions on high-dpi devices, for instance) by using values with higher precision.

Single breakpoint

There are also media queries and mixins for targeting a single segment of screen sizes using the minimum and maximum breakpoint widths.

```
@include media-breakpoint-only(xs) { ... }
@include media-breakpoint-only(sm) { ... }
@include media-breakpoint-only(md) { ... }
@include media-breakpoint-only(lg) { ... }
@include media-breakpoint-only(xl) { ... }
@include media-breakpoint-only(xxl) { ... }
```

For example the `@include media-breakpoint-only(md) { ... }` will result in :

```
@media (min-width: 768px) and (max-width: 991.98px) { ... }
```

Between breakpoints

Similarly, media queries may span multiple breakpoint widths:

```
@include media-breakpoint-between(md, xl) { ... }
```

Which results in:

```
// Example
// Apply styles starting from medium devices and up to extra large devices
@media (min-width: 768px) and (max-width: 1199.98px) { ... }
```

Containers

Containers are a fundamental building block of Bootstrap that contain, pad, and align your content within a given device or viewport.

On this page

How they work

Containers are the most basic layout element in Bootstrap and are **required when using our default grid system**. Containers are used to contain, pad, and (sometimes) center the content within them. While containers *can* be nested, most layouts do not require a nested container.

Bootstrap comes with three different containers:

- `.container`, which sets a `max-width` at each responsive breakpoint
- `.container-{breakpoint}`, which is `width: 100%` until the specified breakpoint
- `.container-fluid`, which is `width: 100%` at all breakpoints

The table below illustrates how each container's `max-width` compares to the original `.container` and `.container-fluid` across each breakpoint.

See them in action and compare them in our Grid example.

	Extra small <576px	**Small ≥576px**	**Medium ≥768px**	**Large ≥992px**	**X-Large ≥1200px**	**XX-Large ≥1400px**
`.container`	100%	540px	720px	960px	1140px	1320px
`.container-sm`	100%	540px	720px	960px	1140px	1320px
`.container-md`	100%	100%	720px	960px	1140px	1320px
`.container-lg`	100%	100%	100%	960px	1140px	1320px
`.container-xl`	100%	100%	100%	100%	1140px	1320px
`.container-xxl`	100%	100%	100%	100%	100%	1320px
`.container-fluid`	100%	100%	100%	100%	100%	100%

Default container

Our default `.container` class is a responsive, fixed-width container, meaning its `max-width` changes at each breakpoint.

```
<div class="container">
  <!-- Content here -->
</div>
```

Responsive containers

Responsive containers allow you to specify a class that is 100% wide until the specified breakpoint is reached, after which we apply `max-width`s for each of the higher breakpoints. For example, `.container-sm` is 100% wide to start until the `sm` breakpoint is reached, where it will scale up with `md`, `lg`, `xl`, and `xxl`.

```
<div class="container-sm">100% wide until small breakpoint</div>
<div class="container-md">100% wide until medium breakpoint</div>
<div class="container-lg">100% wide until large breakpoint</div>
<div class="container-xl">100% wide until extra large breakpoint</div>
<div class="container-xxl">100% wide until extra extra large breakpoint</div>
```

Fluid containers

Use `.container-fluid` for a full width container, spanning the entire width of the viewport.

```
<div class="container-fluid">
  ...
</div>
```

Sass

As shown above, Bootstrap generates a series of predefined container classes to help you build the layouts you desire. You may customize these predefined container classes by modifying the Sass map (found in `_variables.scss`) that powers them:

```
$container-max-widths: (
  sm: 540px,
  md: 720px,
  lg: 960px,
  xl: 1140px,
  xxl: 1320px
);
```

In addition to customizing the Sass, you can also create your own containers with our Sass mixin.

```
// Source mixin
@mixin make-container($padding-x: $container-padding-x) {
  width: 100%;
  padding-right: $padding-x;
  padding-left: $padding-x;
  margin-right: auto;
  margin-left: auto;
}

// Usage
.custom-container {
  @include make-container();
}
```

For more information and examples on how to modify our Sass maps and variables, please refer to the Sass section of the Grid documentation.

Grid system

Use our powerful mobile-first flexbox grid to build layouts of all shapes and sizes thanks to a twelve column system, six default responsive tiers, Sass variables and mixins, and dozens of predefined classes.

On this page

Example

Bootstrap’s grid system uses a series of containers, rows, and columns to layout and align content. It’s built with flexbox and is fully responsive. Below is an example and an in-depth explanation for how the grid system comes together.

New to or unfamiliar with flexbox? Read this CSS Tricks flexbox guide for background, terminology, guidelines, and code snippets.

Column
Column
Column

html

```
<div class="container">
  <div class="row">
    <div class="col">
      Column
    </div>
    <div class="col">
```

```
      Column
    </div>
    <div class="col">
      Column
    </div>
  </div>
</div>
```

The above example creates three equal-width columns across all devices and viewports using our predefined grid classes. Those columns are centered in the page with the parent `.container`.

How it works

Breaking it down, here's how the grid system comes together:

- **Our grid supports six responsive breakpoints.** Breakpoints are based on `min-width` media queries, meaning they affect that breakpoint and all those above it (e.g., `.col-sm-4` applies to `sm`, `md`, `lg`, `xl`, and `xxl`). This means you can control container and column sizing and behavior by each breakpoint.
- **Containers center and horizontally pad your content.** Use `.container` for a responsive pixel width, `.container-fluid` for `width: 100%` across all viewports and devices, or a responsive container (e.g., `.container-md`) for a combination of fluid and pixel widths.
- **Rows are wrappers for columns.** Each column has horizontal `padding` (called a gutter) for controlling the space between them. This `padding` is then counteracted on the rows with negative margins to ensure the content in your columns is visually aligned down the left side. Rows also support modifier classes to uniformly apply column sizing and gutter classes to change the spacing of your content.
- **Columns are incredibly flexible.** There are 12 template columns available per row, allowing you to create different combinations of elements that span any number of columns. Column classes indicate the number of template columns to span (e.g., `col-4` spans four). `width`s are set in percentages so you always have the same relative sizing.
- **Gutters are also responsive and customizable.** Gutter classes are available across all breakpoints, with all the same sizes as our margin and padding spacing. Change horizontal gutters with `.gx-*` classes, vertical gutters with `.gy-*`, or all gutters with `.g-*` classes. `.g-0` is also available to remove gutters.
- **Sass variables, maps, and mixins power the grid.** If you don't want to use the predefined grid classes in Bootstrap, you can use our grid's source Sass to create your own with more semantic markup. We also include some CSS custom properties to consume these Sass variables for even greater flexibility for you.

Be aware of the limitations and bugs around flexbox, like the inability to use some HTML elements as flex containers.

Grid options

Bootstrap’s grid system can adapt across all six default breakpoints, and any breakpoints you customize. The six default grid tiers are as follow:

- Extra small (xs)
- Small (sm)
- Medium (md)
- Large (lg)
- Extra large (xl)
- Extra extra large (xxl)

As noted above, each of these breakpoints have their own container, unique class prefix, and modifiers. Here’s how the grid changes across these breakpoints:

	xs **<576px**	**sm** **≥576px**	**md** **≥768px**	**lg** **≥992px**	**xl** **≥1200px**	**xxl** **≥1400px**
Container `max-width`	None (auto)	540px	720px	960px	1140px	1320px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`	`.col-xxl-`
# of columns	12					
Gutter width	1.5rem (.75rem on left and right)					
Custom gutters	Yes					
Nestable	Yes					
Column ordering	Yes					

Auto-layout columns

Utilize breakpoint-specific column classes for easy column sizing without an explicit numbered class like `.col-sm-6`.

Equal-width

For example, here are two grid layouts that apply to every device and viewport, from `xs` to `xxl`. Add any number of unit-less classes for each breakpoint you need and every column will be the same width.

1 of 2
2 of 2
1 of 3
2 of 3
3 of 3

html

```
<div class="container">
  <div class="row">
    <div class="col">
      1 of 2
    </div>
    <div class="col">
      2 of 2
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col">
      2 of 3
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
</div>
```

Setting one column width

Auto-layout for flexbox grid columns also means you can set the width of one column and have the sibling columns automatically resize around it. You may use predefined grid classes (as shown below), grid mixins, or inline widths. Note that the other columns will resize no matter the width of the center column.

1 of 3

2 of 3 (wider)

3 of 3

1 of 3

2 of 3 (wider)

3 of 3

html

```
<div class="container">
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-6">
      2 of 3 (wider)
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-5">
      2 of 3 (wider)
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
</div>
```

Variable width content

Use `col-{breakpoint}-auto` classes to size columns based on the natural width of their content.

1 of 3
Variable width content
3 of 3
1 of 3
Variable width content
3 of 3

html

```
<div class="container">
  <div class="row justify-content-md-center">
    <div class="col col-lg-2">
      1 of 3
    </div>
    <div class="col-md-auto">
      Variable width content
    </div>
    <div class="col col-lg-2">
      3 of 3
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-md-auto">
      Variable width content
    </div>
    <div class="col col-lg-2">
      3 of 3
    </div>
  </div>
</div>
```

Responsive classes

Bootstrap’s grid includes six tiers of predefined classes for building complex responsive layouts. Customize the size of your columns on extra small, small, medium, large, or extra large devices however you see fit.

All breakpoints

For grids that are the same from the smallest of devices to the largest, use the `.col` and `.col-*` classes. Specify a numbered class when you need a particularly sized column; otherwise, feel free to stick to `.col`.

col
col
col
col
col-8

col-4

html

```
<div class="container">
  <div class="row">
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="col">col</div>
  </div>
  <div class="row">
    <div class="col-8">col-8</div>
    <div class="col-4">col-4</div>
  </div>
</div>
```

Stacked to horizontal

Using a single set of `.col-sm-*` classes, you can create a basic grid system that starts out stacked and becomes horizontal at the small breakpoint (`sm`).

col-sm-8

col-sm-4

col-sm

col-sm

col-sm

html

```
<div class="container">
  <div class="row">
    <div class="col-sm-8">col-sm-8</div>
    <div class="col-sm-4">col-sm-4</div>
  </div>
  <div class="row">
    <div class="col-sm">col-sm</div>
    <div class="col-sm">col-sm</div>
    <div class="col-sm">col-sm</div>
  </div>
</div>
```

Mix and match

Don’t want your columns to simply stack in some grid tiers? Use a combination of different classes for each tier as needed. See the example below for a better idea of how it all works.

.col-md-8

.col-6 .col-md-4

.col-6 .col-md-4

.col-6 .col-md-4

.col-6 .col-md-4

.col-6

.col-6

html

```
<div class="container">
```

```
  <!-- Stack the columns on mobile by making one full-width and the other half-
width -->
  <div class="row">
    <div class="col-md-8">.col-md-8</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>

  <!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on desktop
-->
  <div class="row">
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>

  <!-- Columns are always 50% wide, on mobile and desktop -->
  <div class="row">
    <div class="col-6">.col-6</div>
    <div class="col-6">.col-6</div>
  </div>
</div>
```

Row columns

Use the responsive `.row-cols-*` classes to quickly set the number of columns that best render your content and layout. Whereas normal `.col-*` classes apply to the individual columns (e.g., `.col-md-4`), the row columns classes are set on the parent `.row` as a shortcut. With `.row-cols-auto` you can give the columns their natural width.

Use these row columns classes to quickly create basic grid layouts or to control your card layouts.

Column
Column
Column
Column

html

```
<div class="container">
  <div class="row row-cols-2">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column
Column
Column
Column

html

```
<div class="container">
  <div class="row row-cols-3">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column

Column

Column

Column

html

```
<div class="container">
  <div class="row row-cols-auto">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column

Column

Column

Column

html

```
<div class="container">
  <div class="row row-cols-4">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column

Column

Column

Column

html

```
<div class="container">
  <div class="row row-cols-4">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col-6">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column

Column

Column

Column

html

```
<div class="container">
  <div class="row row-cols-1 row-cols-sm-2 row-cols-md-4">
```

```
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

You can also use the accompanying Sass mixin, `row-cols()`:

```
.element {
  // Three columns to start
  @include row-cols(3);

  // Five columns from medium breakpoint up
  @include media-breakpoint-up(md) {
    @include row-cols(5);
  }
}
```

Nesting

To nest your content with the default grid, add a new `.row` and set of `.col-sm-*` columns within an existing `.col-sm-*` column. Nested rows should include a set of columns that add up to 12 or fewer (it is not required that you use all 12 available columns).

Level 1: .col-sm-3
Level 2: .col-8 .col-sm-6
Level 2: .col-4 .col-sm-6

html

```
<div class="container">
  <div class="row">
    <div class="col-sm-3">
      Level 1: .col-sm-3
    </div>
    <div class="col-sm-9">
      <div class="row">
        <div class="col-8 col-sm-6">
          Level 2: .col-8 .col-sm-6
        </div>
        <div class="col-4 col-sm-6">
          Level 2: .col-4 .col-sm-6
        </div>
      </div>
    </div>
  </div>
</div>
```

Sass

When using Bootstrap’s source Sass files, you have the option of using Sass variables and mixins to create custom, semantic, and responsive page layouts. Our predefined grid classes use these same variables and mixins to provide a whole suite of ready-to-use classes for fast responsive layouts.

Variables

Variables and maps determine the number of columns, the gutter width, and the media query point at which to begin floating columns. We use these to generate the predefined grid classes documented above, as well as for the custom mixins listed below.

```
$grid-columns:      12;
$grid-gutter-width: 1.5rem;

$grid-breakpoints: (
  xs: 0,
  sm: 576px,
  md: 768px,
  lg: 992px,
  xl: 1200px,
  xxl: 1400px
);

$container-max-widths: (
  sm: 540px,
  md: 720px,
  lg: 960px,
  xl: 1140px,
  xxl: 1320px
);
```

Mixins

Mixins are used in conjunction with the grid variables to generate semantic CSS for individual grid columns.

```
// Creates a wrapper for a series of columns
@include make-row();

// Make the element grid-ready (applying everything but the width)
@include make-col-ready();

// Without optional size values, the mixin will create equal columns (similar to
using .col)
@include make-col();
@include make-col($size, $columns: $grid-columns);

// Offset with margins
@include make-col-offset($size, $columns: $grid-columns);
```

Example usage

You can modify the variables to your own custom values, or just use the mixins with their default values. Here’s an example of using the default settings to create a two-column layout with a gap between.

```
.example-container {
  @include make-container();
  // Make sure to define this width after the mixin to override
  // `width: 100%` generated by `make-container()`
  width: 800px;
}

.example-row {
  @include make-row();
}

.example-content-main {
  @include make-col-ready();

  @include media-breakpoint-up(sm) {
    @include make-col(6);
  }
  @include media-breakpoint-up(lg) {
    @include make-col(8);
  }
}

.example-content-secondary {
  @include make-col-ready();

  @include media-breakpoint-up(sm) {
    @include make-col(6);
  }
  @include media-breakpoint-up(lg) {
    @include make-col(4);
  }
}
```

Main content

Secondary content

html

```
<div class="example-container">
  <div class="example-row">
    <div class="example-content-main">Main content</div>
    <div class="example-content-secondary">Secondary content</div>
  </div>
</div>
```

Customizing the grid

Using our built-in grid Sass variables and maps, it’s possible to completely customize the predefined grid classes. Change the number of tiers, the media query dimensions, and the container widths—then recompile.

Columns and gutters

The number of grid columns can be modified via Sass variables. `$grid-columns` is used to generate the widths (in percent) of each individual column while `$grid-gutter-width` sets the width for the column gutters.

```
$grid-columns: 12 !default;
$grid-gutter-width: 1.5rem !default;
```

Grid tiers

Moving beyond the columns themselves, you may also customize the number of grid tiers. If you wanted just four grid tiers, you’d update the `$grid-breakpoints` and `$container-max-widths` to something like this:

```
$grid-breakpoints: (
  xs: 0,
  sm: 480px,
  md: 768px,
  lg: 1024px
);

$container-max-widths: (
  sm: 420px,
  md: 720px,
  lg: 960px
);
```

When making any changes to the Sass variables or maps, you’ll need to save your changes and recompile. Doing so will output a brand new set of predefined grid classes for column widths, offsets, and ordering. Responsive visibility utilities will also be updated to use the custom breakpoints. Make sure to set grid values in `px` (not `rem`, `em`, or `%`).

Columns

Learn how to modify columns with a handful of options for alignment, ordering, and offsetting thanks to our flexbox grid system. Plus, see how to use column classes to manage widths of non-grid elements.

On this page

Heads up! Be sure to read the Grid page first before diving into how to modify and customize your grid columns.

How they work

- **Columns build on the grid's flexbox architecture.** Flexbox means we have options for changing individual columns and modifying groups of columns at the row level. You choose how columns grow, shrink, or otherwise change.
- **When building grid layouts, all content goes in columns.** The hierarchy of Bootstrap's grid goes from container to row to column to your content. On rare occasions, you may combine content and column, but be aware there can be unintended consequences.
- **Bootstrap includes predefined classes for creating fast, responsive layouts.** With six breakpoints and a dozen columns at each grid tier, we have dozens of classes already built for you to create your desired layouts. This can be disabled via Sass if you wish.

Alignment

Use flexbox alignment utilities to vertically and horizontally align columns.

Vertical alignment

One of three columns
One of three columns
One of three columns
One of three columns

One of three columns

One of three columns

One of three columns

One of three columns

One of three columns

html

```
<div class="container">
  <div class="row align-items-start">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
  <div class="row align-items-center">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
  <div class="row align-items-end">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
</div>
```

One of three columns

One of three columns

One of three columns

html

```
<div class="container">
  <div class="row">
    <div class="col align-self-start">
      One of three columns
    </div>
    <div class="col align-self-center">
      One of three columns
    </div>
    <div class="col align-self-end">
      One of three columns
    </div>
  </div>
```

```
</div>
```

Horizontal alignment

One of two columns
One of two columns
One of two columns
One of two columns
One of two columns
One of two columns
One of two columns
One of two columns
One of two columns
One of two columns
One of two columns
One of two columns

html

```
<div class="container">
  <div class="row justify-content-start">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-center">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-end">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-around">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-between">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-evenly">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
</div>
```

Column wrapping

If more than 12 columns are placed within a single row, each group of extra columns will, as one unit, wrap onto a new line.

.col-9

.col-4

Since 9 + 4 = 13 > 12, this 4-column-wide div gets wrapped onto a new line as one contiguous unit.

.col-6

Subsequent columns continue along the new line.

html

```
<div class="container">
  <div class="row">
    <div class="col-9">.col-9</div>
    <div class="col-4">.col-4<br>Since 9 + 4 = 13 &gt; 12, this 4-column-wide
div gets wrapped onto a new line as one contiguous unit.</div>
    <div class="col-6">.col-6<br>Subsequent columns continue along the new
line.</div>
  </div>
</div>
```

Column breaks

Breaking columns to a new line in flexbox requires a small hack: add an element with `width: 100%` wherever you want to wrap your columns to a new line. Normally this is accomplished with multiple `.row`s, but not every implementation method can account for this.

.col-6 .col-sm-3

.col-6 .col-sm-3

.col-6 .col-sm-3

.col-6 .col-sm-3

html

```
<div class="container">
  <div class="row">
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>

    <!-- Force next columns to break to new line -->
    <div class="w-100"></div>

    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  </div>
</div>
```

You may also apply this break at specific breakpoints with our responsive display utilities.

.col-6 .col-sm-4
.col-6 .col-sm-4
.col-6 .col-sm-4
.col-6 .col-sm-4

html

```
<div class="container">
  <div class="row">
    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>

    <!-- Force next columns to break to new line at md breakpoint and up -->
    <div class="w-100 d-none d-md-block"></div>

    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
  </div>
</div>
```

Reordering

Order classes

Use `.order-` classes for controlling the **visual order** of your content. These classes are responsive, so you can set the `order` by breakpoint (e.g., `.order-1.order-md-2`). Includes support for `1` through `5` across all six grid tiers.

First in DOM, no order applied
Second in DOM, with a larger order
Third in DOM, with an order of 1

html

```
<div class="container">
  <div class="row">
    <div class="col">
      First in DOM, no order applied
    </div>
    <div class="col order-5">
      Second in DOM, with a larger order
    </div>
    <div class="col order-1">
      Third in DOM, with an order of 1
    </div>
  </div>
</div>
```

There are also responsive `.order-first` and `.order-last` classes that change the `order` of an element by applying `order: -1` and `order: 6`, respectively. These classes can also be intermixed with the numbered `.order-*` classes as needed.

First in DOM, ordered last
Second in DOM, unordered
Third in DOM, ordered first

html

```
<div class="container">
  <div class="row">
    <div class="col order-last">
      First in DOM, ordered last
    </div>
    <div class="col">
      Second in DOM, unordered
    </div>
    <div class="col order-first">
      Third in DOM, ordered first
    </div>
  </div>
</div>
```

Offsetting columns

You can offset grid columns in two ways: our responsive `.offset-` grid classes and our margin utilities. Grid classes are sized to match columns while margins are more useful for quick layouts where the width of the offset is variable.

Offset classes

Move columns to the right using `.offset-md-*` classes. These classes increase the left margin of a column by `*` columns. For example, `.offset-md-4` moves `.col-md-4` over four columns.

.col-md-4

.col-md-4 .offset-md-4

.col-md-3 .offset-md-3

.col-md-3 .offset-md-3

.col-md-6 .offset-md-3

html

```
<div class="container">
  <div class="row">
    <div class="col-md-4">.col-md-4</div>
    <div class="col-md-4 offset-md-4">.col-md-4 .offset-md-4</div>
  </div>
  <div class="row">
    <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
    <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
  </div>
  <div class="row">
    <div class="col-md-6 offset-md-3">.col-md-6 .offset-md-3</div>
  </div>
</div>
```

In addition to column clearing at responsive breakpoints, you may need to reset offsets. See this in action in the grid example.

.col-sm-5 .col-md-6

.col-sm-5 .offset-sm-2 .col-md-6 .offset-md-0

.col-sm-6 .col-md-5 .col-lg-6

.col-sm-6 .col-md-5 .offset-md-2 .col-lg-6 .offset-lg-0

html

```
<div class="container">
  <div class="row">
    <div class="col-sm-5 col-md-6">.col-sm-5 .col-md-6</div>
    <div class="col-sm-5 offset-sm-2 col-md-6 offset-md-0">.col-sm-5 .offset-sm-
2 .col-md-6 .offset-md-0</div>
  </div>
  <div class="row">
    <div class="col-sm-6 col-md-5 col-lg-6">.col-sm-6 .col-md-5 .col-lg-6</div>
    <div class="col-sm-6 col-md-5 offset-md-2 col-lg-6 offset-lg-0">.col-sm-
6 .col-md-5 .offset-md-2 .col-lg-6 .offset-lg-0</div>
  </div>
</div>
```

Margin utilities

With the move to flexbox in v4, you can use margin utilities like `.me-auto` to force sibling columns away from one another.

.col-md-4

.col-md-4 .ms-auto

.col-md-3 .ms-md-auto

.col-md-3 .ms-md-auto

.col-auto .me-auto

.col-auto

html

```
<div class="container">
  <div class="row">
    <div class="col-md-4">.col-md-4</div>
    <div class="col-md-4 ms-auto">.col-md-4 .ms-auto</div>
  </div>
  <div class="row">
    <div class="col-md-3 ms-md-auto">.col-md-3 .ms-md-auto</div>
    <div class="col-md-3 ms-md-auto">.col-md-3 .ms-md-auto</div>
  </div>
  <div class="row">
    <div class="col-auto me-auto">.col-auto .me-auto</div>
    <div class="col-auto">.col-auto</div>
  </div>
</div>
```

Standalone column classes

The `.col-*` classes can also be used outside a `.row` to give an element a specific width. Whenever column classes are used as non direct children of a row, the paddings are omitted.

.col-3: width of 25%

.col-sm-9: width of 75% above sm breakpoint

html

```
<div class="col-3 bg-light p-3 border">
  .col-3: width of 25%
</div>
<div class="col-sm-9 bg-light p-3 border">
  .col-sm-9: width of 75% above sm breakpoint
</div>
```

The classes can be used together with utilities to create responsive floated images. Make sure to wrap the content in a `.clearfix` wrapper to clear the float if the text is shorter.

A paragraph of placeholder text. We're using it here to show the use of the clearfix class. We're adding quite a few meaningless phrases here to demonstrate how the columns interact here with the floated image.

As you can see the paragraphs gracefully wrap around the floated image. Now imagine how this would look with some actual content in here, rather than just this boring placeholder text that goes on and on, but actually conveys no tangible information at. It simply takes up space and should not really be read.

And yet, here you are, still persevering in reading this placeholder text, hoping for some more insights, or some hidden easter egg of content. A joke, perhaps. Unfortunately, there's none of that here.

html

```html
<div class="clearfix">
  <img src="..." class="col-md-6 float-md-end mb-3 ms-md-3" alt="...">

  <p>
    A paragraph of placeholder text. We're using it here to show the use of the
clearfix class. We're adding quite a few meaningless phrases here to demonstrate
how the columns interact here with the floated image.
  </p>

  <p>
    As you can see the paragraphs gracefully wrap around the floated image. Now
imagine how this would look with some actual content in here, rather than just
this boring placeholder text that goes on and on, but actually conveys no
tangible information at. It simply takes up space and should not really be read.
  </p>

  <p>
    And yet, here you are, still persevering in reading this placeholder text,
hoping for some more insights, or some hidden easter egg of content. A joke,
perhaps. Unfortunately, there's none of that here.
  </p>
</div>
```

Gutters

Gutters are the padding between your columns, used to responsively space and align content in the Bootstrap grid system.

On this page

How they work

- **Gutters are the gaps between column content, created by horizontal `padding`.** We set `padding-right` and `padding-left` on each column, and use negative `margin` to offset that at the start and end of each row to align content.
- **Gutters start at `1.5rem` (`24px`) wide.** This allows us to match our grid to the padding and margin spacers scale.
- **Gutters can be responsively adjusted.** Use breakpoint-specific gutter classes to modify horizontal gutters, vertical gutters, and all gutters.

Horizontal gutters

`.gx-*` classes can be used to control the horizontal gutter widths. The `.container` or `.container-fluid` parent may need to be adjusted if larger gutters are used too to avoid unwanted overflow, using a matching padding utility. For example, in the following example we've increased the padding with `.px-4`:

Custom column padding
Custom column padding

html

```
<div class="container px-4">
  <div class="row gx-5">
    <div class="col">
     <div class="p-3 border bg-light">Custom column padding</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Custom column padding</div>
    </div>
  </div>
</div>
```

An alternative solution is to add a wrapper around the `.row` with the `.overflow-hidden` class:

Custom column padding
Custom column padding
html

```html
<div class="container overflow-hidden">
  <div class="row gx-5">
    <div class="col">
     <div class="p-3 border bg-light">Custom column padding</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Custom column padding</div>
    </div>
  </div>
</div>
```

Vertical gutters

`.gy-*` classes can be used to control the vertical gutter widths. Like the horizontal gutters, the vertical gutters can cause some overflow below the `.row` at the end of a page. If this occurs, you add a wrapper around `.row` with the `.overflow-hidden` class:

Custom column padding
Custom column padding
Custom column padding
Custom column padding
html

```html
<div class="container overflow-hidden">
  <div class="row gy-5">
    <div class="col-6">
      <div class="p-3 border bg-light">Custom column padding</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Custom column padding</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Custom column padding</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Custom column padding</div>
    </div>
  </div>
</div>
```

Horizontal & vertical gutters

`.g-*` classes can be used to control the horizontal gutter widths, for the following example we use a smaller gutter width, so there won’t be a need to add the `.overflow-hidden` wrapper class.

Custom column padding
Custom column padding
Custom column padding
Custom column padding
html

```
<div class="container">
  <div class="row g-2">
    <div class="col-6">
      <div class="p-3 border bg-light">Custom column padding</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Custom column padding</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Custom column padding</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Custom column padding</div>
    </div>
  </div>
</div>
```

Row columns gutters

Gutter classes can also be added to row columns. In the following example, we use responsive row columns and responsive gutter classes.

Row column
Row column
Row column
Row column
Row column
Row column
Row column
Row column
Row column
Row column

html

```
<div class="container">
  <div class="row row-cols-2 row-cols-lg-5 g-2 g-lg-3">
    <div class="col">
      <div class="p-3 border bg-light">Row column</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Row column</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Row column</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Row column</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Row column</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Row column</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Row column</div>
    </div>
    <div class="col">
```

```
      <div class="p-3 border bg-light">Row column</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Row column</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Row column</div>
    </div>
  </div>
</div>
```

No gutters

The gutters between columns in our predefined grid classes can be removed with `.g-0`. This removes the negative `margin`s from `.row` and the horizontal `padding` from all immediate children columns.

Need an edge-to-edge design? Drop the parent `.container` or `.container-fluid` and add `.mx-0` to the `.row` to prevent overflow.

In practice, here's how it looks. Note you can continue to use this with all other predefined grid classes (including column widths, responsive tiers, reorders, and more).

.col-sm-6 .col-md-8

.col-6 .col-md-4

html

```
<div class="row g-0">
  <div class="col-sm-6 col-md-8">.col-sm-6 .col-md-8</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>
```

Change the gutters

Classes are built from the `$gutters` Sass map which is inherited from the `$spacers` Sass map.

```
$grid-gutter-width: 1.5rem;
$gutters: (
  0: 0,
  1: $spacer * .25,
  2: $spacer * .5,
  3: $spacer,
  4: $spacer * 1.5,
  5: $spacer * 3,
);
```

https://getbootstrap.com/docs/5.2/layout/gutters/

Migrating to v5

Track and review changes to the Bootstrap source files, documentation, and components to help you migrate from v4 to v5.

On this page

Dependencies

- Dropped jQuery.
- Upgraded from Popper v1.x to Popper v2.x.
- Replaced Libsass with Dart Sass as our Sass compiler given Libsass was deprecated.

- Migrated from Jekyll to Hugo for building our documentation

Browser support

- Dropped Internet Explorer 10 and 11
- Dropped Microsoft Edge < 16 (Legacy Edge)
- Dropped Firefox < 60
- Dropped Safari < 12
- Dropped iOS Safari < 12
- Dropped Chrome < 60

Documentation changes

- Redesigned homepage, docs layout, and footer.
- Added new Parcel guide.
- Added new Customize section, replacing v4's Theming page, with new details on Sass, global configuration options, color schemes, CSS variables, and more.
- Reorganized all form documentation into new Forms section, breaking apart the content into more focused pages.
- Similarly, updated the Layout section, to flesh out grid content more clearly.
- Renamed "Navs" component page to "Navs & Tabs".
- Renamed "Checks" page to "Checks & radios".
- Redesigned the navbar and added a new subnav to make it easier to get around our sites and docs versions.
- Added new keyboard shortcut for the search field: `Ctrl + /`.

Sass

- We've ditched the default Sass map merges to make it easier to remove redundant values. Keep in mind you now have to define all values in the Sass maps like `$theme-colors`. Check out how to deal with Sass maps.
- Breaking Renamed `color-yiq()` function and related variables to `color-contrast()` as it's no longer related to YIQ colorspace. See #30168.
  - `$yiq-contrasted-threshold` is renamed to `$min-contrast-ratio`.
  - `$yiq-text-dark` and `$yiq-text-light` are respectively renamed to `$color-contrast-dark` and `$color-contrast-light`.
- Breaking Media query mixins parameters have changed for a more logical approach.
  - `media-breakpoint-down()` uses the breakpoint itself instead of the next breakpoint (e.g., `media-breakpoint-down(lg)` instead of `media-breakpoint-down(md)` targets viewports smaller than `lg`).
  - Similarly, the second parameter in `media-breakpoint-between()` also uses the breakpoint itself instead of the next breakpoint (e.g., `media-between(sm,`

`lg)` instead of `media-breakpoint-between(sm, md)` targets viewports between `sm` and `lg`).

- Breaking Removed print styles and `$enable-print-styles` variable. Print display classes are still around. See #28339.
- Breaking Dropped `color()`, `theme-color()`, and `gray()` functions in favor of variables. See #29083.
- Breaking Renamed `theme-color-level()` function to `color-level()` and now accepts any color you want instead of only `$theme-color` colors. See #29083 **Watch out:** `color-level()` was later on dropped in `v5.0.0-alpha3`.
- Breaking Renamed `$enable-prefers-reduced-motion-media-query` and `$enable-pointer-cursor-for-buttons` to `$enable-reduced-motion` and `$enable-button-pointers` for brevity.
- Breaking Removed the `bg-gradient-variant()` mixin. Use the `.bg-gradient` class to add gradients to elements instead of the generated `.bg-gradient-*` classes.
- Breaking **Removed previously deprecated mixins:**
  - `hover`, `hover-focus`, `plain-hover-focus`, and `hover-focus-active`
  - `float()`
  - `form-control-mixin()`
  - `nav-divider()`
  - `retina-img()`
  - `text-hide()` (also dropped the associated utility class, `.text-hide`)
  - `visibility()`
  - `form-control-focus()`
- Breaking Renamed `scale-color()` function to `shift-color()` to avoid collision with Sass's own color scaling function.
- `box-shadow` mixins now allow `null` values and drop `none` from multiple arguments. See #30394.
- The `border-radius()` mixin now has a default value.

Color system

- The color system which worked with `color-level()` and `$theme-color-interval` was removed in favor of a new color system. All `lighten()` and `darken()` functions in our codebase are replaced by `tint-color()` and `shade-color()`. These functions will mix the color with either white or black instead of changing its lightness by a fixed amount. The `shift-color()` will either tint or shade a color depending on whether its weight parameter is positive or negative. See #30622 for more details.
- Added new tints and shades for every color, providing nine separate colors for each base color, as new Sass variables.

- Improved color contrast. Bumped color contrast ratio from 3:1 to 4.5:1 and updated blue, green, cyan, and pink colors to ensure WCAG 2.1 AA contrast. Also changed our color contrast color from `$gray-900` to `$black`.
- To support our color system, we've added new custom `tint-color()` and `shade-color()` functions to mix our colors appropriately.

Grid updates

- **New breakpoint!** Added new `xxl` breakpoint for `1400px` and up. No changes to all other breakpoints.
- **Improved gutters.** Gutters are now set in rems, and are narrower than v4 (`1.5rem`, or about `24px`, down from `30px`). This aligns our grid system's gutters with our spacing utilities.
  - Added new gutter class (`.g-*`, `.gx-*`, and `.gy-*`) to control horizontal/vertical gutters, horizontal gutters, and vertical gutters.
  - Breaking Renamed `.no-gutters` to `.g-0` to match new gutter utilities.
- Columns no longer have `position: relative` applied, so you may have to add `.position-relative` to some elements to restore that behavior.
- Breaking Dropped several `.order-*` classes that often went unused. We now only provide `.order-1` to `.order-5` out of the box.
- Breaking Dropped the `.media` component as it can be easily replicated with utilities. See #28265 and the flex utilities page for an example.
- Breaking `bootstrap-grid.css` now only applies `box-sizing: border-box` to the column instead of resetting the global box-sizing. This way, our grid styles can be used in more places without interference.
- `$enable-grid-classes` no longer disables the generation of container classes anymore. See #29146.
- Updated the `make-col` mixin to default to equal columns without a specified size.

Content, Reboot, etc

- **RFS is now enabled by default.** Headings using the `font-size()` mixin will automatically adjust their `font-size` to scale with the viewport. *This feature was previously opt-in with v4.*
- Breaking Overhauled our display typography to replace our `$display-*` variables and with a `$display-font-sizes` Sass map. Also removed the individual `$display-*-weight` variables for a single `$display-font-weight` and adjusted `font-size`s.
- Added two new `.display-*` heading sizes, `.display-5` and `.display-6`.
- **Links are underlined by default** (not just on hover), unless they're part of specific components.

- **Redesigned tables** to refresh their styles and rebuild them with CSS variables for more control over styling.
- Breaking Nested tables do not inherit styles anymore.
- Breaking `.thead-light` and `.thead-dark` are dropped in favor of the `.table-*` variant classes which can be used for all table elements (`thead`, `tbody`, `tfoot`, `tr`, `th` and `td`).
- Breaking The `table-row-variant()` mixin is renamed to `table-variant()` and accepts only 2 parameters: `$color` (color name) and `$value` (color code). The border color and accent colors are automatically calculated based on the table factor variables.
- Split table cell padding variables into `-y` and `-x`.
- Breaking Dropped `.pre-scrollable` class. See #29135
- Breaking `.text-*` utilities do not add hover and focus states to links anymore. `.link-*` helper classes can be used instead. See #29267
- Breaking Dropped `.text-justify` class. See #29793
- Reset default horizontal `padding-left` on `<ul>` and `<ol>` elements from browser default `40px` to `2rem`.
- Added `$enable-smooth-scroll`, which applies `scroll-behavior: smooth` globally—except for users asking for reduced motion through `prefers-reduced-motion` media query. See #31877

RTL

- Horizontal direction specific variables, utilities, and mixins have all been renamed to use logical properties like those found in flexbox layouts—e.g., `start` and `end` in lieu of `left` and `right`.

Forms

- **Added new floating forms!** We've promoted the Floating labels example to fully supported form components. See the new Floating labels page.
- Breaking **Consolidated native and custom form elements.** Checkboxes, radios, selects, and other inputs that had native and custom classes in v4 have been consolidated. Now nearly all our form elements are entirely custom, most without the need for custom HTML.
    - `.custom-check` is now `.form-check`.
    - `.custom-check.custom-switch` is now `.form-check.form-switch`.
    - `.custom-select` is now `.form-select`.
    - `.custom-file` and `.form-file` have been replaced by custom styles on top of `.form-control`.
    - `.custom-range` is now `.form-range`.
    - Dropped native `.form-control-file` and `.form-control-range`.

- Breaking Dropped `.input-group-append` and `.input-group-prepend`. You can now just add buttons and `.input-group-text` as direct children of the input groups.
- The longstanding Missing border radius on input group with validation feedback bug is finally fixed by adding an additional `.has-validation` class to input groups with validation.
- Breaking **Dropped form-specific layout classes for our grid system.** Use our grid and utilities instead of `.form-group`, `.form-row`, or `.form-inline`.
- Breaking Form labels now require `.form-label`.
- Breaking `.form-text` no longer sets `display`, allowing you to create inline or block help text as you wish just by changing the HTML element.
- Validation icons are no longer applied to `<select>`s with `multiple`.
- Rearranged source Sass files under `scss/forms/`, including input group styles.

Components

- Unified `padding` values for alerts, breadcrumbs, cards, dropdowns, list groups, modals, popovers, and tooltips to be based on our `$spacer` variable. See #30564.

Accordion

- Added new accordion component.

Alerts

- Alerts now have examples with icons.
- Removed custom styles for `<hr>`s in each alert since they already use `currentColor`.

Badges

- Breaking Dropped all `.badge-*` color classes for background utilities (e.g., use `.bg-primary` instead of `.badge-primary`).
- Breaking Dropped `.badge-pill`—use the `.rounded-pill` utility instead.
- Breaking Removed hover and focus styles for `<a>` and `<button>` elements.
- Increased default padding for badges from `.25em`/`.5em` to `.35em`/`.65em`.

Breadcrumbs

- Simplified the default appearance of breadcrumbs by removing `padding`, `background-color`, and `border-radius`.
- Added new CSS custom property `--bs-breadcrumb-divider` for easy customization without needing to recompile CSS.

Buttons

- Breaking **Toggle buttons, with checkboxes or radios, no longer require JavaScript and have new markup.** We no longer require a wrapping element, add `.btn-check` to the `<input>`, and pair it with any `.btn` classes on the `<label>`. See #30650. *The docs for this has moved from our Buttons page to the new Forms section.*
- Breaking **Dropped `.btn-block` for utilities.** Instead of using `.btn-block` on the `.btn`, wrap your buttons with `.d-grid` and a `.gap-*` utility to space them as needed. Switch to responsive classes for even more control over them. Read the docs for some examples.
- Updated our `button-variant()` and `button-outline-variant()` mixins to support additional parameters.
- Updated buttons to ensure increased contrast on hover and active states.
- Disabled buttons now have `pointer-events: none;`.

Card

- Breaking Dropped `.card-deck` in favor of our grid. Wrap your cards in column classes and add a parent `.row-cols-*` container to recreate card decks (but with more control over responsive alignment).
- Breaking Dropped `.card-columns` in favor of Masonry. See #28922.
- Breaking Replaced the `.card` based accordion with a new Accordion component.

Carousel

- Added new `.carousel-dark` variant for dark text, controls, and indicators (great for lighter backgrounds).
- Replaced chevron icons for carousel controls with new SVGs from Bootstrap Icons.

Close button

- Breaking Renamed `.close` to `.btn-close` for a less generic name.
- Close buttons now use a `background-image` (embedded SVG) instead of a `&times;` in the HTML, allowing for easier customization without the need to touch your markup.
- Added new `.btn-close-white` variant that uses `filter: invert(1)` to enable higher contrast dismiss icons against darker backgrounds.

Collapse

- Removed scroll anchoring for accordions.

Dropdowns

- Added new `.dropdown-menu-dark` variant and associated variables for on-demand dark dropdowns.

- Added new variable for `$dropdown-padding-x`.
- Darkened the dropdown divider for improved contrast.
- Breaking All the events for the dropdown are now triggered on the dropdown toggle button and then bubbled up to the parent element.
- Dropdown menus now have a `data-bs-popper="static"` attribute set when the positioning of the dropdown is static and `data-bs-popper="none"` when dropdown is in the navbar. This is added by our JavaScript and helps us use custom position styles without interfering with Popper's positioning.
- Breaking Dropped `flip` option for dropdown plugin in favor of native Popper configuration. You can now disable the flipping behavior by passing an empty array for `fallbackPlacements` option in flip modifier.
- Dropdown menus can now be clickable with a new `autoClose` option to handle the auto close behavior. You can use this option to accept the click inside or outside the dropdown menu to make it interactive.
- Dropdowns now support `.dropdown-item`s wrapped in `<li>`s.

Jumbotron

- Breaking Dropped the jumbotron component as it can be replicated with utilities. See our new Jumbotron example for a demo.

List group

- Added new `.list-group-numbered` modifier to list groups.

Navs and tabs

- Added new `null` variables for `font-size`, `font-weight`, `color`, and `:hover color` to the `.nav-link` class.

Navbars

- Breaking Navbars now require a container within (to drastically simplify spacing requirements and CSS required).

Offcanvas

- Added the new offcanvas component.

Pagination

- Pagination links now have customizable `margin-left` that are dynamically rounded on all corners when separated from one another.
- Added `transition`s to pagination links.

Popovers

- Breaking Renamed `.arrow` to `.popover-arrow` in our default popover template.
- Renamed `whiteList` option to `allowList`.

Spinners

- Spinners now honor `prefers-reduced-motion: reduce` by slowing down animations. See #31882.
- Improved spinner vertical alignment.

Toasts

- Toasts can now be positioned in a `.toast-container` with the help of positioning utilities.
- Changed default toast duration to 5 seconds.
- Removed `overflow: hidden` from toasts and replaced with proper `border-radius`s with `calc()` functions.

Tooltips

- Breaking Renamed `.arrow` to `.tooltip-arrow` in our default tooltip template.
- Breaking The default value for the `fallbackPlacements` is changed to `['top', 'right', 'bottom', 'left']` for better placement of popper elements.
- Breaking Renamed `whiteList` option to `allowList`.

Utilities

- Breaking Renamed several utilities to use logical property names instead of directional names with the addition of RTL support:
  - Renamed `.left-*` and `.right-*` to `.start-*` and `.end-*`.
  - Renamed `.float-left` and `.float-right` to `.float-start` and `.float-end`.
  - Renamed `.border-left` and `.border-right` to `.border-start` and `.border-end`.
  - Renamed `.rounded-left` and `.rounded-right` to `.rounded-start` and `.rounded-end`.
  - Renamed `.ml-*` and `.mr-*` to `.ms-*` and `.me-*`.
  - Renamed `.pl-*` and `.pr-*` to `.ps-*` and `.pe-*`.
  - Renamed `.text-left` and `.text-right` to `.text-start` and `.text-end`.
- Breaking Disabled negative margins by default.
- Added new `.bg-body` class for quickly setting the `<body>`’s background to additional elements.

- Added new position utilities for `top`, `right`, `bottom`, and `left`. Values include `0`, `50%`, and `100%` for each property.
- Added new `.translate-middle-x` & `.translate-middle-y` utilities to horizontally or vertically center absolute/fixed positioned elements.
- Added new `border-width` utilities.
- Breaking Renamed `.text-monospace` to `.font-monospace`.
- Breaking Removed `.text-hide` as it's an antiquated method for hiding text that shouldn't be used anymore.
- Added `.fs-*` utilities for `font-size` utilities (with RFS enabled). These use the same scale as HTML's default headings (1-6, large to small), and can be modified via Sass map.
- Breaking Renamed `.font-weight-*` utilities as `.fw-*` for brevity and consistency.
- Breaking Renamed `.font-style-*` utilities as `.fst-*` for brevity and consistency.
- Added `.d-grid` to display utilities and new `gap` utilities (`.gap`) for CSS Grid and flexbox layouts.
- Breaking Removed `.rounded-sm` and `rounded-lg`, and introduced a new scale of classes, `.rounded-0` to `.rounded-3`. See #31687.
- Added new `line-height` utilities: `.lh-1`, `.lh-sm`, `.lh-base` and `.lh-lg`. See here.
- Moved the `.d-none` utility in our CSS to give it more weight over other display utilities.
- Extended the `.visually-hidden-focusable` helper to also work on containers, using `:focus-within`.

Helpers

- Breaking **Responsive embed helpers have been renamed to ratio helpers** with new class names and improved behaviors, as well as a helpful CSS variable.
  - Classes have been renamed to change `by` to `x` in the aspect ratio. For example, `.ratio-16by9` is now `.ratio-16x9`.
  - We've dropped the `.embed-responsive-item` and element group selector in favor of a simpler `.ratio > *` selector. No more class is needed, and the ratio helper now works with any HTML element.
  - The `$embed-responsive-aspect-ratios` Sass map has been renamed to `$aspect-ratios` and its values have been simplified to include the class name and the percentage as the `key: value` pair.
  - CSS variables are now generated and included for each value in the Sass map. Modify the `--bs-aspect-ratio` variable on the `.ratio` to create any custom aspect ratio.
- Breaking **"Screen reader" classes are now "visually hidden" classes.**

  - Changed the Sass file from `scss/helpers/_screenreaders.scss` to `scss/helpers/_visually-hidden.scss`
  - Renamed `.sr-only` and `.sr-only-focusable` to `.visually-hidden` and `.visually-hidden-focusable`
  - Renamed `sr-only()` and `sr-only-focusable()` mixins to `visually-hidden()` and `visually-hidden-focusable()`.
- `bootstrap-utilities.css` now also includes our helpers. Helpers don't need to be imported in custom builds anymore.

JavaScript

- **Dropped jQuery dependency** and rewrote plugins to be in regular JavaScript.
- Breaking Data attributes for all JavaScript plugins are now namespaced to help distinguish Bootstrap functionality from third parties and your own code. For example, we use `data-bs-toggle` instead of `data-toggle`.
- **All plugins can now accept a CSS selector as the first argument.** You can either pass a DOM element or any valid CSS selector to create a new instance of the plugin:

```
• var modal = new bootstrap.Modal('#myModal')
var dropdown = new bootstrap.Dropdown('[data-bs-toggle="dropdown"]')
```

- `popperConfig` can be passed as a function that accepts the Bootstrap's default Popper config as an argument, so that you can merge this default configuration in your way. **Applies to dropdowns, popovers, and tooltips.**
- The default value for the `fallbackPlacements` is changed to `['top', 'right', 'bottom', 'left']` for better placement of Popper elements. **Applies to dropdowns, popovers, and tooltips.**
- Removed underscore from public static methods like `_getInstance()` → `getInstance()`.

Utilities for layout

For faster mobile-friendly and responsive development, Bootstrap includes dozens of utility classes for showing, hiding, aligning, and spacing content.

On this page

Changing `display`

Use our display utilities for responsively toggling common values of the `display` property. Mix it with our grid system, content, or components to show or hide them across specific viewports.

Flexbox options

Bootstrap is built with flexbox, but not every element's `display` has been changed to `display: flex` as this would add many unnecessary overrides and unexpectedly change key browser behaviors. Most of our components are built with flexbox enabled.

Should you need to add `display: flex` to an element, do so with `.d-flex` or one of the responsive variants (e.g., `.d-sm-flex`). You'll need this class or `display` value to allow the use of our extra flexbox utilities for sizing, alignment, spacing, and more.

Margin and padding

Use the `margin` and `padding` spacing utilities to control how elements and components are spaced and sized. Bootstrap includes a six-level scale for spacing utilities, based on a `1rem` value default `$spacer` variable. Choose values for all viewports (e.g., `.me-3` for `margin-right: 1rem` in LTR), or pick responsive variants to target specific viewports (e.g., `.me-md-3` for `margin-right: 1rem` —in LTR— starting at the `md` breakpoint).

Toggle `visibility`

When toggling `display` isn't needed, you can toggle the `visibility` of an element with our visibility utilities. Invisible elements will still affect the layout of the page, but are visually hidden from visitors.

Z-index

While not a part of Bootstrap's grid system, z-indexes play an important part in how our components overlay and interact with one another.

Several Bootstrap components utilize `z-index`, the CSS property that helps control layout by providing a third axis to arrange content. We utilize a default z-index scale in Bootstrap that's been designed to properly layer navigation, tooltips and popovers, modals, and more.

These higher values start at an arbitrary number, high and specific enough to ideally avoid conflicts. We need a standard set of these across our layered components—tooltips, popovers, navbars, dropdowns, modals—so we can be reasonably consistent in the behaviors. There's no reason we couldn't have used `100`+ or `500`+.

We don't encourage customization of these individual values; should you change one, you likely need to change them all.

```
$zindex-dropdown:                   1000;
$zindex-sticky:                     1020;
$zindex-fixed:                      1030;
$zindex-offcanvas-backdrop:         1040;
$zindex-offcanvas:                  1045;
$zindex-modal-backdrop:             1050;
$zindex-modal:                      1055;
$zindex-popover:                    1070;
$zindex-tooltip:                    1080;
$zindex-toast:                      1090;
```

To handle overlapping borders within components (e.g., buttons and inputs in input groups), we use low single digit `z-index` values of `1`, `2`, and `3` for default, hover, and active states. On hover/focus/active, we bring a particular element to the forefront with a higher `z-index` value to show their border over the sibling elements.

CSS Grid

Learn how to enable, use, and customize our alternate layout system built on CSS Grid with examples and code snippets.

On this page

Bootstrap’s default grid system represents the culmination of over a decade of CSS layout techniques, tried and tested by millions of people. But, it was also created without many of the modern CSS features and techniques we’re seeing in browsers like the new CSS Grid.

Heads up—our CSS Grid system is experimental and opt-in as of v5.1.0! We included it in our documentation’s CSS to demonstrate it for you, but it’s disabled by default. Keep reading to learn how to enable it in your projects.

How it works

With Bootstrap 5, we’ve added the option to enable a separate grid system that’s built on CSS Grid, but with a Bootstrap twist. You still get classes you can apply on a whim to build responsive layouts, but with a different approach under the hood.

- **CSS Grid is opt-in.** Disable the default grid system by setting `$enable-grid-classes: false` and enable the CSS Grid by setting `$enable-cssgrid: true`. Then, recompile your Sass.
- **Replace instances of `.row` with `.grid`.** The `.grid` class sets `display: grid` and creates a `grid-template` that you build on with your HTML.
- **Replace `.col-*` classes with `.g-col-*` classes.** This is because our CSS Grid columns use the `grid-column` property instead of `width`.

- **Columns and gutter sizes are set via CSS variables.** Set these on the parent `.grid` and customize however you want, inline or in a stylesheet, with `--bs-columns` and `--bs-gap`.

In the future, Bootstrap will likely shift to a hybrid solution as the `gap` property has achieved nearly full browser support for flexbox.

Key differences

Compared to the default grid system:

- Flex utilities don’t affect the CSS Grid columns in the same way.
- Gaps replaces gutters. The `gap` property replaces the horizontal `padding` from our default grid system and functions more like `margin`.
- As such, unlike `.rows`, `.grids` have no negative margins and margin utilities cannot be used to change the grid gutters. Grid gaps are applied horizontally and vertically by default. See the customizing section for more details.
- Inline and custom styles should be viewed as replacements for modifier classes (e.g., `style="--bs-columns: 3;"` vs `class="row-cols-3"`).
- Nesting works similarly, but may require you to reset your column counts on each instance of a nested `.grid`. See the nesting section for details.

Examples

Three columns

Three equal-width columns across all viewports and devices can be created by using the `.g-col-4` classes. Add responsive classes to change the layout by viewport size.

.g-col-4
.g-col-4
.g-col-4

html

```
<div class="grid">
  <div class="g-col-4">.g-col-4</div>
  <div class="g-col-4">.g-col-4</div>
  <div class="g-col-4">.g-col-4</div>
</div>
```

Responsive

Use responsive classes to adjust your layout across viewports. Here we start with two columns on the narrowest viewports, and then grow to three columns on medium viewports and above.

.g-col-6 .g-col-md-4
.g-col-6 .g-col-md-4
.g-col-6 .g-col-md-4

html

```
<div class="grid">
  <div class="g-col-6 g-col-md-4">.g-col-6 .g-col-md-4</div>
  <div class="g-col-6 g-col-md-4">.g-col-6 .g-col-md-4</div>
  <div class="g-col-6 g-col-md-4">.g-col-6 .g-col-md-4</div>
</div>
```

Compare that to this two column layout at all viewports.

.g-col-6

.g-col-6

html

```
<div class="grid">
  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>
</div>
```

Wrapping

Grid items automatically wrap to the next line when there's no more room horizontally. Note that the `gap` applies to horizontal and vertical gaps between grid items.

.g-col-6

.g-col-6

.g-col-6

.g-col-6

html

```
<div class="grid">
  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>

  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>
</div>
```

Starts

Start classes aim to replace our default grid's offset classes, but they're not entirely the same. CSS Grid creates a grid template through styles that tell browsers to "start at this column" and "end at this column." Those properties are `grid-column-start` and `grid-column-end`. Start classes are shorthand for the former. Pair them with the column classes to size and align your columns however you need. Start classes begin at `1` as `0` is an invalid value for these properties.

.g-col-3 .g-start-2

.g-col-4 .g-start-6

html

```
<div class="grid">
  <div class="g-col-3 g-start-2">.g-col-3 .g-start-2</div>
  <div class="g-col-4 g-start-6">.g-col-4 .g-start-6</div>
</div>
```

Auto columns

When there are no classes on the grid items (the immediate children of a `.grid`), each grid item will automatically be sized to one column.

1
1
1
1
1
1
1
1
1
1
1
1

html

```
<div class="grid">
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
</div>
```

This behavior can be mixed with grid column classes.

.g-col-6
1
1
1
1
1
1

html

```
<div class="grid">
  <div class="g-col-6">.g-col-6</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
</div>
```

Nesting

Similar to our default grid system, our CSS Grid allows for easy nesting of `.grid`s. However, unlike the default, this grid inherits changes in the rows, columns, and gaps. Consider the example below:

- We override the default number of columns with a local CSS variable: `--bs-columns: 3`.
- In the first auto-column, the column count is inherited and each column is one-third of the available width.
- In the second auto-column, we've reset the column count on the nested `.grid` to 12 (our default).
- The third auto-column has no nested content.

In practice this allows for more complex and custom layouts when compared to our default grid system.

First auto-column
Auto-column
Auto-column
Second auto-column
6 of 12
4 of 12
2 of 12
Third auto-column

html

```
<div class="grid" style="--bs-columns: 3;">
  <div>
    First auto-column
    <div class="grid">
      <div>Auto-column</div>
      <div>Auto-column</div>
    </div>
  </div>
  <div>
    Second auto-column
    <div class="grid" style="--bs-columns: 12;">
      <div class="g-col-6">6 of 12</div>
      <div class="g-col-4">4 of 12</div>
      <div class="g-col-2">2 of 12</div>
    </div>
  </div>
  <div>Third auto-column</div>
</div>
```

Customizing

Customize the number of columns, the number of rows, and the width of the gaps with local CSS variables.

Variable	Fallback value	Description
`--bs-rows`	`1`	The number of rows in your grid template

Variable	Fallback value	Description
`--bs-columns`	`12`	The number of columns in your grid template
`--bs-gap`	`1.5rem`	The size of the gap between columns (vertical and horizontal)

These CSS variables have no default value; instead, they apply fallback values that are used *until* a local instance is provided. For example, we use `var(--bs-rows, 1)` for our CSS Grid rows, which ignores `--bs-rows` because that hasn't been set anywhere yet. Once it is, the `.grid` instance will use that value instead of the fallback value of `1`.

No grid classes

Immediate children elements of `.grid` are grid items, so they'll be sized without explicitly adding a `.g-col` class.

Auto-column
Auto-column
Auto-column

html

```
<div class="grid" style="--bs-columns: 3;">
  <div>Auto-column</div>
  <div>Auto-column</div>
  <div>Auto-column</div>
</div>
```

Columns and gaps

Adjust the number of columns and the gap.

.g-col-2
.g-col-2

html

```
<div class="grid" style="--bs-columns: 4; --bs-gap: 5rem;">
  <div class="g-col-2">.g-col-2</div>
  <div class="g-col-2">.g-col-2</div>
</div>
```

.g-col-6
.g-col-4

html

```
<div class="grid" style="--bs-columns: 10; --bs-gap: 1rem;">
  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-4">.g-col-4</div>
</div>
```

Adding rows

Adding more rows and changing the placement of columns:

Auto-column
Auto-column
Auto-column

html

```
<div class="grid" style="--bs-rows: 3; --bs-columns: 3;">
  <div>Auto-column</div>
  <div class="g-start-2" style="grid-row: 2">Auto-column</div>
  <div class="g-start-3" style="grid-row: 3">Auto-column</div>
</div>
```

Gaps

Change the vertical gaps only by modifying the `row-gap`. Note that we use `gap` on `.grid`s, but `row-gap` and `column-gap` can be modified as needed.

.g-col-6

.g-col-6

.g-col-6

.g-col-6

html

```
<div class="grid" style="row-gap: 0;">
  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>

  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>
</div>
```

Because of that, you can have different vertical and horizontal `gap`s, which can take a single value (all sides) or a pair of values (vertical and horizontal). This can be applied with an inline style for `gap`, or with our `--bs-gap` CSS variable.

.g-col-6

.g-col-6

.g-col-6

.g-col-6

html

```
<div class="grid" style="--bs-gap: .25rem 1rem;">
  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>

  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>
</div>
```

Sass

One limitation of the CSS Grid is that our default classes are still generated by two Sass variables, `$grid-columns` and `$grid-gutter-width`. This effectively predetermines the number of classes generated in our compiled CSS. You have two options here:

- Modify those default Sass variables and recompile your CSS.
- Use inline or custom styles to augment the provided classes.

For example, you can increase the column count and change the gap size, and then size your “columns” with a mix of inline styles and predefined CSS Grid column classes (e.g., `.g-col-4`).

14 columns

.g-col-4

html

```html
<div class="grid" style="--bs-columns: 18; --bs-gap: .5rem;">
  <div style="grid-column: span 14;">14 columns</div>
  <div class="g-col-4">.g-col-4</div>
</div>
```

Reboot

Reboot, a collection of element-specific CSS changes in a single file, kickstart Bootstrap to provide an elegant, consistent, and simple baseline to build upon.

On this page

Approach

Reboot builds upon Normalize, providing many HTML elements with somewhat opinionated styles using only element selectors. Additional styling is done only with classes. For example, we reboot some `<table>` styles for a simpler baseline and later provide `.table`, `.table-bordered`, and more.

Here are our guidelines and reasons for choosing what to override in Reboot:

- Update some browser default values to use `rem`s instead of `em`s for scalable component spacing.
- Avoid `margin-top`. Vertical margins can collapse, yielding unexpected results. More importantly though, a single direction of `margin` is a simpler mental model.
- For easier scaling across device sizes, block elements should use `rem`s for `margin`s.
- Keep declarations of `font`-related properties to a minimum, using `inherit` whenever possible.

CSS variables

Added in v5.2.0

With v5.1.1, we standardized our required `@import`s across all our CSS bundles (including `bootstrap.css`, `bootstrap-reboot.css`, and `bootstrap-grid.css`) to include `_root.scss`. This adds `:root` level CSS variables to all bundles, regardless of how many of them are used in that bundle. Ultimately Bootstrap 5 will continue to see more CSS variables added over time, in order to provide more real-time customization without the need to always recompile Sass. Our approach is to take our source Sass variables and transform them into CSS variables. That way, even if you don't use CSS variables, you still have all the power of Sass. **This is still in-progress and will take time to fully implement.**

For example, consider these `:root` CSS variables for common `<body>` styles:

```
  @if $font-size-root != null {
    --#{$prefix}root-font-size: #{$font-size-root};
  }
  --#{$prefix}body-font-family: #{$font-family-base};
  @include rfs($font-size-base, --#{$prefix}body-font-size);
  --#{$prefix}body-font-weight: #{$font-weight-base};
  --#{$prefix}body-line-height: #{$line-height-base};
  --#{$prefix}body-color: #{$body-color};
  @if $body-text-align != null {
    --#{$prefix}body-text-align: #{$body-text-align};
  }
  --#{$prefix}body-bg: #{$body-bg};
```

In practice, those variables are then applied in Reboot like so:

```
body {
  margin: 0; // 1
  font-family: var(--#{$prefix}body-font-family);
  @include font-size(var(--#{$prefix}body-font-size));
  font-weight: var(--#{$prefix}body-font-weight);
  line-height: var(--#{$prefix}body-line-height);
  color: var(--#{$prefix}body-color);
  text-align: var(--#{$prefix}body-text-align);
  background-color: var(--#{$prefix}body-bg); // 2
  -webkit-text-size-adjust: 100%; // 3
  -webkit-tap-highlight-color: rgba($black, 0); // 4
}
```

Which allows you to make real-time customizations however you like:

```
<body style="--bs-body-color: #333;">
  <!-- ... -->
</body>
```

Page defaults

The `<html>` and `<body>` elements are updated to provide better page-wide defaults. More specifically:

- The `box-sizing` is globally set on every element—including `*::before` and `*::after`, to `border-box`. This ensures that the declared width of element is never exceeded due to padding or border.
    - No base `font-size` is declared on the `<html>`, but `16px` is assumed (the browser default). `font-size: 1rem` is applied on the `<body>` for easy responsive type-scaling via media queries while respecting user preferences and ensuring a more accessible approach. This browser default can be overridden by modifying the `$font-size-root` variable.
- The `<body>` also sets a global `font-family`, `font-weight`, `line-height`, and `color`. This is inherited later by some form elements to prevent font inconsistencies.
- For safety, the `<body>` has a declared `background-color`, defaulting to `#fff`.

Native font stack

Bootstrap utilizes a “native font stack” or “system font stack” for optimum text rendering on every device and OS. These system fonts have been designed specifically with today’s devices in mind, with improved rendering on screens, variable font support, and more. Read more about native font stacks in this *Smashing Magazine* article.

```
$font-family-sans-serif:
  // Cross-platform generic font family (default user interface font)
  system-ui,
  // Safari for macOS and iOS (San Francisco)
  -apple-system,
  // Windows
  "Segoe UI",
  // Android
  Roboto,
  // older macOS and iOS
  "Helvetica Neue"
  // Linux
  "Noto Sans",
  "Liberation Sans",
  // Basic web fallback
  Arial,
  // Sans serif fallback
  sans-serif,
  // Emoji fonts
  "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji" !
default;
```

Note that because the font stack includes emoji fonts, many common symbol/dingbat unicode characters will be rendered as multi-colored pictographs. Their appearance will vary, depending on the style used in the browser/platform’s native emoji font, and they won’t be affected by any CSS `color` styles.

This `font-family` is applied to the `<body>` and automatically inherited globally throughout Bootstrap. To switch the global `font-family`, update `$font-family-base` and recompile Bootstrap.

Headings and paragraphs

All heading elements—e.g., `<h1>`—and `<p>` are reset to have their `margin-top` removed. Headings have `margin-bottom: .5rem` added and paragraphs `margin-bottom: 1rem` for easy spacing.

Heading	Example
`<h1></h1>`	h1. Bootstrap heading
`<h2></h2>`	h2. Bootstrap heading
`<h3></h3>`	h3. Bootstrap heading
`<h4></h4>`	h4. Bootstrap heading
`<h5></h5>`	h5. Bootstrap heading
`<h6></h6>`	h6. Bootstrap heading

Horizontal rules

The `<hr>` element has been simplified. Similar to browser defaults, `<hr>`s are styled via `border-top`, have a default `opacity: .25`, and automatically inherit their `border-color` via `color`, including when `color` is set via the parent. They can be modified with text, border, and opacity utilities.

html

```
<hr>

<div class="text-success">
  <hr>
</div>

<hr class="text-danger border-2 opacity-50">
<hr class="border-primary border-3 opacity-75">
```

Lists

All lists—`<ul>`, `<ol>`, and `<dl>`—have their `margin-top` removed and a `margin-bottom: 1rem`. Nested lists have no `margin-bottom`. We've also reset the `padding-left` on `<ul>` and `<ol>` elements.

- All lists have their top margin removed
- And their bottom margin normalized
- Nested lists have no bottom margin
    - This way they have a more even appearance
    - Particularly when followed by more list items
- The left padding has also been reset

1. Here’s an ordered list
2. With a few list items
3. It has the same overall look
4. As the previous unordered list

For simpler styling, clear hierarchy, and better spacing, description lists have updated `margin`s. `<dd>`s reset `margin-left` to `0` and add `margin-bottom: .5rem`. `<dt>`s are **bolded**.

Description lists
    A description list is perfect for defining terms.
Term
    Definition for the term.
    A second definition for the same term.
Another term
    Definition for this other term.

Inline code

Wrap inline snippets of code with `<code>`. Be sure to escape HTML angle brackets.

For example, `<section>` should be wrapped as inline.

html

```
For example, <code>&lt;section&gt;</code> should be wrapped as inline.
```

Code blocks

Use `<pre>`s for multiple lines of code. Once again, be sure to escape any angle brackets in the code for proper rendering. The `<pre>` element is reset to remove its `margin-top` and use `rem` units for its `margin-bottom`.

```
<p>Sample text here...</p>
<p>And another line of sample text here...</p>
```

html

```
<pre><code>&lt;p&gt;Sample text here...&lt;/p&gt;
&lt;p&gt;And another line of sample text here...&lt;/p&gt;
</code></pre>
```

Variables

For indicating variables use the `<var>` tag.

$y = mx + b$

html

```
<var>y</var> = <var>m</var><var>x</var> + <var>b</var>
```

User input

Use the `<kbd>` to indicate input that is typically entered via keyboard.

To switch directories, type `cd` followed by the name of the directory.
To edit settings, press `ctrl + ,`

html

```
To switch directories, type <kbd>cd</kbd> followed by the name of the
directory.<br>
To edit settings, press <kbd><kbd>ctrl</kbd> + <kbd>,</kbd></kbd>
```

Sample output

For indicating sample output from a program use the `<samp>` tag.

```
This text is meant to be treated as sample output from a computer
program.
```

html

```
<samp>This text is meant to be treated as sample output from a computer
program.</samp>
```

Tables

Tables are slightly adjusted to style `<caption>`s, collapse borders, and ensure consistent `text-align` throughout. Additional changes for borders, padding, and more come with [the `.table` class](#).

This is an example table, and this is its caption to describe the contents.

Table heading	Table heading	Table heading	Table heading
Table cell	Table cell	Table cell	Table cell
Table cell	Table cell	Table cell	Table cell
Table cell	Table cell	Table cell	Table cell

html

```
<table>
  <caption>
    This is an example table, and this is its caption to describe the contents.
  </caption>
  <thead>
    <tr>
      <th>Table heading</th>
      <th>Table heading</th>
      <th>Table heading</th>
      <th>Table heading</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Table cell</td>
      <td>Table cell</td>
      <td>Table cell</td>
      <td>Table cell</td>
    </tr>
    <tr>
      <td>Table cell</td>
      <td>Table cell</td>
      <td>Table cell</td>
      <td>Table cell</td>
    </tr>
    <tr>
      <td>Table cell</td>
      <td>Table cell</td>
      <td>Table cell</td>
      <td>Table cell</td>
    </tr>
  </tbody>
</table>
```

Forms

Various form elements have been rebooted for simpler base styles. Here are some of the most notable changes:

- `<fieldset>`s have no borders, padding, or margin so they can be easily used as wrappers for individual inputs or groups of inputs.
- `<legend>`s, like fieldsets, have also been restyled to be displayed as a heading of sorts.
- `<label>`s are set to `display: inline-block` to allow `margin` to be applied.
- `<input>`s, `<select>`s, `<textarea>`s, and `<button>`s are mostly addressed by Normalize, but Reboot removes their `margin` and sets `line-height: inherit`, too.
- `<textarea>`s are modified to only be resizable vertically as horizontal resizing often "breaks" page layout.
- `<button>`s and `<input>` button elements have `cursor: pointer` when `:not(:disabled)`.

These changes, and more, are demonstrated below.

Check this checkbox

Option one is this and that Option two is something else that's also super long to demonstrate the wrapping of these fancy form controls. Option three is disabled

Date & color input support

Keep in mind date inputs are not fully supported by all browsers, namely Safari.

Pointers on buttons

Reboot includes an enhancement for `role="button"` to change the default cursor to `pointer`. Add this attribute to elements to help indicate elements are interactive. This role isn't necessary for `<button>` elements, which get their own `cursor` change.

Non-button element button

html

```
<span role="button" tabindex="0">Non-button element button</span>
```

Misc elements

Address

The `<address>` element is updated to reset the browser default `font-style` from `italic` to `normal`. `line-height` is also now inherited, and `margin-bottom: 1rem` has been added. `<address>`s are for presenting contact information for the nearest ancestor (or an entire body of work). Preserve formatting by ending lines with `<br>`.

Twitter, Inc.
1355 Market St, Suite 900
San Francisco, CA 94103
(123) 456-7890
Full Name
first.last@example.com

Blockquote

The default `margin` on blockquotes is `1em 40px`, so we reset that to `0 0 1rem` for something more consistent with other elements.

> A well-known quote, contained in a blockquote element.

Someone famous in *Source Title*

Inline elements

The `<abbr>` element receives basic styling to make it stand out amongst paragraph text.

The abbreviation element.

Summary

The default `cursor` on summary is `text`, so we reset that to `pointer` to convey that the element can be interacted with by clicking on it.

Here are even more details about the details.

HTML5 `[hidden]` attribute

HTML5 adds a new global attribute named `[hidden]`, which is styled as `display: none` by default. Borrowing an idea from PureCSS, we improve upon this default by making `[hidden] { display: none !important; }` to help prevent its `display` from getting accidentally overridden.

```
<input type="text" hidden>
```

jQuery incompatibility

`[hidden]` is not compatible with jQuery's `$(...).hide()` and `$(...).show()` methods. Therefore, we don't currently especially endorse `[hidden]` over other techniques for managing the `display` of elements.

To merely toggle the visibility of an element, meaning its `display` is not modified and the element can still affect the flow of the document, use the `.invisible` class instead.

Typography

Documentation and examples for Bootstrap typography, including global settings, headings, body text, lists, and more.

On this page

Global settings

Bootstrap sets basic global display, typography, and link styles. When more control is needed, check out the textual utility classes.

- Use a native font stack that selects the best `font-family` for each OS and device.
- For a more inclusive and accessible type scale, we use the browser's default root `font-size` (typically 16px) so visitors can customize their browser defaults as needed.
- Use the `$font-family-base`, `$font-size-base`, and `$line-height-base` attributes as our typographic base applied to the `<body>`.
- Set the global link color via `$link-color`.
- Use `$body-bg` to set a `background-color` on the `<body>` (`#fff` by default).

These styles can be found within `_reboot.scss`, and the global variables are defined in `_variables.scss`. Make sure to set `$font-size-base` in `rem`.

Headings

All HTML headings, `<h1>` through `<h6>`, are available.

Heading	Example
`<h1></h1>`	h1. Bootstrap heading
`<h2></h2>`	h2. Bootstrap heading
`<h3></h3>`	h3. Bootstrap heading
`<h4></h4>`	h4. Bootstrap heading
`<h5></h5>`	h5. Bootstrap heading
`<h6></h6>`	h6. Bootstrap heading

```
<h1>h1. Bootstrap heading</h1>
<h2>h2. Bootstrap heading</h2>
<h3>h3. Bootstrap heading</h3>
<h4>h4. Bootstrap heading</h4>
<h5>h5. Bootstrap heading</h5>
<h6>h6. Bootstrap heading</h6>
```

`.h1` through `.h6` classes are also available, for when you want to match the font styling of a heading but cannot use the associated HTML element.

h1. Bootstrap heading

h2. Bootstrap heading

h3. Bootstrap heading

h4. Bootstrap heading

h5. Bootstrap heading

h6. Bootstrap heading

html

```
<p class="h1">h1. Bootstrap heading</p>
<p class="h2">h2. Bootstrap heading</p>
<p class="h3">h3. Bootstrap heading</p>
<p class="h4">h4. Bootstrap heading</p>
<p class="h5">h5. Bootstrap heading</p>
<p class="h6">h6. Bootstrap heading</p>
```

Customizing headings

Use the included utility classes to recreate the small secondary heading text from Bootstrap 3.

Fancy display heading With faded secondary text

html

```
<h3>
  Fancy display heading
  <small class="text-muted">With faded secondary text</small>
</h3>
```

Display headings

Traditional heading elements are designed to work best in the meat of your page content. When you need a heading to stand out, consider using a **display heading**—a larger, slightly more opinionated heading style.

Display 1
Display 2
Display 3
Display 4
Display 5
Display 6

```
<h1 class="display-1">Display 1</h1>
<h1 class="display-2">Display 2</h1>
<h1 class="display-3">Display 3</h1>
<h1 class="display-4">Display 4</h1>
<h1 class="display-5">Display 5</h1>
<h1 class="display-6">Display 6</h1>
```

Display headings are configured via the `$display-font-sizes` Sass map and two variables, `$display-font-weight` and `$display-line-height`.

```
$display-font-sizes: (
  1: 5rem,
  2: 4.5rem,
  3: 4rem,
  4: 3.5rem,
  5: 3rem,
  6: 2.5rem
);

$display-font-weight: 300;
$display-line-height: $headings-line-height;
```

Lead

Make a paragraph stand out by adding `.lead`.

This is a lead paragraph. It stands out from regular paragraphs.

html

```
<p class="lead">
  This is a lead paragraph. It stands out from regular paragraphs.
</p>
```

Inline text elements

Styling for common inline HTML5 elements.

You can use the mark tag to text.

This line of text is meant to be treated as deleted text.

~~This line of text is meant to be treated as no longer accurate.~~

This line of text is meant to be treated as an addition to the document.

This line of text will render as underlined.

This line of text is meant to be treated as fine print.

This line rendered as bold text.

This line rendered as italicized text.

html

```
<p>You can use the mark tag to <mark>highlight</mark> text.</p>
<p><del>This line of text is meant to be treated as deleted text.</del></p>
<p><s>This line of text is meant to be treated as no longer accurate.</s></p>
<p><ins>This line of text is meant to be treated as an addition to the
document.</ins></p>
<p><u>This line of text will render as underlined.</u></p>
<p><small>This line of text is meant to be treated as fine print.</small></p>
<p><strong>This line rendered as bold text.</strong></p>
<p><em>This line rendered as italicized text.</em></p>
```

Beware that those tags should be used for semantic purpose:

- `<mark>` represents text which is marked or highlighted for reference or notation purposes.
- `<small>` represents side-comments and small print, like copyright and legal text.
- `<s>` represents element that are no longer relevant or no longer accurate.
- `<u>` represents a span of inline text which should be rendered in a way that indicates that it has a non-textual annotation.

If you want to style your text, you should use the following classes instead:

- `.mark` will apply the same styles as `<mark>`.
- `.small` will apply the same styles as `<small>`.
- `.text-decoration-underline` will apply the same styles as `<u>`.
- `.text-decoration-line-through` will apply the same styles as `<s>`.

While not shown above, feel free to use `<b>` and `<i>` in HTML5. `<b>` is meant to highlight words or phrases without conveying additional importance, while `<i>` is mostly for voice, technical terms, etc.

Text utilities

Change text alignment, transform, style, weight, line-height, decoration and color with our text utilities and color utilities.

Abbreviations

Stylized implementation of HTML's `<abbr>` element for abbreviations and acronyms to show the expanded version on hover. Abbreviations have a default underline and gain a help cursor to provide additional context on hover and to users of assistive technologies.

Add `.initialism` to an abbreviation for a slightly smaller font-size.

html

```
<p><abbr title="attribute">attr</abbr></p>
<p><abbr title="HyperText Markup Language" class="initialism">HTML</abbr></p>
```

Blockquotes

For quoting blocks of content from another source within your document. Wrap `<blockquote class="blockquote">` around any HTML as the quote.

> A well-known quote, contained in a blockquote element.

html

```
<blockquote class="blockquote">
  <p>A well-known quote, contained in a blockquote element.</p>
</blockquote>
```

Naming a source

The HTML spec requires that blockquote attribution be placed outside the `<blockquote>`. When providing attribution, wrap your `<blockquote>` in a `<figure>` and use a `<figcaption>` or a block level element (e.g., `<p>`) with the `.blockquote-footer` class. Be sure to wrap the name of the source work in `<cite>` as well.

> A well-known quote, contained in a blockquote element.

Source Title

html

```
<figure>
  <blockquote class="blockquote">
    <p>A well-known quote, contained in a blockquote element.</p>
  </blockquote>
  <figcaption class="blockquote-footer">
    Someone famous in <cite title="Source Title">Source Title</cite>
  </figcaption>
</figure>
```

Alignment

Use text utilities as needed to change the alignment of your blockquote.

> A well-known quote, contained in a blockquote element.

Source Title

html

```
<figure class="text-center">
  <blockquote class="blockquote">
    <p>A well-known quote, contained in a blockquote element.</p>
  </blockquote>
  <figcaption class="blockquote-footer">
    Someone famous in <cite title="Source Title">Source Title</cite>
  </figcaption>
</figure>
```

> A well-known quote, contained in a blockquote element.

Source Title

html

```
<figure class="text-end">
  <blockquote class="blockquote">
    <p>A well-known quote, contained in a blockquote element.</p>
  </blockquote>
  <figcaption class="blockquote-footer">
    Someone famous in <cite title="Source Title">Source Title</cite>
  </figcaption>
</figure>
```

Lists

Unstyled

Remove the default `list-style` and left margin on list items (immediate children only). **This only applies to immediate children list items**, meaning you will need to add the class for any nested lists as well.

- This is a list.
- It appears completely unstyled.
- Structurally, it's still a list.
- However, this style only applies to immediate child elements.
- Nested lists:
    - are unaffected by this style
    - will still show a bullet
    - and have appropriate left margin
- This may still come in handy in some situations.

html

```
<ul class="list-unstyled">
  <li>This is a list.</li>
  <li>It appears completely unstyled.</li>
  <li>Structurally, it's still a list.</li>
  <li>However, this style only applies to immediate child elements.</li>
  <li>Nested lists:
    <ul>
      <li>are unaffected by this style</li>
      <li>will still show a bullet</li>
      <li>and have appropriate left margin</li>
    </ul>
  </li>
  <li>This may still come in handy in some situations.</li>
</ul>
```

Inline

Remove a list’s bullets and apply some light `margin` with a combination of two classes, `.list-inline` and `.list-inline-item`.

- This is a list item.
- And another one.
- But they're displayed inline.

html

```
<ul class="list-inline">
  <li class="list-inline-item">This is a list item.</li>
  <li class="list-inline-item">And another one.</li>
  <li class="list-inline-item">But they're displayed inline.</li>
</ul>
```

Description list alignment

Align terms and descriptions horizontally by using our grid system's predefined classes (or semantic mixins). For longer terms, you can optionally add a `.text-truncate` class to truncate the text with an ellipsis.

Description lists
    A description list is perfect for defining terms.
Term

    Definition for the term.

    And some more placeholder definition text.

Another term
    This definition is short, so no extra paragraphs or anything.
Truncated term is truncated
    This can be useful when space is tight. Adds an ellipsis at the end.
Nesting
    Nested definition list
        I heard you like definition lists. Let me put a definition list inside your definition list.

html

```
<dl class="row">
  <dt class="col-sm-3">Description lists</dt>
  <dd class="col-sm-9">A description list is perfect for defining terms.</dd>

  <dt class="col-sm-3">Term</dt>
  <dd class="col-sm-9">
    <p>Definition for the term.</p>
    <p>And some more placeholder definition text.</p>
  </dd>

  <dt class="col-sm-3">Another term</dt>
  <dd class="col-sm-9">This definition is short, so no extra paragraphs or 
anything.</dd>

  <dt class="col-sm-3 text-truncate">Truncated term is truncated</dt>
  <dd class="col-sm-9">This can be useful when space is tight. Adds an ellipsis 
at the end.</dd>

  <dt class="col-sm-3">Nesting</dt>
  <dd class="col-sm-9">
    <dl class="row">
      <dt class="col-sm-4">Nested definition list</dt>
      <dd class="col-sm-8">I heard you like definition lists. Let me put a 
definition list inside your definition list.</dd>
    </dl>
  </dd>
</dl>
```

Responsive font sizes

In Bootstrap 5, we’ve enabled responsive font sizes by default, allowing text to scale more naturally across device and viewport sizes. Have a look at the RFS page to find out how this works.

Sass

Variables

Headings have some dedicated variables for sizing and spacing.

```
$headings-margin-bottom:      $spacer * .5;
$headings-font-family:        null;
$headings-font-style:         null;
$headings-font-weight:        500;
$headings-line-height:        1.2;
$headings-color:              null;
```

Miscellaneous typography elements covered here and in Reboot also have dedicated variables.

```
$lead-font-size:              $font-size-base * 1.25;
$lead-font-weight:            300;

$small-font-size:             .875em;

$sub-sup-font-size:           .75em;

$text-muted:                  rgba(var(--#{$prefix}body-color-rgb), .75);

$initialism-font-size:        $small-font-size;

$blockquote-margin-y:         $spacer;
$blockquote-font-size:        $font-size-base * 1.25;
$blockquote-footer-color:     $gray-600;
$blockquote-footer-font-size: $small-font-size;

$hr-margin-y:                 $spacer;
$hr-color:                    inherit;

// fusv-disable
$hr-bg-color:                 null; // Deprecated in v5.2.0
$hr-height:                   null; // Deprecated in v5.2.0
// fusv-enable

$hr-border-color:             null; // Allows for inherited colors
$hr-border-width:             $border-width;
$hr-opacity:                  .25;

$legend-margin-bottom:        .5rem;
$legend-font-size:            1.5rem;
$legend-font-weight:          null;

$dt-font-weight:              $font-weight-bold;

$list-inline-padding:         .5rem;

$mark-padding:                .1875em;
$mark-bg:                     $yellow-100;
```

Mixins

There are no dedicated mixins for typography, but Bootstrap does use Responsive Font Sizing (RFS).

Images

Documentation and examples for opting images into responsive behavior (so they never become wider than their parent) and add lightweight styles to them—all via classes.

On this page

Responsive images

Images in Bootstrap are made responsive with `.img-fluid`. This applies `max-width: 100%;` and `height: auto;` to the image so that it scales with the parent width.

html

```
<img src="..." class="img-fluid" alt="...">
```

Image thumbnails

In addition to our border-radius utilities, you can use `.img-thumbnail` to give an image a rounded 1px border appearance.

html

```
<img src="..." class="img-thumbnail" alt="...">
```

Aligning images

Align images with the helper float classes or text alignment classes. `block`-level images can be centered using the `.mx-auto` margin utility class.

html

```
<img src="..." class="rounded float-start" alt="...">
<img src="..." class="rounded float-end" alt="...">
```

html

```
<img src="..." class="rounded mx-auto d-block" alt="...">
```

html

```
<div class="text-center">
  <img src="..." class="rounded" alt="...">
</div>
```

Picture

If you are using the `<picture>` element to specify multiple `<source>` elements for a specific `<img>`, make sure to add the `.img-*` classes to the `<img>` and not to the `<picture>` tag.

```
<picture>
  <source srcset="..." type="image/svg+xml">
  <img src="..." class="img-fluid img-thumbnail" alt="...">
</picture>
```

Sass

Variables

Variables are available for image thumbnails.

```
$thumbnail-padding:                 .25rem;
$thumbnail-bg:                      $body-bg;
$thumbnail-border-width:            $border-width;
$thumbnail-border-color:            var(--#{$prefix}border-color);
$thumbnail-border-radius:           $border-radius;
$thumbnail-box-shadow:              $box-shadow-sm;
```

Tables

Documentation and examples for opt-in styling of tables (given their prevalent use in JavaScript plugins) with Bootstrap.

On this page

Overview

Due to the widespread use of `<table>` elements across third-party widgets like calendars and date pickers, Bootstrap's tables are **opt-in**. Add the base class `.table` to any `<table>`, then extend with our optional modifier classes or custom styles. All table styles are not inherited in Bootstrap, meaning any nested tables can be styled independent from the parent.

Using the most basic table markup, here's how `.table`-based tables look in Bootstrap.

#	First	Last	Handle
1	Mark	Otto	@mdo

#	First	Last	Handle
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">First</th>
      <th scope="col">Last</th>
      <th scope="col">Handle</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
      <td>Mark</td>
      <td>Otto</td>
      <td>@mdo</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Jacob</td>
      <td>Thornton</td>
      <td>@fat</td>
    </tr>
    <tr>
      <th scope="row">3</th>
      <td colspan="2">Larry the Bird</td>
      <td>@twitter</td>
    </tr>
  </tbody>
</table>
```

Variants

Use contextual classes to color tables, table rows or individual cells.

Class	Heading	Heading
Default	Cell	Cell
Primary	Cell	Cell
Secondary	Cell	Cell
Success	Cell	Cell
Danger	Cell	Cell
Warning	Cell	Cell
Info	Cell	Cell
Light	Cell	Cell
Dark	Cell	Cell

```
<!-- On tables -->
<table class="table-primary">...</table>
<table class="table-secondary">...</table>
<table class="table-success">...</table>
<table class="table-danger">...</table>
<table class="table-warning">...</table>
<table class="table-info">...</table>
<table class="table-light">...</table>
<table class="table-dark">...</table>
```

```
<!-- On rows -->
<tr class="table-primary">...</tr>
<tr class="table-secondary">...</tr>
<tr class="table-success">...</tr>
<tr class="table-danger">...</tr>
<tr class="table-warning">...</tr>
<tr class="table-info">...</tr>
<tr class="table-light">...</tr>
<tr class="table-dark">...</tr>

<!-- On cells (`td` or `th`) -->
<tr>
  <td class="table-primary">...</td>
  <td class="table-secondary">...</td>
  <td class="table-success">...</td>
  <td class="table-danger">...</td>
  <td class="table-warning">...</td>
  <td class="table-info">...</td>
  <td class="table-light">...</td>
  <td class="table-dark">...</td>
</tr>
```

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.

Accented tables

Striped rows

Use `.table-striped` to add zebra-striping to any table row within the `<tbody>`.

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-striped">
  ...
</table>
```

Striped columns

Use `.table-striped-columns` to add zebra-striping to any table column.

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-striped-columns">
  ...
</table>
```

These classes can also be added to table variants:

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-dark table-striped">
  ...
</table>
```

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-dark table-striped-columns">
  ...
</table>
```

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-success table-striped">
  ...
</table>
```

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-success table-striped-columns">
  ...
</table>
```

Hoverable rows

Add `.table-hover` to enable a hover state on table rows within a `<tbody>`.

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-hover">
  ...
</table>
```

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-dark table-hover">
```

```
  ...
</table>
```

These hoverable rows can also be combined with the striped rows variant:

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-striped table-hover">
  ...
</table>
```

Active tables

Highlight a table row or cell by adding a `.table-active` class.

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table">
  <thead>
    ...
  </thead>
  <tbody>
    <tr class="table-active">
      ...
    </tr>
    <tr>
      ...
    </tr>
    <tr>
      <th scope="row">3</th>
      <td colspan="2" class="table-active">Larry the Bird</td>
      <td>@twitter</td>
    </tr>
  </tbody>
</table>
```

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-dark">
  <thead>
    ...
  </thead>
  <tbody>
    <tr class="table-active">
      ...
    </tr>
    <tr>
      ...
    </tr>
    <tr>
      <th scope="row">3</th>
      <td colspan="2" class="table-active">Larry the Bird</td>
      <td>@twitter</td>
    </tr>
  </tbody>
</table>
```

How do the variants and accented tables work?

For the accented tables (striped rows, striped columns, hoverable rows, and active tables), we used some techniques to make these effects work for all our table variants:

- We start by setting the background of a table cell with the `--bs-table-bg` custom property. All table variants then set that custom property to colorize the table cells. This way, we don't get into trouble if semi-transparent colors are used as table backgrounds.
- Then we add an inset box shadow on the table cells with `box-shadow: inset 0 0 0 9999px var(--bs-table-accent-bg);` to layer on top of any specified `background-color`. Because we use a huge spread and no blur, the color will be monotone. Since `--bs-table-accent-bg` is unset by default, we don't have a default box shadow.
- When either `.table-striped`, `.table-striped-columns`, `.table-hover` or `.table-active` classes are added, the `--bs-table-accent-bg` is set to a semitransparent color to colorize the background.
- For each table variant, we generate a `--bs-table-accent-bg` color with the highest contrast depending on that color. For example, the accent color for `.table-primary` is darker while `.table-dark` has a lighter accent color.
- Text and border colors are generated the same way, and their colors are inherited by default.

Behind the scenes it looks like this:

```
@mixin table-variant($state, $background) {
  .table-#{$state} {
    $color: color-contrast(opaque($body-bg, $background));
    $hover-bg: mix($color, $background, percentage($table-hover-bg-factor));
    $striped-bg: mix($color, $background, percentage($table-striped-bg-factor));
    $active-bg: mix($color, $background, percentage($table-active-bg-factor));
    $border-color: mix($color, $background, percentage($table-border-factor));

    --#{$prefix}table-color: #{$color};
    --#{$prefix}table-bg: #{$background};
    --#{$prefix}table-border-color: #{$border-color};
    --#{$prefix}table-striped-bg: #{$striped-bg};
    --#{$prefix}table-striped-color: #{color-contrast($striped-bg)};
    --#{$prefix}table-active-bg: #{$active-bg};
    --#{$prefix}table-active-color: #{color-contrast($active-bg)};
    --#{$prefix}table-hover-bg: #{$hover-bg};
    --#{$prefix}table-hover-color: #{color-contrast($hover-bg)};

    color: var(--#{$prefix}table-color);
    border-color: var(--#{$prefix}table-border-color);
  }
}
```

Table borders

Bordered tables

Add `.table-bordered` for borders on all sides of the table and cells.

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-bordered">
  ...
</table>
```

Border color utilities can be added to change colors:

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-bordered border-primary">
  ...
</table>
```

Tables without borders

Add `.table-borderless` for a table without borders.

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-borderless">
  ...
</table>
```

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-dark table-borderless">
  ...
</table>
```

Small tables

Add `.table-sm` to make any `.table` more compact by cutting all cell `padding` in half.

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat

#	First	Last	Handle
3	Larry the Bird		@twitter

```
<table class="table table-sm">
  ...
</table>
```

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-dark table-sm">
  ...
</table>
```

Table group dividers

Add a thicker border, darker between table groups—`<thead>`, `<tbody>`, and `<tfoot>`—with `.table-group-divider`. Customize the color by changing the `border-top-color` (which we don't currently provide a utility class for at this time).

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

html

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">First</th>
      <th scope="col">Last</th>
      <th scope="col">Handle</th>
    </tr>
  </thead>
  <tbody class="table-group-divider">
    <tr>
      <th scope="row">1</th>
      <td>Mark</td>
      <td>Otto</td>
      <td>@mdo</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Jacob</td>
      <td>Thornton</td>
      <td>@fat</td>
    </tr>
    <tr>
      <th scope="row">3</th>
      <td colspan="2">Larry the Bird</td>
      <td>@twitter</td>
    </tr>
  </tbody>
</table>
```

Vertical alignment

Table cells of `<thead>` are always vertical aligned to the bottom. Table cells in `<tbody>` inherit their alignment from `<table>` and are aligned to the top by default. Use the vertical align classes to re-align where needed.

Heading 1	Heading 2	Heading 3	Heading 4
This cell inherits `vertical-align: middle;` from the table	This cell inherits `vertical-align: middle;` from the table	This cell inherits `vertical-align: middle;` from the table	This here is some placeholder text, intended to take up quite a bit of vertical space, to demonstrate how the vertical alignment works in the preceding cells.
This cell inherits `vertical-align: bottom;` from the table row	This cell inherits `vertical-align: bottom;` from the table row	This cell inherits `vertical-align: bottom;` from the table row	This here is some placeholder text, intended to take up quite a bit of vertical space, to demonstrate how the vertical alignment works in the preceding cells.
This cell inherits `vertical-align: middle;` from the table	This cell inherits `vertical-align: middle;` from the table	This cell is aligned to the top.	This here is some placeholder text, intended to take up quite a bit of vertical space, to demonstrate how the vertical alignment works in the preceding cells.

```
<div class="table-responsive">
  <table class="table align-middle">
    <thead>
      <tr>
        ...
      </tr>
    </thead>
    <tbody>
      <tr>
        ...
      </tr>
      <tr class="align-bottom">
        ...
      </tr>
      <tr>
        <td>...</td>
        <td>...</td>
        <td class="align-top">This cell is aligned to the top.</td>
        <td>...</td>
      </tr>
    </tbody>
  </table>
</div>
```

Nesting

Border styles, active styles, and table variants are not inherited by nested tables.

#	First	Last	Handle
1	Mark	Otto	@mdo
3	Larry	the Bird	@twitter

Header	Header	Header
A	First	Last
B	First	Last
C	First	Last

```
<table class="table table-striped">
  <thead>
    ...
  </thead>
  <tbody>
    ...
    <tr>
      <td colspan="4">
        <table class="table mb-0">
          ...
        </table>
      </td>
    </tr>
    ...
  </tbody>
</table>
```

How nesting works

To prevent *any* styles from leaking to nested tables, we use the child combinator (`>`) selector in our CSS. Since we need to target all the `td`s and `th`s in the `thead`, `tbody`, and `tfoot`, our selector would look pretty long without it. As such, we use the rather odd looking `.table > :not(caption) > * > *` selector to target all `td`s and `th`s of the `.table`, but none of any potential nested tables.

Note that if you add `<tr>`s as direct children of a table, those `<tr>` will be wrapped in a `<tbody>` by default, thus making our selectors work as intended.

Anatomy

Table head

Similar to tables and dark tables, use the modifier classes `.table-light` or `.table-dark` to make `<thead>`s appear light or dark gray.

#	First	Last	Handle
1	Mark	Otto	@mdo

#	First	Last	Handle
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

```
<table class="table">
  <thead class="table-light">
    ...
  </thead>
  <tbody>
    ...
  </tbody>
</table>
```

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

```
<table class="table">
  <thead class="table-dark">
    ...
  </thead>
  <tbody>
    ...
  </tbody>
</table>
```

Table foot

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter
Footer	Footer	Footer	Footer

```
<table class="table">
  <thead>
    ...
  </thead>
  <tbody>
    ...
  </tbody>
  <tfoot>
    ...
  </tfoot>
</table>
```

Captions

A `<caption>` functions like a heading for a table. It helps users with screen readers to find a table and understand what it’s about and decide if they want to read it.

List of users

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-sm">
  <caption>List of users</caption>
  <thead>
    ...
  </thead>
  <tbody>
    ...
  </tbody>
</table>
```

You can also put the `<caption>` on the top of the table with `.caption-top`.

List of users

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

html

```
<table class="table caption-top">
  <caption>List of users</caption>
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">First</th>
      <th scope="col">Last</th>
      <th scope="col">Handle</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
      <td>Mark</td>
      <td>Otto</td>
      <td>@mdo</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Jacob</td>
      <td>Thornton</td>
      <td>@fat</td>
    </tr>
    <tr>
      <th scope="row">3</th>
      <td>Larry</td>
      <td>the Bird</td>
      <td>@twitter</td>
    </tr>
  </tbody>
</table>
```

Responsive tables

Responsive tables allow tables to be scrolled horizontally with ease. Make any table responsive across all viewports by wrapping a `.table` with `.table-responsive`. Or, pick a maximum breakpoint with which to have a responsive table up to by using `.table-responsive{-sm|-md|-lg|-xl|-xxl}`.

Vertical clipping/truncation

Responsive tables make use of `overflow-y: hidden`, which clips off any content that goes beyond the bottom or top edges of the table. In particular, this can clip off dropdown menus and other third-party widgets.

Always responsive

Across every breakpoint, use `.table-responsive` for horizontally scrolling tables.

#	Heading	Heading	Heading	Heading	Heading	Heading	Heading	Heading	Heading
1	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
2	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
3	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell

```
<div class="table-responsive">
  <table class="table">
    ...
  </table>
</div>
```

Breakpoint specific

Use `.table-responsive{-sm|-md|-lg|-xl|-xxl}` as needed to create responsive tables up to a particular breakpoint. From that breakpoint and up, the table will behave normally and not scroll horizontally.

These tables may appear broken until their responsive styles apply at specific viewport widths.

#	Heading	Heading	Heading	Heading	Heading	Heading	Heading	Heading
1	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
2	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
3	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell

#	Heading	Heading	Heading	Heading	Heading	Heading	Heading	Heading
1	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
2	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
3	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell

#	Heading	Heading	Heading	Heading	Heading	Heading	Heading	Heading
1	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
2	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
3	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell

#	Heading	Heading	Heading	Heading	Heading	Heading	Heading	Heading
1	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
2	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
3	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell

#	Heading	Heading	Heading	Heading	Heading	Heading	Heading	Heading
1	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
2	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
3	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell

#	Heading	Heading	Heading	Heading	Heading	Heading	Heading	Heading
1	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
2	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
3	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell

```
<div class="table-responsive">
  <table class="table">
    ...
  </table>
</div>

<div class="table-responsive-sm">
  <table class="table">
    ...
  </table>
</div>

<div class="table-responsive-md">
  <table class="table">
    ...
  </table>
</div>

<div class="table-responsive-lg">
  <table class="table">
    ...
  </table>
</div>

<div class="table-responsive-xl">
  <table class="table">
    ...
  </table>
</div>

<div class="table-responsive-xxl">
  <table class="table">
    ...
  </table>
</div>
```

Sass

Variables

```
$table-cell-padding-y:        .5rem;
$table-cell-padding-x:        .5rem;
$table-cell-padding-y-sm:     .25rem;
$table-cell-padding-x-sm:     .25rem;

$table-cell-vertical-align:   top;

$table-color:                 var(--#{$prefix}body-color);
$table-bg:                    transparent;
$table-accent-bg:             transparent;

$table-th-font-weight:        null;

$table-striped-color:         $table-color;
$table-striped-bg-factor:     .05;
$table-striped-bg:            rgba($black, $table-striped-bg-factor);
```

```
$table-active-color:          $table-color;
$table-active-bg-factor:      .1;
$table-active-bg:             rgba($black, $table-active-bg-factor);

$table-hover-color:           $table-color;
$table-hover-bg-factor:       .075;
$table-hover-bg:              rgba($black, $table-hover-bg-factor);

$table-border-factor:         .1;
$table-border-width:          $border-width;
$table-border-color:          var(--#{$prefix}border-color);

$table-striped-order:         odd;
$table-striped-columns-order: even;

$table-group-separator-color: currentcolor;

$table-caption-color:         $text-muted;

$table-bg-scale:              -80%;
```

Loop

```
$table-variants: (
  "primary":    shift-color($primary, $table-bg-scale),
  "secondary":  shift-color($secondary, $table-bg-scale),
  "success":    shift-color($success, $table-bg-scale),
  "info":       shift-color($info, $table-bg-scale),
  "warning":    shift-color($warning, $table-bg-scale),
  "danger":     shift-color($danger, $table-bg-scale),
  "light":      $light,
  "dark":       $dark,
);
```

Customizing

- The factor variables (`$table-striped-bg-factor`, `$table-active-bg-factor` & `$table-hover-bg-factor`) are used to determine the contrast in table variants.
- Apart from the light & dark table variants, theme colors are lightened by the `$table-bg-scale` variable.

Figures

Documentation and examples for displaying related images and text with the figure component in Bootstrap.

On this page

Anytime you need to display a piece of content—like an image with an optional caption, consider using a `<figure>`.

Use the included `.figure`, `.figure-img` and `.figure-caption` classes to provide some baseline styles for the HTML5 `<figure>` and `<figcaption>` elements. Images in figures have no explicit size, so be sure to add the `.img-fluid` class to your `<img>` to make it responsive.

html

```html
<figure class="figure">
  <img src="..." class="figure-img img-fluid rounded" alt="...">
  <figcaption class="figure-caption">A caption for the above image.</figcaption>
</figure>
```

Aligning the figure's caption is easy with our text utilities.

html

```html
<figure class="figure">
  <img src="..." class="figure-img img-fluid rounded" alt="...">
  <figcaption class="figure-caption text-end">A caption for the above
image.</figcaption>
</figure>
```

Sass

Variables

```scss
$figure-caption-font-size:          $small-font-size;
$figure-caption-color:              $gray-600;
```

Forms

Examples and usage guidelines for form control styles, layout options, and custom components for creating a wide variety of forms.

On this page

Form control Style textual inputs and textareas with support for multiple states.
Select Improve browser default select elements with a custom initial appearance.
Checks & radios Use our custom radio buttons and checkboxes in forms for selecting input options.
Range Replace browser default range inputs with our custom version.
Input group Attach labels and buttons to your inputs for increased semantic value.
Floating labels Create beautifully simple form labels that float over your input fields.
Layout Create inline, horizontal, or complex grid-based layouts with your forms.
Validation Validate your forms with custom or native validation behaviors and styles.

Overview

Bootstrap's form controls expand on our Rebooted form styles with classes. Use these classes to opt into their customized displays for a more consistent rendering across browsers and devices.

Be sure to use an appropriate `type` attribute on all inputs (e.g., `email` for email address or `number` for numerical information) to take advantage of newer input controls like email verification, number selection, and more.

Here's a quick example to demonstrate Bootstrap's form styles. Keep reading for documentation on required classes, form layout, and more.

We'll never share your email with anyone else.

☐

html

```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail1" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-
describedby="emailHelp">
```

```
    <div id="emailHelp" class="form-text">We'll never share your email with
anyone else.</div>
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword1" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

Form text

Block-level or inline-level form text can be created using `.form-text`.

Associating form text with form controls

Form text should be explicitly associated with the form control it relates to using the `aria-describedby` attribute. This will ensure that assistive technologies—such as screen readers—will announce this form text when the user focuses or enters the control.

Form text below inputs can be styled with `.form-text`. If a block-level element will be used, a top margin is added for easy spacing from the inputs above.

Your password must be 8-20 characters long, contain letters and numbers, and must not contain spaces, special characters, or emoji.

html

```
<label for="inputPassword5" class="form-label">Password</label>
<input type="password" id="inputPassword5" class="form-control" aria-
describedby="passwordHelpBlock">
<div id="passwordHelpBlock" class="form-text">
  Your password must be 8-20 characters long, contain letters and numbers, and
must not contain spaces, special characters, or emoji.
</div>
```

Inline text can use any typical inline HTML element (be it a `<span>`, `<small>`, or something else) with nothing more than the `.form-text` class.

Must be 8-20 characters long.

html

```
<div class="row g-3 align-items-center">
  <div class="col-auto">
    <label for="inputPassword6" class="col-form-label">Password</label>
  </div>
  <div class="col-auto">
    <input type="password" id="inputPassword6" class="form-control" aria-
describedby="passwordHelpInline">
  </div>
  <div class="col-auto">
    <span id="passwordHelpInline" class="form-text">
      Must be 8-20 characters long.
    </span>
  </div>
```

```
</div>
```

Disabled forms

Add the `disabled` boolean attribute on an input to prevent user interactions and make it appear lighter.

```
<input class="form-control" id="disabledInput" type="text" placeholder="Disabled
input here..." disabled>
```

Add the `disabled` attribute to a `<fieldset>` to disable all the controls within. Browsers treat all native form controls (`<input>`, `<select>`, and `<button>` elements) inside a `<fieldset disabled>` as disabled, preventing both keyboard and mouse interactions on them.

However, if your form also includes custom button-like elements such as `<a class="btn btn-*">...</a>`, these will only be given a style of `pointer-events: none`, meaning they are still focusable and operable using the keyboard. In this case, you must manually modify these controls by adding `tabindex="-1"` to prevent them from receiving focus and `aria-disabled="disabled"` to signal their state to assistive technologies.

html

```
<form>
  <fieldset disabled>
    <legend>Disabled fieldset example</legend>
    <div class="mb-3">
      <label for="disabledTextInput" class="form-label">Disabled input</label>
      <input type="text" id="disabledTextInput" class="form-control"
placeholder="Disabled input">
    </div>
    <div class="mb-3">
      <label for="disabledSelect" class="form-label">Disabled select
menu</label>
      <select id="disabledSelect" class="form-select">
        <option>Disabled select</option>
      </select>
    </div>
    <div class="mb-3">
      <div class="form-check">
        <input class="form-check-input" type="checkbox"
id="disabledFieldsetCheck" disabled>
        <label class="form-check-label" for="disabledFieldsetCheck">
          Can't check this
        </label>
      </div>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
  </fieldset>
</form>
```

Accessibility

Ensure that all form controls have an appropriate accessible name so that their purpose can be conveyed to users of assistive technologies. The simplest way to achieve this is to use a `<label>`

element, or—in the case of buttons—to include sufficiently descriptive text as part of the `<button>...</button>` content.

For situations where it's not possible to include a visible `<label>` or appropriate text content, there are alternative ways of still providing an accessible name, such as:

- `<label>` elements hidden using the `.visually-hidden` class
- Pointing to an existing element that can act as a label using `aria-labelledby`
- Providing a `title` attribute
- Explicitly setting the accessible name on an element using `aria-label`

If none of these are present, assistive technologies may resort to using the `placeholder` attribute as a fallback for the accessible name on `<input>` and `<textarea>` elements. The examples in this section provide a few suggested, case-specific approaches.

While using visually hidden content (`.visually-hidden`, `aria-label`, and even `placeholder` content, which disappears once a form field has content) will benefit assistive technology users, a lack of visible label text may still be problematic for certain users. Some form of visible label is generally the best approach, both for accessibility and usability.

Sass

Many form variables are set at a general level to be re-used and extended by individual form components. You'll see these most often as `$input-btn-*` and `$input-*` variables.

Variables

`$input-btn-*` variables are shared global variables between our buttons and our form components. You'll find these frequently reassigned as values to other component-specific variables.

```
$input-btn-padding-y:         .375rem;
$input-btn-padding-x:         .75rem;
$input-btn-font-family:       null;
$input-btn-font-size:         $font-size-base;
$input-btn-line-height:       $line-height-base;

$input-btn-focus-width:         .25rem;
$input-btn-focus-color-opacity: .25;
$input-btn-focus-color:         rgba($component-active-bg, $input-btn-focus-
color-opacity);
$input-btn-focus-blur:          0;
$input-btn-focus-box-shadow:    0 0 $input-btn-focus-blur $input-btn-focus-width
$input-btn-focus-color;

$input-btn-padding-y-sm:      .25rem;
$input-btn-padding-x-sm:      .5rem;
$input-btn-font-size-sm:      $font-size-sm;

$input-btn-padding-y-lg:      .5rem;
$input-btn-padding-x-lg:      1rem;
$input-btn-font-size-lg:      $font-size-lg;

$input-btn-border-width:      $border-width;
```

Form controls

Give textual form controls like `<input>`s and `<textarea>`s an upgrade with custom styles, sizing, focus states, and more.

On this page

Example

html

```
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Email address</label>
  <input type="email" class="form-control" id="exampleFormControlInput1"
placeholder="name@example.com">
</div>
<div class="mb-3">
  <label for="exampleFormControlTextarea1" class="form-label">Example
textarea</label>
  <textarea class="form-control" id="exampleFormControlTextarea1"
rows="3"></textarea>
</div>
```

Sizing

Set heights using classes like `.form-control-lg` and `.form-control-sm`.

html

```
<input class="form-control form-control-lg" type="text" placeholder=".form-
control-lg" aria-label=".form-control-lg example">
<input class="form-control" type="text" placeholder="Default input" aria-
label="default input example">
<input class="form-control form-control-sm" type="text" placeholder=".form-
control-sm" aria-label=".form-control-sm example">
```

Disabled

Add the `disabled` boolean attribute on an input to give it a grayed out appearance and remove pointer events.

html

```
<input class="form-control" type="text" placeholder="Disabled input" aria-
label="Disabled input example" disabled>
<input class="form-control" type="text" value="Disabled readonly input" aria-
label="Disabled input example" disabled readonly>
```

Readonly

Add the `readonly` boolean attribute on an input to prevent modification of the input’s value.

html

```
<input class="form-control" type="text" value="Readonly input here..." aria-
label="readonly input example" readonly>
```

Readonly plain text

If you want to have `<input readonly>` elements in your form styled as plain text, use the `.form-control-plaintext` class to remove the default form field styling and preserve the correct margin and padding.

html

```
  <div class="mb-3 row">
    <label for="staticEmail" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
      <input type="text" readonly class="form-control-plaintext"
id="staticEmail" value="email@example.com">
    </div>
  </div>
  <div class="mb-3 row">
    <label for="inputPassword" class="col-sm-2 col-form-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword">
    </div>
  </div>
```

email@example.com

html

```
<form class="row g-3">
  <div class="col-auto">
    <label for="staticEmail2" class="visually-hidden">Email</label>
    <input type="text" readonly class="form-control-plaintext" id="staticEmail2"
value="email@example.com">
  </div>
  <div class="col-auto">
    <label for="inputPassword2" class="visually-hidden">Password</label>
    <input type="password" class="form-control" id="inputPassword2"
placeholder="Password">
  </div>
  <div class="col-auto">
    <button type="submit" class="btn btn-primary mb-3">Confirm identity</button>
  </div>
</form>
```

File input

html

```html
<div class="mb-3">
  <label for="formFile" class="form-label">Default file input example</label>
  <input class="form-control" type="file" id="formFile">
</div>
<div class="mb-3">
  <label for="formFileMultiple" class="form-label">Multiple files input
example</label>
  <input class="form-control" type="file" id="formFileMultiple" multiple>
</div>
<div class="mb-3">
  <label for="formFileDisabled" class="form-label">Disabled file input
example</label>
  <input class="form-control" type="file" id="formFileDisabled" disabled>
</div>
<div class="mb-3">
  <label for="formFileSm" class="form-label">Small file input example</label>
  <input class="form-control form-control-sm" id="formFileSm" type="file">
</div>
<div>
  <label for="formFileLg" class="form-label">Large file input example</label>
  <input class="form-control form-control-lg" id="formFileLg" type="file">
</div>
```

Color

html

```html
<label for="exampleColorInput" class="form-label">Color picker</label>
<input type="color" class="form-control form-control-color"
id="exampleColorInput" value="#563d7c" title="Choose your color">
```

Datalists

Datalists allow you to create a group of `<option>`s that can be accessed (and autocompleted) from within an `<input>`. These are similar to `<select>` elements, but come with more menu styling limitations and differences. While most browsers and operating systems include some support for `<datalist>` elements, their styling is inconsistent at best.

Learn more about support for datalist elements.

html

```html
<label for="exampleDataList" class="form-label">Datalist example</label>
<input class="form-control" list="datalistOptions" id="exampleDataList"
placeholder="Type to search...">
<datalist id="datalistOptions">
  <option value="San Francisco">
  <option value="New York">
  <option value="Seattle">
  <option value="Los Angeles">
  <option value="Chicago">
</datalist>
```

Sass

Variables

`$input-*` are shared across most of our form controls (and not buttons).

```
$input-padding-y:                       $input-btn-padding-y;
$input-padding-x:                       $input-btn-padding-x;
$input-font-family:                     $input-btn-font-family;
$input-font-size:                       $input-btn-font-size;
$input-font-weight:                     $font-weight-base;
$input-line-height:                     $input-btn-line-height;

$input-padding-y-sm:                    $input-btn-padding-y-sm;
$input-padding-x-sm:                    $input-btn-padding-x-sm;
$input-font-size-sm:                    $input-btn-font-size-sm;

$input-padding-y-lg:                    $input-btn-padding-y-lg;
$input-padding-x-lg:                    $input-btn-padding-x-lg;
$input-font-size-lg:                    $input-btn-font-size-lg;

$input-bg:                              $body-bg;
$input-disabled-color:                  null;
$input-disabled-bg:                     $gray-200;
$input-disabled-border-color:           null;

$input-color:                           $body-color;
$input-border-color:                    $gray-400;
$input-border-width:                    $input-btn-border-width;
$input-box-shadow:                      $box-shadow-inset;

$input-border-radius:                   $border-radius;
$input-border-radius-sm:                $border-radius-sm;
$input-border-radius-lg:                $border-radius-lg;

$input-focus-bg:                        $input-bg;
$input-focus-border-color:              tint-color($component-active-bg, 50%);
$input-focus-color:                     $input-color;
$input-focus-width:                     $input-btn-focus-width;
$input-focus-box-shadow:                $input-btn-focus-box-shadow;

$input-placeholder-color:               $gray-600;
$input-plaintext-color:                 $body-color;

$input-height-border:                   $input-border-width * 2;

$input-height-inner:                    add($input-line-height * 1em, $input-
padding-y * 2);
$input-height-inner-half:               add($input-line-height * .5em, $input-
padding-y);
$input-height-inner-quarter:            add($input-line-height * .25em, $input-
padding-y * .5);

$input-height:                          add($input-line-height * 1em,
add($input-padding-y * 2, $input-height-border, false));
$input-height-sm:                       add($input-line-height * 1em,
add($input-padding-y-sm * 2, $input-height-border, false));
$input-height-lg:                       add($input-line-height * 1em,
add($input-padding-y-lg * 2, $input-height-border, false));

$input-transition:                      border-color .15s ease-in-out, box-
shadow .15s ease-in-out;
```

```
$form-color-width:                      3rem;
```

`$form-label-*` and `$form-text-*` are for our `<label>`s and `.form-text` component.

```
$form-label-margin-bottom:              .5rem;
$form-label-font-size:                  null;
$form-label-font-style:                 null;
$form-label-font-weight:                null;
$form-label-color:                      null;
```

```
$form-text-margin-top:                  .25rem;
$form-text-font-size:                   $small-font-size;
$form-text-font-style:                  null;
$form-text-font-weight:                 null;
$form-text-color:                       $text-muted;
```

`$form-file-*` are for file input.

```
$form-file-button-color:          $input-color;
$form-file-button-bg:             $input-group-addon-bg;
$form-file-button-hover-bg:       shade-color($form-file-button-bg, 5%);
```

Select

Customize the native `<select>`s with custom CSS that changes the element's initial appearance.

On this page

Default

Custom `<select>` menus need only a custom class, `.form-select` to trigger the custom styles. Custom styles are limited to the `<select>`'s initial appearance and cannot modify the `<option>`s due to browser limitations.

html

```
<select class="form-select" aria-label="Default select example">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

Sizing

You may also choose from small and large custom selects to match our similarly sized text inputs.

html

```
<select class="form-select form-select-lg mb-3" aria-label=".form-select-lg
example">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>

<select class="form-select form-select-sm" aria-label=".form-select-sm example">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

The `multiple` attribute is also supported:

html

```
<select class="form-select" multiple aria-label="multiple select example">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

As is the `size` attribute:

html

```
<select class="form-select" size="3" aria-label="size 3 select example">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

Disabled

Add the `disabled` boolean attribute on a select to give it a grayed out appearance and remove pointer events.

html

```
<select class="form-select" aria-label="Disabled select example" disabled>
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

Sass

Variables

```
$form-select-padding-y:             $input-padding-y;
$form-select-padding-x:             $input-padding-x;
$form-select-font-family:           $input-font-family;
$form-select-font-size:             $input-font-size;
$form-select-indicator-padding:     $form-select-padding-x * 3; // Extra padding
for background-image
$form-select-font-weight:           $input-font-weight;
$form-select-line-height:           $input-line-height;
$form-select-color:                 $input-color;
$form-select-bg:                    $input-bg;
$form-select-disabled-color:        null;
$form-select-disabled-bg:           $gray-200;
$form-select-disabled-border-color: $input-disabled-border-color;
$form-select-bg-position:           right $form-select-padding-x center;
$form-select-bg-size:               16px 12px; // In pixels because image
dimensions
$form-select-indicator-color:       $gray-800;
$form-select-indicator:             url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 16 16'><path fill='none'
stroke='#{$form-select-indicator-color}' stroke-linecap='round' stroke-
linejoin='round' stroke-width='2' d='m2 5 6 6 6-6'/></svg>");

$form-select-feedback-icon-padding-end: $form-select-padding-x * 2.5 + $form-
select-indicator-padding;
```

```
$form-select-feedback-icon-position:    center right $form-select-indicator-
padding;
$form-select-feedback-icon-size:        $input-height-inner-half $input-height-
inner-half;

$form-select-border-width:        $input-border-width;
$form-select-border-color:        $input-border-color;
$form-select-border-radius:       $input-border-radius;
$form-select-box-shadow:          $box-shadow-inset;

$form-select-focus-border-color:  $input-focus-border-color;
$form-select-focus-width:         $input-focus-width;
$form-select-focus-box-shadow:    0 0 0 $form-select-focus-width $input-btn-
focus-color;

$form-select-padding-y-sm:        $input-padding-y-sm;
$form-select-padding-x-sm:        $input-padding-x-sm;
$form-select-font-size-sm:        $input-font-size-sm;
$form-select-border-radius-sm:    $input-border-radius-sm;

$form-select-padding-y-lg:        $input-padding-y-lg;
$form-select-padding-x-lg:        $input-padding-x-lg;
$form-select-font-size-lg:        $input-font-size-lg;
$form-select-border-radius-lg:    $input-border-radius-lg;

$form-select-transition:          $input-transition;
```

Checks and radios

Create consistent cross-browser and cross-device checkboxes and radios with our completely rewritten checks component.

On this page

Approach

Browser default checkboxes and radios are replaced with the help of `.form-check`, a series of classes for both input types that improves the layout and behavior of their HTML elements, that provide greater customization and cross browser consistency. Checkboxes are for selecting one or several options in a list, while radios are for selecting one option from many.

Structurally, our `<input>`s and `<label>`s are sibling elements as opposed to an `<input>` within a `<label>`. This is slightly more verbose as you must specify `id` and `for` attributes to relate the `<input>` and `<label>`. We use the sibling selector (~) for all our `<input>` states, like `:checked` or `:disabled`. When combined with the `.form-check-label` class, we can easily style the text for each item based on the `<input>`'s state.

Our checks use custom Bootstrap icons to indicate checked or indeterminate states.

Checks

html

```
<div class="form-check">
  <input class="form-check-input" type="checkbox" value=""
id="flexCheckDefault">
  <label class="form-check-label" for="flexCheckDefault">
```

```
    Default checkbox
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="checkbox" value="" id="flexCheckChecked"
checked>
  <label class="form-check-label" for="flexCheckChecked">
    Checked checkbox
  </label>
</div>
```

Indeterminate

Checkboxes can utilize the `:indeterminate` pseudo class when manually set via JavaScript (there is no available HTML attribute for specifying it).

html

```
<div class="form-check">
  <input class="form-check-input" type="checkbox" value=""
id="flexCheckIndeterminate">
  <label class="form-check-label" for="flexCheckIndeterminate">
    Indeterminate checkbox
  </label>
</div>
```

Disabled

Add the `disabled` attribute and the associated `<label>`s are automatically styled to match with a lighter color to help indicate the input's state.

html

```
<div class="form-check">
  <input class="form-check-input" type="checkbox" value=""
id="flexCheckDisabled" disabled>
  <label class="form-check-label" for="flexCheckDisabled">
    Disabled checkbox
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="checkbox" value=""
id="flexCheckCheckedDisabled" checked disabled>
  <label class="form-check-label" for="flexCheckCheckedDisabled">
    Disabled checked checkbox
  </label>
</div>
```

Radios

html

```
<div class="form-check">
  <input class="form-check-input" type="radio" name="flexRadioDefault"
id="flexRadioDefault1">
  <label class="form-check-label" for="flexRadioDefault1">
    Default radio
  </label>
</div>
<div class="form-check">
```

```
  <input class="form-check-input" type="radio" name="flexRadioDefault"
id="flexRadioDefault2" checked>
  <label class="form-check-label" for="flexRadioDefault2">
    Default checked radio
  </label>
</div>
```

Disabled

Add the `disabled` attribute and the associated `<label>`s are automatically styled to match with a lighter color to help indicate the input's state.

html

```
<div class="form-check">
  <input class="form-check-input" type="radio" name="flexRadioDisabled"
id="flexRadioDisabled" disabled>
  <label class="form-check-label" for="flexRadioDisabled">
    Disabled radio
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="flexRadioDisabled"
id="flexRadioCheckedDisabled" checked disabled>
  <label class="form-check-label" for="flexRadioCheckedDisabled">
    Disabled checked radio
  </label>
</div>
```

Switches

A switch has the markup of a custom checkbox but uses the `.form-switch` class to render a toggle switch. Consider using `role="switch"` to more accurately convey the nature of the control to assistive technologies that support this role. In older assistive technologies, it will simply be announced as a regular checkbox as a fallback. Switches also support the `disabled` attribute.

html

```
<div class="form-check form-switch">
  <input class="form-check-input" type="checkbox" role="switch"
id="flexSwitchCheckDefault">
  <label class="form-check-label" for="flexSwitchCheckDefault">Default switch
checkbox input</label>
</div>
<div class="form-check form-switch">
  <input class="form-check-input" type="checkbox" role="switch"
id="flexSwitchCheckChecked" checked>
  <label class="form-check-label" for="flexSwitchCheckChecked">Checked switch
checkbox input</label>
</div>
<div class="form-check form-switch">
  <input class="form-check-input" type="checkbox" role="switch"
id="flexSwitchCheckDisabled" disabled>
  <label class="form-check-label" for="flexSwitchCheckDisabled">Disabled switch
checkbox input</label>
</div>
<div class="form-check form-switch">
  <input class="form-check-input" type="checkbox" role="switch"
id="flexSwitchCheckCheckedDisabled" checked disabled>
```

```
  <label class="form-check-label" for="flexSwitchCheckCheckedDisabled">Disabled
checked switch checkbox input</label>
</div>
```

Default (stacked)

By default, any number of checkboxes and radios that are immediate sibling will be vertically stacked and appropriately spaced with `.form-check`.

html

```
<div class="form-check">
  <input class="form-check-input" type="checkbox" value="" id="defaultCheck1">
  <label class="form-check-label" for="defaultCheck1">
    Default checkbox
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="checkbox" value="" id="defaultCheck2"
disabled>
  <label class="form-check-label" for="defaultCheck2">
    Disabled checkbox
  </label>
</div>
```

html

```
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios"
id="exampleRadios1" value="option1" checked>
  <label class="form-check-label" for="exampleRadios1">
    Default radio
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios"
id="exampleRadios2" value="option2">
  <label class="form-check-label" for="exampleRadios2">
    Second default radio
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios"
id="exampleRadios3" value="option3" disabled>
  <label class="form-check-label" for="exampleRadios3">
    Disabled radio
  </label>
</div>
```

Inline

Group checkboxes or radios on the same horizontal row by adding `.form-check-inline` to any `.form-check`.

html

```
<div class="form-check form-check-inline">
  <input class="form-check-input" type="checkbox" id="inlineCheckbox1"
value="option1">
```

```
  <label class="form-check-label" for="inlineCheckbox1">1</label>
</div>
<div class="form-check form-check-inline">
  <input class="form-check-input" type="checkbox" id="inlineCheckbox2"
value="option2">
  <label class="form-check-label" for="inlineCheckbox2">2</label>
</div>
<div class="form-check form-check-inline">
  <input class="form-check-input" type="checkbox" id="inlineCheckbox3"
value="option3" disabled>
  <label class="form-check-label" for="inlineCheckbox3">3 (disabled)</label>
</div>
```

html

```
<div class="form-check form-check-inline">
  <input class="form-check-input" type="radio" name="inlineRadioOptions"
id="inlineRadio1" value="option1">
  <label class="form-check-label" for="inlineRadio1">1</label>
</div>
<div class="form-check form-check-inline">
  <input class="form-check-input" type="radio" name="inlineRadioOptions"
id="inlineRadio2" value="option2">
  <label class="form-check-label" for="inlineRadio2">2</label>
</div>
<div class="form-check form-check-inline">
  <input class="form-check-input" type="radio" name="inlineRadioOptions"
id="inlineRadio3" value="option3" disabled>
  <label class="form-check-label" for="inlineRadio3">3 (disabled)</label>
</div>
```

Reverse

Put your checkboxes, radios, and switches on the opposite side with the `.form-check-reverse` modifier class.

html

```
<div class="form-check form-check-reverse">
  <input class="form-check-input" type="checkbox" value="" id="reverseCheck1">
  <label class="form-check-label" for="reverseCheck1">
    Reverse checkbox
  </label>
</div>
<div class="form-check form-check-reverse">
  <input class="form-check-input" type="checkbox" value="" id="reverseCheck2"
disabled>
  <label class="form-check-label" for="reverseCheck2">
    Disabled reverse checkbox
  </label>
</div>

<div class="form-check form-switch form-check-reverse">
  <input class="form-check-input" type="checkbox" id="flexSwitchCheckReverse">
  <label class="form-check-label" for="flexSwitchCheckReverse">Reverse switch
checkbox input</label>
</div>
```

Without labels

Omit the wrapping `.form-check` for checkboxes and radios that have no label text. Remember to still provide some form of accessible name for assistive technologies (for instance, using `aria-label`). See the forms overview accessibility section for details.

html

```
<div>
  <input class="form-check-input" type="checkbox" id="checkboxNoLabel" value="" 
aria-label="...">
</div>

<div>
  <input class="form-check-input" type="radio" name="radioNoLabel" 
id="radioNoLabel1" value="" aria-label="...">
</div>
```

Toggle buttons

Create button-like checkboxes and radio buttons by using `.btn` styles rather than `.form-check-label` on the `<label>` elements. These toggle buttons can further be grouped in a button group if needed.

Checkbox toggle buttons

html

```
<input type="checkbox" class="btn-check" id="btn-check" autocomplete="off">
<label class="btn btn-primary" for="btn-check">Single toggle</label>
```

html

```
<input type="checkbox" class="btn-check" id="btn-check-2" checked 
autocomplete="off">
<label class="btn btn-primary" for="btn-check-2">Checked</label>
```

html

```
<input type="checkbox" class="btn-check" id="btn-check-3" autocomplete="off" 
disabled>
<label class="btn btn-primary" for="btn-check-3">Disabled</label>
```

Visually, these checkbox toggle buttons are identical to the button plugin toggle buttons. However, they are conveyed differently by assistive technologies: the checkbox toggles will be announced by screen readers as “checked”/“not checked” (since, despite their appearance, they are fundamentally still checkboxes), whereas the button plugin toggle buttons will be announced as “button”/“button pressed”. The choice between these two approaches will depend on the type of toggle you are creating, and whether or not the toggle will make sense to users when announced as a checkbox or as an actual button.

Radio toggle buttons

html

```
<input type="radio" class="btn-check" name="options" id="option1"
autocomplete="off" checked>
<label class="btn btn-secondary" for="option1">Checked</label>

<input type="radio" class="btn-check" name="options" id="option2"
autocomplete="off">
<label class="btn btn-secondary" for="option2">Radio</label>

<input type="radio" class="btn-check" name="options" id="option3"
autocomplete="off" disabled>
<label class="btn btn-secondary" for="option3">Disabled</label>

<input type="radio" class="btn-check" name="options" id="option4"
autocomplete="off">
<label class="btn btn-secondary" for="option4">Radio</label>
```

Outlined styles

Different variants of `.btn`, such at the various outlined styles, are supported.

html

```
<input type="checkbox" class="btn-check" id="btn-check-outlined"
autocomplete="off">
<label class="btn btn-outline-primary" for="btn-check-outlined">Single
toggle</label><br>

<input type="checkbox" class="btn-check" id="btn-check-2-outlined" checked
autocomplete="off">
<label class="btn btn-outline-secondary"
for="btn-check-2-outlined">Checked</label><br>

<input type="radio" class="btn-check" name="options-outlined" id="success-
outlined" autocomplete="off" checked>
<label class="btn btn-outline-success" for="success-outlined">Checked success
radio</label>

<input type="radio" class="btn-check" name="options-outlined" id="danger-
outlined" autocomplete="off">
<label class="btn btn-outline-danger" for="danger-outlined">Danger radio</label>
```

Sass

Variables

```
$form-check-input-width:                  1em;
$form-check-min-height:                   $font-size-base * $line-height-base;
$form-check-padding-start:                $form-check-input-width + .5em;
$form-check-margin-bottom:                .125rem;
$form-check-label-color:                  null;
$form-check-label-cursor:                 null;
$form-check-transition:                   null;

$form-check-input-active-filter:          brightness(90%);

$form-check-input-bg:                     $input-bg;
$form-check-input-border:                 1px solid rgba($black, .25);
$form-check-input-border-radius:          .25em;
$form-check-radio-border-radius:          50%;
```

```
$form-check-input-focus-border:           $input-focus-border-color;
$form-check-input-focus-box-shadow:       $input-btn-focus-box-shadow;

$form-check-input-checked-color:          $component-active-color;
$form-check-input-checked-bg-color:       $component-active-bg;
$form-check-input-checked-border-color:   $form-check-input-checked-bg-color;
$form-check-input-checked-bg-image:       url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 20 20'><path fill='none'
stroke='#{$form-check-input-checked-color}' stroke-linecap='round' stroke-
linejoin='round' stroke-width='3' d='m6 10 3 3 6-6'/></svg>");
$form-check-radio-checked-bg-image:       url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='-4 -4 8 8'><circle r='2'
fill='#{$form-check-input-checked-color}'/></svg>");

$form-check-input-indeterminate-color:          $component-active-color;
$form-check-input-indeterminate-bg-color:       $component-active-bg;
$form-check-input-indeterminate-border-color:   $form-check-input-indeterminate-
bg-color;
$form-check-input-indeterminate-bg-image:       url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 20 20'><path fill='none'
stroke='#{$form-check-input-indeterminate-color}' stroke-linecap='round' stroke-
linejoin='round' stroke-width='3' d='M6 10h8'/></svg>");

$form-check-input-disabled-opacity:        .5;
$form-check-label-disabled-opacity:        $form-check-input-disabled-opacity;
$form-check-btn-check-disabled-opacity:    $btn-disabled-opacity;

$form-check-inline-margin-end:    1rem;
```

Input group

Easily extend form controls by adding text, buttons, or button groups on either side of textual inputs, custom selects, and custom file inputs.

On this page

Basic example

Place one add-on or button on either side of an input. You may also place one on both sides of an input. Remember to place `<label>`s outside the input group.

@

@example.com

https://example.com/users/

$ .00

@

With textarea

html

```
<div class="input-group mb-3">
  <span class="input-group-text" id="basic-addon1">@</span>
  <input type="text" class="form-control" placeholder="Username" aria-
label="Username" aria-describedby="basic-addon1">
</div>

<div class="input-group mb-3">
  <input type="text" class="form-control" placeholder="Recipient's username"
aria-label="Recipient's username" aria-describedby="basic-addon2">
  <span class="input-group-text" id="basic-addon2">@example.com</span>
</div>

<label for="basic-url" class="form-label">Your vanity URL</label>
<div class="input-group mb-3">
  <span class="input-group-text"
id="basic-addon3">https://example.com/users/</span>
```

```
  <input type="text" class="form-control" id="basic-url" aria-
describedby="basic-addon3">
</div>

<div class="input-group mb-3">
  <span class="input-group-text">$</span>
  <input type="text" class="form-control" aria-label="Amount (to the nearest
dollar)">
  <span class="input-group-text">.00</span>
</div>

<div class="input-group mb-3">
  <input type="text" class="form-control" placeholder="Username" aria-
label="Username">
  <span class="input-group-text">@</span>
  <input type="text" class="form-control" placeholder="Server" aria-
label="Server">
</div>

<div class="input-group">
  <span class="input-group-text">With textarea</span>
  <textarea class="form-control" aria-label="With textarea"></textarea>
</div>
```

Wrapping

Input groups wrap by default via `flex-wrap: wrap` in order to accommodate custom form field validation within an input group. You may disable this with `.flex-nowrap`.

@

html

```
<div class="input-group flex-nowrap">
  <span class="input-group-text" id="addon-wrapping">@</span>
  <input type="text" class="form-control" placeholder="Username" aria-
label="Username" aria-describedby="addon-wrapping">
</div>
```

Sizing

Add the relative form sizing classes to the `.input-group` itself and contents within will automatically resize—no need for repeating the form control size classes on each element.

Sizing on the individual input group elements isn’t supported.

Small

Default

Large

html

```
<div class="input-group input-group-sm mb-3">
  <span class="input-group-text" id="inputGroup-sizing-sm">Small</span>
  <input type="text" class="form-control" aria-label="Sizing example input"
aria-describedby="inputGroup-sizing-sm">
</div>

<div class="input-group mb-3">
  <span class="input-group-text" id="inputGroup-sizing-default">Default</span>
  <input type="text" class="form-control" aria-label="Sizing example input"
aria-describedby="inputGroup-sizing-default">
</div>

<div class="input-group input-group-lg">
  <span class="input-group-text" id="inputGroup-sizing-lg">Large</span>
  <input type="text" class="form-control" aria-label="Sizing example input"
aria-describedby="inputGroup-sizing-lg">
</div>
```

Checkboxes and radios

Place any checkbox or radio option within an input group's addon instead of text. We recommend adding `.mt-0` to the `.form-check-input` when there's no visible text next to the input.

html

```
<div class="input-group mb-3">
  <div class="input-group-text">
    <input class="form-check-input mt-0" type="checkbox" value="" aria-
label="Checkbox for following text input">
  </div>
  <input type="text" class="form-control" aria-label="Text input with checkbox">
</div>

<div class="input-group">
  <div class="input-group-text">
    <input class="form-check-input mt-0" type="radio" value="" aria-label="Radio
button for following text input">
  </div>
  <input type="text" class="form-control" aria-label="Text input with radio
button">
</div>
```

Multiple inputs

While multiple `<input>`s are supported visually, validation styles are only available for input groups with a single `<input>`.

First and last name

html

```
<div class="input-group">
  <span class="input-group-text">First and last name</span>
  <input type="text" aria-label="First name" class="form-control">
  <input type="text" aria-label="Last name" class="form-control">
</div>
```

Multiple addons

Multiple add-ons are supported and can be mixed with checkbox and radio input versions.

$ 0.00

$ 0.00

html

```
<div class="input-group mb-3">
```

```
  <span class="input-group-text">$</span>
  <span class="input-group-text">0.00</span>
  <input type="text" class="form-control" aria-label="Dollar amount (with dot
and two decimal places)">
</div>

<div class="input-group">
  <input type="text" class="form-control" aria-label="Dollar amount (with dot
and two decimal places)">
  <span class="input-group-text">$</span>
  <span class="input-group-text">0.00</span>
</div>
```

Button addons

html

```
<div class="input-group mb-3">
  <button class="btn btn-outline-secondary" type="button" id="button-
addon1">Button</button>
  <input type="text" class="form-control" placeholder="" aria-label="Example
text with button addon" aria-describedby="button-addon1">
</div>

<div class="input-group mb-3">
  <input type="text" class="form-control" placeholder="Recipient's username"
aria-label="Recipient's username" aria-describedby="button-addon2">
  <button class="btn btn-outline-secondary" type="button" id="button-
addon2">Button</button>
</div>

<div class="input-group mb-3">
  <button class="btn btn-outline-secondary" type="button">Button</button>
  <button class="btn btn-outline-secondary" type="button">Button</button>
  <input type="text" class="form-control" placeholder="" aria-label="Example
text with two button addons">
</div>

<div class="input-group">
  <input type="text" class="form-control" placeholder="Recipient's username"
aria-label="Recipient's username with two button addons">
  <button class="btn btn-outline-secondary" type="button">Button</button>
  <button class="btn btn-outline-secondary" type="button">Button</button>
</div>
```

Buttons with dropdowns

html

```
<div class="input-group mb-3">
  <button class="btn btn-outline-secondary dropdown-toggle" type="button" data-
bs-toggle="dropdown" aria-expanded="false">Dropdown</button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
  <input type="text" class="form-control" aria-label="Text input with dropdown
button">
```

```
</div>

<div class="input-group mb-3">
  <input type="text" class="form-control" aria-label="Text input with dropdown
button">
  <button class="btn btn-outline-secondary dropdown-toggle" type="button" data-
bs-toggle="dropdown" aria-expanded="false">Dropdown</button>
  <ul class="dropdown-menu dropdown-menu-end">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>

<div class="input-group">
  <button class="btn btn-outline-secondary dropdown-toggle" type="button" data-
bs-toggle="dropdown" aria-expanded="false">Dropdown</button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action before</a></li>
    <li><a class="dropdown-item" href="#">Another action before</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
  <input type="text" class="form-control" aria-label="Text input with 2 dropdown
buttons">
  <button class="btn btn-outline-secondary dropdown-toggle" type="button" data-
bs-toggle="dropdown" aria-expanded="false">Dropdown</button>
  <ul class="dropdown-menu dropdown-menu-end">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>
```

Segmented buttons

html

```
<div class="input-group mb-3">
  <button type="button" class="btn btn-outline-secondary">Action</button>
  <button type="button" class="btn btn-outline-secondary dropdown-toggle
dropdown-toggle-split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
  <input type="text" class="form-control" aria-label="Text input with segmented
dropdown button">
</div>

<div class="input-group">
```

```
  <input type="text" class="form-control" aria-label="Text input with segmented
dropdown button">
  <button type="button" class="btn btn-outline-secondary">Action</button>
  <button type="button" class="btn btn-outline-secondary dropdown-toggle
dropdown-toggle-split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu dropdown-menu-end">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>
```

Custom forms

Input groups include support for custom selects and custom file inputs. Browser default versions of these are not supported.

Custom select

html

```
<div class="input-group mb-3">
  <label class="input-group-text" for="inputGroupSelect01">Options</label>
  <select class="form-select" id="inputGroupSelect01">
    <option selected>Choose...</option>
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </select>
</div>

<div class="input-group mb-3">
  <select class="form-select" id="inputGroupSelect02">
    <option selected>Choose...</option>
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </select>
  <label class="input-group-text" for="inputGroupSelect02">Options</label>
</div>

<div class="input-group mb-3">
  <button class="btn btn-outline-secondary" type="button">Button</button>
  <select class="form-select" id="inputGroupSelect03" aria-label="Example select
with button addon">
    <option selected>Choose...</option>
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </select>
</div>

<div class="input-group">
  <select class="form-select" id="inputGroupSelect04" aria-label="Example select
with button addon">
    <option selected>Choose...</option>
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </select>
  <button class="btn btn-outline-secondary" type="button">Button</button>
</div>
```

Custom file input

html

```
<div class="input-group mb-3">
  <label class="input-group-text" for="inputGroupFile01">Upload</label>
  <input type="file" class="form-control" id="inputGroupFile01">
</div>

<div class="input-group mb-3">
  <input type="file" class="form-control" id="inputGroupFile02">
  <label class="input-group-text" for="inputGroupFile02">Upload</label>
</div>

<div class="input-group mb-3">
  <button class="btn btn-outline-secondary" type="button" 
id="inputGroupFileAddon03">Button</button>
  <input type="file" class="form-control" id="inputGroupFile03" aria-
describedby="inputGroupFileAddon03" aria-label="Upload">
</div>

<div class="input-group">
  <input type="file" class="form-control" id="inputGroupFile04" aria-
describedby="inputGroupFileAddon04" aria-label="Upload">
  <button class="btn btn-outline-secondary" type="button" 
id="inputGroupFileAddon04">Button</button>
</div>
```

Sass

Variables

```
$input-group-addon-padding-y:           $input-padding-y;
$input-group-addon-padding-x:           $input-padding-x;
$input-group-addon-font-weight:         $input-font-weight;
$input-group-addon-color:               $input-color;
$input-group-addon-bg:                  $gray-200;
$input-group-addon-border-color:        $input-border-color;
```

Layout

Give your forms some structure—from inline to horizontal to custom grid implementations—with our form layout options.

On this page

Forms

Every group of form fields should reside in a `<form>` element. Bootstrap provides no default styling for the `<form>` element, but there are some powerful browser features that are provided by default.

- New to browser forms? Consider reviewing the MDN form docs for an overview and complete list of available attributes.
- `<button>`s within a `<form>` default to `type="submit"`, so strive to be specific and always include a `type`.

Since Bootstrap applies `display: block` and `width: 100%` to almost all our form controls, forms will by default stack vertically. Additional classes can be used to vary this layout on a per-form basis.

Utilities

Margin utilities are the easiest way to add some structure to forms. They provide basic grouping of labels, controls, optional form text, and form validation messaging. We recommend sticking to `margin-bottom` utilities, and using a single direction throughout the form for consistency.

Feel free to build your forms however you like, with `<fieldset>`s, `<div>`s, or nearly any other element.

html

```
<div class="mb-3">
  <label for="formGroupExampleInput" class="form-label">Example label</label>
  <input type="text" class="form-control" id="formGroupExampleInput"
placeholder="Example input placeholder">
</div>
<div class="mb-3">
  <label for="formGroupExampleInput2" class="form-label">Another label</label>
  <input type="text" class="form-control" id="formGroupExampleInput2"
placeholder="Another input placeholder">
</div>
```

Form grid

More complex forms can be built using our grid classes. Use these for form layouts that require multiple columns, varied widths, and additional alignment options. **Requires the `$enable-grid-classes` Sass variable to be enabled** (on by default).

html

```
<div class="row">
  <div class="col">
    <input type="text" class="form-control" placeholder="First name" aria-
label="First name">
  </div>
  <div class="col">
    <input type="text" class="form-control" placeholder="Last name" aria-
label="Last name">
  </div>
</div>
```

Gutters

By adding gutter modifier classes, you can have control over the gutter width in as well the inline as block direction. **Also requires the `$enable-grid-classes` Sass variable to be enabled** (on by default).

html

```
<div class="row g-3">
  <div class="col">
    <input type="text" class="form-control" placeholder="First name" aria-
label="First name">
  </div>
  <div class="col">
    <input type="text" class="form-control" placeholder="Last name" aria-
label="Last name">
  </div>
</div>
```

More complex layouts can also be created with the grid system.

html

```
<form class="row g-3">
  <div class="col-md-6">
```

```
    <label for="inputEmail4" class="form-label">Email</label>
    <input type="email" class="form-control" id="inputEmail4">
  </div>
  <div class="col-md-6">
    <label for="inputPassword4" class="form-label">Password</label>
    <input type="password" class="form-control" id="inputPassword4">
  </div>
  <div class="col-12">
    <label for="inputAddress" class="form-label">Address</label>
    <input type="text" class="form-control" id="inputAddress" placeholder="1234
Main St">
  </div>
  <div class="col-12">
    <label for="inputAddress2" class="form-label">Address 2</label>
    <input type="text" class="form-control" id="inputAddress2"
placeholder="Apartment, studio, or floor">
  </div>
  <div class="col-md-6">
    <label for="inputCity" class="form-label">City</label>
    <input type="text" class="form-control" id="inputCity">
  </div>
  <div class="col-md-4">
    <label for="inputState" class="form-label">State</label>
    <select id="inputState" class="form-select">
      <option selected>Choose...</option>
      <option>...</option>
    </select>
  </div>
  <div class="col-md-2">
    <label for="inputZip" class="form-label">Zip</label>
    <input type="text" class="form-control" id="inputZip">
  </div>
  <div class="col-12">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" id="gridCheck">
      <label class="form-check-label" for="gridCheck">
        Check me out
      </label>
    </div>
  </div>
  <div class="col-12">
    <button type="submit" class="btn btn-primary">Sign in</button>
  </div>
</form>
```

Horizontal form

Create horizontal forms with the grid by adding the `.row` class to form groups and using the `.col-*-*` classes to specify the width of your labels and controls. Be sure to add `.col-form-label` to your `<label>`s as well so they're vertically centered with their associated form controls.

At times, you maybe need to use margin or padding utilities to create that perfect alignment you need. For example, we've removed the `padding-top` on our stacked radio inputs label to better align the text baseline.

◉

○

○

☐

html

```
<form>
  <div class="row mb-3">
    <label for="inputEmail3" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="inputEmail3">
    </div>
  </div>
  <div class="row mb-3">
    <label for="inputPassword3" class="col-sm-2 col-form-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword3">
    </div>
  </div>
  <fieldset class="row mb-3">
    <legend class="col-form-label col-sm-2 pt-0">Radios</legend>
    <div class="col-sm-10">
      <div class="form-check">
        <input class="form-check-input" type="radio" name="gridRadios"
id="gridRadios1" value="option1" checked>
        <label class="form-check-label" for="gridRadios1">
          First radio
        </label>
      </div>
      <div class="form-check">
        <input class="form-check-input" type="radio" name="gridRadios"
id="gridRadios2" value="option2">
        <label class="form-check-label" for="gridRadios2">
          Second radio
        </label>
      </div>
      <div class="form-check disabled">
        <input class="form-check-input" type="radio" name="gridRadios"
id="gridRadios3" value="option3" disabled>
        <label class="form-check-label" for="gridRadios3">
          Third disabled radio
        </label>
      </div>
    </div>
  </fieldset>
  <div class="row mb-3">
    <div class="col-sm-10 offset-sm-2">
      <div class="form-check">
        <input class="form-check-input" type="checkbox" id="gridCheck1">
        <label class="form-check-label" for="gridCheck1">
          Example checkbox
        </label>
      </div>
    </div>
  </div>
  <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

Horizontal form label sizing

Be sure to use `.col-form-label-sm` or `.col-form-label-lg` to your `<label>`s or `<legend>`s to correctly follow the size of `.form-control-lg` and `.form-control-sm`.

html

```
<div class="row mb-3">
  <label for="colFormLabelSm" class="col-sm-2 col-form-label col-form-label-
sm">Email</label>
  <div class="col-sm-10">
    <input type="email" class="form-control form-control-sm" id="colFormLabelSm"
placeholder="col-form-label-sm">
  </div>
</div>
<div class="row mb-3">
  <label for="colFormLabel" class="col-sm-2 col-form-label">Email</label>
  <div class="col-sm-10">
    <input type="email" class="form-control" id="colFormLabel" placeholder="col-
form-label">
  </div>
</div>
<div class="row">
  <label for="colFormLabelLg" class="col-sm-2 col-form-label col-form-label-
lg">Email</label>
  <div class="col-sm-10">
    <input type="email" class="form-control form-control-lg" id="colFormLabelLg"
placeholder="col-form-label-lg">
  </div>
</div>
```

Column sizing

As shown in the previous examples, our grid system allows you to place any number of `.col`s within a `.row`. They'll split the available width equally between them. You may also pick a subset of your columns to take up more or less space, while the remaining `.col`s equally split the rest, with specific column classes like `.col-sm-7`.

html

```
<div class="row g-3">
  <div class="col-sm-7">
    <input type="text" class="form-control" placeholder="City" aria-
label="City">
  </div>
  <div class="col-sm">
    <input type="text" class="form-control" placeholder="State" aria-
label="State">
  </div>
  <div class="col-sm">
    <input type="text" class="form-control" placeholder="Zip" aria-label="Zip">
  </div>
</div>
```

Auto-sizing

The example below uses a flexbox utility to vertically center the contents and changes `.col` to `.col-auto` so that your columns only take up as much space as needed. Put another way, the column sizes itself based on the contents.

@

html

```
<form class="row gy-2 gx-3 align-items-center">
  <div class="col-auto">
    <label class="visually-hidden" for="autoSizingInput">Name</label>
    <input type="text" class="form-control" id="autoSizingInput"
placeholder="Jane Doe">
  </div>
  <div class="col-auto">
    <label class="visually-hidden" for="autoSizingInputGroup">Username</label>
    <div class="input-group">
      <div class="input-group-text">@</div>
      <input type="text" class="form-control" id="autoSizingInputGroup"
placeholder="Username">
    </div>
  </div>
  <div class="col-auto">
    <label class="visually-hidden" for="autoSizingSelect">Preference</label>
    <select class="form-select" id="autoSizingSelect">
      <option selected>Choose...</option>
      <option value="1">One</option>
      <option value="2">Two</option>
      <option value="3">Three</option>
    </select>
  </div>
  <div class="col-auto">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" id="autoSizingCheck">
      <label class="form-check-label" for="autoSizingCheck">
        Remember me
      </label>
    </div>
  </div>
  <div class="col-auto">
    <button type="submit" class="btn btn-primary">Submit</button>
  </div>
</form>
```

You can then remix that once again with size-specific column classes.

@

html

```
<form class="row gx-3 gy-2 align-items-center">
  <div class="col-sm-3">
```

```
    <label class="visually-hidden" for="specificSizeInputName">Name</label>
    <input type="text" class="form-control" id="specificSizeInputName"
placeholder="Jane Doe">
  </div>
  <div class="col-sm-3">
    <label class="visually-hidden"
for="specificSizeInputGroupUsername">Username</label>
    <div class="input-group">
      <div class="input-group-text">@</div>
      <input type="text" class="form-control"
id="specificSizeInputGroupUsername" placeholder="Username">
    </div>
  </div>
  <div class="col-sm-3">
    <label class="visually-hidden" for="specificSizeSelect">Preference</label>
    <select class="form-select" id="specificSizeSelect">
      <option selected>Choose...</option>
      <option value="1">One</option>
      <option value="2">Two</option>
      <option value="3">Three</option>
    </select>
  </div>
  <div class="col-auto">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" id="autoSizingCheck2">
      <label class="form-check-label" for="autoSizingCheck2">
        Remember me
      </label>
    </div>
  </div>
  <div class="col-auto">
    <button type="submit" class="btn btn-primary">Submit</button>
  </div>
</form>
```

Inline forms

Use the `.row-cols-*` classes to create responsive horizontal layouts. By adding gutter modifier classes, we'll have gutters in horizontal and vertical directions. On narrow mobile viewports, the `.col-12` helps stack the form controls and more. The `.align-items-center` aligns the form elements to the middle, making the `.form-checkbox` align properly.

@

html

```
<form class="row row-cols-lg-auto g-3 align-items-center">
  <div class="col-12">
    <label class="visually-hidden"
for="inlineFormInputGroupUsername">Username</label>
    <div class="input-group">
      <div class="input-group-text">@</div>
      <input type="text" class="form-control" id="inlineFormInputGroupUsername"
placeholder="Username">
    </div>
  </div>

  <div class="col-12">
    <label class="visually-hidden" for="inlineFormSelectPref">Preference</label>
    <select class="form-select" id="inlineFormSelectPref">
      <option selected>Choose...</option>
      <option value="1">One</option>
      <option value="2">Two</option>
      <option value="3">Three</option>
    </select>
  </div>

  <div class="col-12">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" id="inlineFormCheck">
      <label class="form-check-label" for="inlineFormCheck">
        Remember me
      </label>
    </div>
  </div>

  <div class="col-12">
    <button type="submit" class="btn btn-primary">Submit</button>
  </div>
</form>
```

Validation

Provide valuable, actionable feedback to your users with HTML5 form validation, via browser default behaviors or custom styles and JavaScript.

On this page

We are aware that currently the client-side custom validation styles and tooltips are not accessible, since they are not exposed to assistive technologies. While we work on a solution, we'd recommend either using the server-side option or the default browser validation method.

How it works

Here's how form validation works with Bootstrap:

- HTML form validation is applied via CSS's two pseudo-classes, `:invalid` and `:valid`. It applies to `<input>`, `<select>`, and `<textarea>` elements.
- Bootstrap scopes the `:invalid` and `:valid` styles to parent `.was-validated` class, usually applied to the `<form>`. Otherwise, any required field without a value shows up as invalid on page load. This way, you may choose when to activate them (typically after form submission is attempted).
- To reset the appearance of the form (for instance, in the case of dynamic form submissions using AJAX), remove the `.was-validated` class from the `<form>` again after submission.
- As a fallback, `.is-invalid` and `.is-valid` classes may be used instead of the pseudo-classes for server-side validation. They do not require a `.was-validated` parent class.
- Due to constraints in how CSS works, we cannot (at present) apply styles to a `<label>` that comes before a form control in the DOM without the help of custom JavaScript.
- All modern browsers support the constraint validation API, a series of JavaScript methods for validating form controls.
- Feedback messages may utilize the browser defaults (different for each browser, and unstylable via CSS) or our custom feedback styles with additional HTML and CSS.
- You may provide custom validity messages with `setCustomValidity` in JavaScript.

With that in mind, consider the following demos for our custom form validation styles, optional server-side classes, and browser defaults.

Custom styles

For custom Bootstrap form validation messages, you'll need to add the `novalidate` boolean attribute to your `<form>`. This disables the browser default feedback tooltips, but still provides access to the form validation APIs in JavaScript. Try to submit the form below; our JavaScript will intercept the submit button and relay feedback to you. When attempting to submit, you'll see the `:invalid` and `:valid` styles applied to your form controls.

Custom feedback styles apply custom colors, borders, focus styles, and background icons to better communicate feedback. Background icons for `<select>`s are only available with `.form-select`, and not `.form-control`.

Mark

Otto

@

html

```
<form class="row g-3 needs-validation" novalidate>
  <div class="col-md-4">
    <label for="validationCustom01" class="form-label">First name</label>
    <input type="text" class="form-control" id="validationCustom01" value="Mark"
required>
    <div class="valid-feedback">
      Looks good!
    </div>
  </div>
  <div class="col-md-4">
    <label for="validationCustom02" class="form-label">Last name</label>
    <input type="text" class="form-control" id="validationCustom02" value="Otto"
required>
    <div class="valid-feedback">
      Looks good!
    </div>
  </div>
  <div class="col-md-4">
    <label for="validationCustomUsername" class="form-label">Username</label>
    <div class="input-group has-validation">
      <span class="input-group-text" id="inputGroupPrepend">@</span>
      <input type="text" class="form-control" id="validationCustomUsername"
aria-describedby="inputGroupPrepend" required>
      <div class="invalid-feedback">
        Please choose a username.
      </div>
    </div>
  </div>
  <div class="col-md-6">
    <label for="validationCustom03" class="form-label">City</label>
    <input type="text" class="form-control" id="validationCustom03" required>
    <div class="invalid-feedback">
      Please provide a valid city.
    </div>
  </div>
  <div class="col-md-3">
    <label for="validationCustom04" class="form-label">State</label>
    <select class="form-select" id="validationCustom04" required>
      <option selected disabled value="">Choose...</option>
      <option>...</option>
    </select>
    <div class="invalid-feedback">
      Please select a valid state.
    </div>
  </div>
  <div class="col-md-3">
    <label for="validationCustom05" class="form-label">Zip</label>
    <input type="text" class="form-control" id="validationCustom05" required>
    <div class="invalid-feedback">
      Please provide a valid zip.
    </div>
  </div>
  <div class="col-12">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" value="" id="invalidCheck"
required>
      <label class="form-check-label" for="invalidCheck">
        Agree to terms and conditions
      </label>
      <div class="invalid-feedback">
        You must agree before submitting.
      </div>
    </div>
  </div>
  <div class="col-12">
    <button class="btn btn-primary" type="submit">Submit form</button>
  </div>
</form>

// Example starter JavaScript for disabling form submissions if there are
invalid fields
(() => {
  'use strict'

  // Fetch all the forms we want to apply custom Bootstrap validation styles to
  const forms = document.querySelectorAll('.needs-validation')

  // Loop over them and prevent submission
  Array.from(forms).forEach(form => {
    form.addEventListener('submit', event => {
      if (!form.checkValidity()) {
        event.preventDefault()
        event.stopPropagation()
      }

      form.classList.add('was-validated')
    }, false)
  })
})()
```

Browser defaults

Not interested in custom validation feedback messages or writing JavaScript to change form behaviors? All good, you can use the browser defaults. Try submitting the form below. Depending on your browser and OS, you’ll see a slightly different style of feedback.

While these feedback styles cannot be styled with CSS, you can still customize the feedback text through JavaScript.

Mark

Otto

@

html

```html
<form class="row g-3">
  <div class="col-md-4">
    <label for="validationDefault01" class="form-label">First name</label>
    <input type="text" class="form-control" id="validationDefault01"
value="Mark" required>
  </div>
  <div class="col-md-4">
    <label for="validationDefault02" class="form-label">Last name</label>
    <input type="text" class="form-control" id="validationDefault02"
value="Otto" required>
  </div>
  <div class="col-md-4">
    <label for="validationDefaultUsername" class="form-label">Username</label>
    <div class="input-group">
      <span class="input-group-text" id="inputGroupPrepend2">@</span>
      <input type="text" class="form-control" id="validationDefaultUsername"
aria-describedby="inputGroupPrepend2" required>
    </div>
  </div>
  <div class="col-md-6">
    <label for="validationDefault03" class="form-label">City</label>
    <input type="text" class="form-control" id="validationDefault03" required>
  </div>
  <div class="col-md-3">
    <label for="validationDefault04" class="form-label">State</label>
    <select class="form-select" id="validationDefault04" required>
      <option selected disabled value="">Choose...</option>
      <option>...</option>
    </select>
  </div>
  <div class="col-md-3">
    <label for="validationDefault05" class="form-label">Zip</label>
    <input type="text" class="form-control" id="validationDefault05" required>
  </div>
  <div class="col-12">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" value=""
id="invalidCheck2" required>
      <label class="form-check-label" for="invalidCheck2">
        Agree to terms and conditions
      </label>
    </div>
  </div>
  <div class="col-12">
    <button class="btn btn-primary" type="submit">Submit form</button>
  </div>
</form>
```

Server side

We recommend using client-side validation, but in case you require server-side validation, you can indicate invalid and valid form fields with `.is-invalid` and `.is-valid`. Note that `.invalid-feedback` is also supported with these classes.

For invalid fields, ensure that the invalid feedback/error message is associated with the relevant form field using `aria-describedby` (noting that this attribute allows more than one `id` to be referenced, in case the field already points to additional form text).

To fix issues with border radius, input groups require an additional `.has-validation` class.

Mark

Looks good!

Otto

Looks good!

@

Please choose a username.

Please provide a valid city.

Please select a valid state.

Please provide a valid zip.

You must agree before submitting.

html

```
<form class="row g-3">
  <div class="col-md-4">
    <label for="validationServer01" class="form-label">First name</label>
    <input type="text" class="form-control is-valid" id="validationServer01"
value="Mark" required>
    <div class="valid-feedback">
      Looks good!
    </div>
  </div>
  <div class="col-md-4">
    <label for="validationServer02" class="form-label">Last name</label>
    <input type="text" class="form-control is-valid" id="validationServer02"
value="Otto" required>
    <div class="valid-feedback">
      Looks good!
    </div>
  </div>
  <div class="col-md-4">
    <label for="validationServerUsername" class="form-label">Username</label>
    <div class="input-group has-validation">
      <span class="input-group-text" id="inputGroupPrepend3">@</span>
      <input type="text" class="form-control is-invalid"
id="validationServerUsername" aria-describedby="inputGroupPrepend3
validationServerUsernameFeedback" required>
      <div id="validationServerUsernameFeedback" class="invalid-feedback">
        Please choose a username.
      </div>
    </div>
  </div>
  <div class="col-md-6">
    <label for="validationServer03" class="form-label">City</label>
    <input type="text" class="form-control is-invalid" id="validationServer03"
aria-describedby="validationServer03Feedback" required>
    <div id="validationServer03Feedback" class="invalid-feedback">
      Please provide a valid city.
    </div>
  </div>
  <div class="col-md-3">
    <label for="validationServer04" class="form-label">State</label>
    <select class="form-select is-invalid" id="validationServer04" aria-
describedby="validationServer04Feedback" required>
      <option selected disabled value="">Choose...</option>
      <option>...</option>
    </select>
    <div id="validationServer04Feedback" class="invalid-feedback">
      Please select a valid state.
    </div>
  </div>
  <div class="col-md-3">
    <label for="validationServer05" class="form-label">Zip</label>
    <input type="text" class="form-control is-invalid" id="validationServer05"
aria-describedby="validationServer05Feedback" required>
    <div id="validationServer05Feedback" class="invalid-feedback">
      Please provide a valid zip.
    </div>
  </div>
  <div class="col-12">
    <div class="form-check">
      <input class="form-check-input is-invalid" type="checkbox" value=""
id="invalidCheck3" aria-describedby="invalidCheck3Feedback" required>
      <label class="form-check-label" for="invalidCheck3">
        Agree to terms and conditions
      </label>
      <div id="invalidCheck3Feedback" class="invalid-feedback">
        You must agree before submitting.
      </div>
    </div>
  </div>
  <div class="col-12">
    <button class="btn btn-primary" type="submit">Submit form</button>
  </div>
</form>
```

Supported elements

Validation styles are available for the following form controls and components:

- `<input>`s and `<textarea>`s with `.form-control` (including up to one `.form-control` in input groups)
- `<select>`s with `.form-select`
- `.form-check`s

Please enter a message in the textarea.

☐

Example invalid feedback text

○

○

More example invalid feedback text

Example invalid select feedback

Example invalid form file feedback

html

```
<form class="was-validated">
  <div class="mb-3">
    <label for="validationTextarea" class="form-label">Textarea</label>
    <textarea class="form-control is-invalid" id="validationTextarea"
placeholder="Required example textarea" required></textarea>
    <div class="invalid-feedback">
      Please enter a message in the textarea.
    </div>
  </div>

  <div class="form-check mb-3">
    <input type="checkbox" class="form-check-input" id="validationFormCheck1"
required>
    <label class="form-check-label" for="validationFormCheck1">Check this
checkbox</label>
    <div class="invalid-feedback">Example invalid feedback text</div>
  </div>

  <div class="form-check">
    <input type="radio" class="form-check-input" id="validationFormCheck2"
name="radio-stacked" required>
    <label class="form-check-label" for="validationFormCheck2">Toggle this
radio</label>
  </div>
  <div class="form-check mb-3">
    <input type="radio" class="form-check-input" id="validationFormCheck3"
name="radio-stacked" required>
    <label class="form-check-label" for="validationFormCheck3">Or toggle this
other radio</label>
    <div class="invalid-feedback">More example invalid feedback text</div>
  </div>

  <div class="mb-3">
    <select class="form-select" required aria-label="select example">
      <option value="">Open this select menu</option>
      <option value="1">One</option>
      <option value="2">Two</option>
      <option value="3">Three</option>
    </select>
    <div class="invalid-feedback">Example invalid select feedback</div>
  </div>

  <div class="mb-3">
    <input type="file" class="form-control" aria-label="file example" required>
    <div class="invalid-feedback">Example invalid form file feedback</div>
  </div>

  <div class="mb-3">
```

```
    <button class="btn btn-primary" type="submit" disabled>Submit form</button>
  </div>
</form>
```

Tooltips

If your form layout allows it, you can swap the `.{valid|invalid}-feedback` classes for `.{valid|invalid}-tooltip` classes to display validation feedback in a styled tooltip. Be sure to have a parent with `position: relative` on it for tooltip positioning. In the example below, our column classes have this already, but your project may require an alternative setup.

Mark

Otto

@

html

```
<form class="row g-3 needs-validation" novalidate>
  <div class="col-md-4 position-relative">
    <label for="validationTooltip01" class="form-label">First name</label>
    <input type="text" class="form-control" id="validationTooltip01"
value="Mark" required>
    <div class="valid-tooltip">
      Looks good!
    </div>
  </div>
  <div class="col-md-4 position-relative">
    <label for="validationTooltip02" class="form-label">Last name</label>
    <input type="text" class="form-control" id="validationTooltip02"
value="Otto" required>
    <div class="valid-tooltip">
      Looks good!
    </div>
  </div>
  <div class="col-md-4 position-relative">
    <label for="validationTooltipUsername" class="form-label">Username</label>
    <div class="input-group has-validation">
      <span class="input-group-text"
id="validationTooltipUsernamePrepend">@</span>
      <input type="text" class="form-control" id="validationTooltipUsername"
aria-describedby="validationTooltipUsernamePrepend" required>
      <div class="invalid-tooltip">
        Please choose a unique and valid username.
      </div>
    </div>
  </div>
  <div class="col-md-6 position-relative">
    <label for="validationTooltip03" class="form-label">City</label>
    <input type="text" class="form-control" id="validationTooltip03" required>
    <div class="invalid-tooltip">
      Please provide a valid city.
    </div>
  </div>
  <div class="col-md-3 position-relative">
    <label for="validationTooltip04" class="form-label">State</label>
    <select class="form-select" id="validationTooltip04" required>
```

```
      <option selected disabled value="">Choose...</option>
      <option>...</option>
    </select>
    <div class="invalid-tooltip">
      Please select a valid state.
    </div>
  </div>
  <div class="col-md-3 position-relative">
    <label for="validationTooltip05" class="form-label">Zip</label>
    <input type="text" class="form-control" id="validationTooltip05" required>
    <div class="invalid-tooltip">
      Please provide a valid zip.
    </div>
  </div>
  <div class="col-12">
    <button class="btn btn-primary" type="submit">Submit form</button>
  </div>
</form>
```

Sass

Variables

```
$form-feedback-margin-top:          $form-text-margin-top;
$form-feedback-font-size:           $form-text-font-size;
$form-feedback-font-style:          $form-text-font-style;
$form-feedback-valid-color:         $success;
$form-feedback-invalid-color:       $danger;

$form-feedback-icon-valid-color:    $form-feedback-valid-color;
$form-feedback-icon-valid:          url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 8 8'><path fill='#{$form-
feedback-icon-valid-color}' d='M2.3 6.73.6 4.53c-.4-1.04.46-1.4 1.1-.8l1.1 1.4
3.4-3.8c.6-.63 1.6-.27 1.2.7l-4 4.6c-.43.5-.8.4-1.1.1z'/></svg>");
$form-feedback-icon-invalid-color:  $form-feedback-invalid-color;
$form-feedback-icon-invalid:        url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 12 12' width='12' height='12'
fill='none' stroke='#{$form-feedback-icon-invalid-color}'><circle cx='6' cy='6'
r='4.5'/><path stroke-linejoin='round' d='M5.8 3.6h.4L6 6.5z'/><circle cx='6'
cy='8.2' r='.6' fill='#{$form-feedback-icon-invalid-color}'
stroke='none'/></svg>");
```

Mixins

Two mixins are combined together, through our loop, to generate our form validation feedback styles.

```
@mixin form-validation-state-selector($state) {
  @if ($state == "valid" or $state == "invalid") {
    .was-validated #{if(&, "&", "")}:#{$state},
    #{if(&, "&", "")}.is-#{$state} {
      @content;
    }
  } @else {
    #{if(&, "&", "")}.is-#{$state} {
      @content;
    }
  }
}
```

```
@mixin form-validation-state(
  $state,
  $color,
  $icon,
  $tooltip-color: color-contrast($color),
  $tooltip-bg-color: rgba($color, $form-feedback-tooltip-opacity),
  $focus-box-shadow: 0 0 $input-btn-focus-blur $input-focus-width rgba($color,
$input-btn-focus-color-opacity)
) {
  .#{$state}-feedback {
    display: none;
    width: 100%;
    margin-top: $form-feedback-margin-top;
    @include font-size($form-feedback-font-size);
    font-style: $form-feedback-font-style;
    color: $color;
  }

  .#{$state}-tooltip {
    position: absolute;
    top: 100%;
    z-index: 5;
    display: none;
    max-width: 100%; // Contain to parent when possible
    padding: $form-feedback-tooltip-padding-y $form-feedback-tooltip-padding-x;
    margin-top: .1rem;
    @include font-size($form-feedback-tooltip-font-size);
    line-height: $form-feedback-tooltip-line-height;
    color: $tooltip-color;
    background-color: $tooltip-bg-color;
    @include border-radius($form-feedback-tooltip-border-radius);
  }

  @include form-validation-state-selector($state) {
    ~ .#{$state}-feedback,
    ~ .#{$state}-tooltip {
      display: block;
    }
  }

  .form-control {
    @include form-validation-state-selector($state) {
      border-color: $color;

      @if $enable-validation-icons {
        padding-right: $input-height-inner;
        background-image: escape-svg($icon);
        background-repeat: no-repeat;
        background-position: right $input-height-inner-quarter center;
        background-size: $input-height-inner-half $input-height-inner-half;
      }

      &:focus {
        border-color: $color;
        box-shadow: $focus-box-shadow;
      }
    }
  }

  // stylelint-disable-next-line selector-no-qualifying-type
  textarea.form-control {
    @include form-validation-state-selector($state) {
      @if $enable-validation-icons {
        padding-right: $input-height-inner;
        background-position: top $input-height-inner-quarter right $input-
height-inner-quarter;
      }
    }
  }

  .form-select {
    @include form-validation-state-selector($state) {
      border-color: $color;

      @if $enable-validation-icons {
        &:not([multiple]):not([size]),
        &:not([multiple])[size="1"] {
          padding-right: $form-select-feedback-icon-padding-end;
          background-image: escape-svg($form-select-indicator), escape-
svg($icon);
          background-position: $form-select-bg-position, $form-select-feedback-
icon-position;
          background-size: $form-select-bg-size, $form-select-feedback-icon-
size;
        }
      }

      &:focus {
        border-color: $color;
        box-shadow: $focus-box-shadow;
      }
    }
  }

  .form-control-color {
    @include form-validation-state-selector($state) {
      @if $enable-validation-icons {
        width: add($form-color-width, $input-height-inner);
      }
    }
  }

  .form-check-input {
    @include form-validation-state-selector($state) {
      border-color: $color;

      &:checked {
        background-color: $color;
      }

      &:focus {
        box-shadow: $focus-box-shadow;
      }

      ~ .form-check-label {
        color: $color;
      }
    }
  }
  .form-check-inline .form-check-input {
    ~ .#{$state}-feedback {
      margin-left: .5em;
    }
  }

  .input-group .form-control,
  .input-group .form-select {
    @include form-validation-state-selector($state) {
      @if $state == "valid" {
        z-index: 1;
      } @else if $state == "invalid" {
        z-index: 2;
      }
      &:focus {
        z-index: 3;
      }
    }
  }
}
```

Map

This is the validation Sass map from `_variables.scss`. Override or extend this to generate different or additional states.

```
$form-validation-states: (
  "valid": (
    "color": $form-feedback-valid-color,
    "icon": $form-feedback-icon-valid
  ),
  "invalid": (
    "color": $form-feedback-invalid-color,
    "icon": $form-feedback-icon-invalid
  )
);
```

Maps of `$form-validation-states` can contain three optional parameters to override tooltips and focus styles.

Loop

Used to iterate over `$form-validation-states` map values to generate our validation styles. Any modifications to the above Sass map will be reflected in your compiled CSS via this loop.

```
@each $state, $data in $form-validation-states {
  @include form-validation-state($state, $data...);
}
```

Customizing

Validation states can be customized via Sass with the `$form-validation-states` map. Located in our `_variables.scss` file, this Sass map is how we generate the default `valid`/`invalid` validation states. Included is a nested map for customizing each state's color, icon, tooltip color, and focus shadow. While no other states are supported by browsers, those using custom styles can easily add more complex form feedback.

Alerts

Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages.

On this page

Examples

Alerts are available for any length of text, as well as an optional close button. For proper styling, use one of the eight **required** contextual classes (e.g., `.alert-success`). For inline dismissal, use the alerts JavaScript plugin.

A simple primary alert—check it out!
A simple secondary alert—check it out!
A simple success alert—check it out!
A simple danger alert—check it out!
A simple warning alert—check it out!
A simple info alert—check it out!
A simple light alert—check it out!
A simple dark alert—check it out!

html

```
<div class="alert alert-primary" role="alert">
  A simple primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert">
  A simple secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  A simple success alert—check it out!
</div>
```

```
<div class="alert alert-danger" role="alert">
  A simple danger alert—check it out!
</div>
<div class="alert alert-warning" role="alert">
  A simple warning alert—check it out!
</div>
<div class="alert alert-info" role="alert">
  A simple info alert—check it out!
</div>
<div class="alert alert-light" role="alert">
  A simple light alert—check it out!
</div>
<div class="alert alert-dark" role="alert">
  A simple dark alert—check it out!
</div>
```

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.

Live example

Click the button below to show an alert (hidden with inline styles to start), then dismiss (and destroy) it with the built-in close button.

html

```
<div id="liveAlertPlaceholder"></div>
<button type="button" class="btn btn-primary" id="liveAlertBtn">Show live
alert</button>
```

We use the following JavaScript to trigger our live alert demo:

```
const alertPlaceholder = document.getElementById('liveAlertPlaceholder')

const alert = (message, type) => {
  const wrapper = document.createElement('div')
  wrapper.innerHTML = [
    `<div class="alert alert-${type} alert-dismissible" role="alert">`,
    `   <div>${message}</div>`,
    '   <button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>',
    '</div>'
  ].join('')

  alertPlaceholder.append(wrapper)
}

const alertTrigger = document.getElementById('liveAlertBtn')
if (alertTrigger) {
  alertTrigger.addEventListener('click', () => {
    alert('Nice, you triggered this alert message!', 'success')
  })
}
```

Link color

Use the `.alert-link` utility class to quickly provide matching colored links within any alert.

A simple primary alert with an example link. Give it a click if you like.
A simple secondary alert with an example link. Give it a click if you like.
A simple success alert with an example link. Give it a click if you like.
A simple danger alert with an example link. Give it a click if you like.
A simple warning alert with an example link. Give it a click if you like.
A simple info alert with an example link. Give it a click if you like.
A simple light alert with an example link. Give it a click if you like.
A simple dark alert with an example link. Give it a click if you like.

html

```
<div class="alert alert-primary" role="alert">
  A simple primary alert with <a href="#" class="alert-link">an example
link</a>. Give it a click if you like.
</div>
<div class="alert alert-secondary" role="alert">
  A simple secondary alert with <a href="#" class="alert-link">an example
link</a>. Give it a click if you like.
</div>
<div class="alert alert-success" role="alert">
  A simple success alert with <a href="#" class="alert-link">an example
link</a>. Give it a click if you like.
</div>
<div class="alert alert-danger" role="alert">
  A simple danger alert with <a href="#" class="alert-link">an example link</a>.
Give it a click if you like.
</div>
<div class="alert alert-warning" role="alert">
  A simple warning alert with <a href="#" class="alert-link">an example
link</a>. Give it a click if you like.
</div>
<div class="alert alert-info" role="alert">
  A simple info alert with <a href="#" class="alert-link">an example link</a>.
Give it a click if you like.
</div>
<div class="alert alert-light" role="alert">
  A simple light alert with <a href="#" class="alert-link">an example link</a>.
Give it a click if you like.
</div>
<div class="alert alert-dark" role="alert">
  A simple dark alert with <a href="#" class="alert-link">an example link</a>.
Give it a click if you like.
</div>
```

Additional content

Alerts can also contain additional HTML elements like headings, paragraphs and dividers.

Well done!

Aww yeah, you successfully read this important alert message. This example text is going to run a bit longer so that you can see how spacing within an alert works with this kind of content.

Whenever you need to, be sure to use margin utilities to keep things nice and tidy.

html

```
<div class="alert alert-success" role="alert">
  <h4 class="alert-heading">Well done!</h4>
  <p>Aww yeah, you successfully read this important alert message. This example
text is going to run a bit longer so that you can see how spacing within an
alert works with this kind of content.</p>
  <hr>
  <p class="mb-0">Whenever you need to, be sure to use margin utilities to keep
things nice and tidy.</p>
</div>
```

Icons

Similarly, you can use flexbox utilities and Bootstrap Icons to create alerts with icons. Depending on your icons and content, you may want to add more utilities or custom styles.

An example alert with an icon

html

```
<div class="alert alert-primary d-flex align-items-center" role="alert">
  <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
fill="currentColor" class="bi bi-exclamation-triangle-fill flex-shrink-0 me-2"
viewBox="0 0 16 16" role="img" aria-label="Warning:">
    <path d="M8.982 1.566a1.13 1.13 0 0 0-1.96 0L.165 13.233c-.457.778.091
1.767.98 1.767h13.713c.889 0 1.438-.99.98-1.767L8.982 1.566zM8 5c.535
0 .954.462.9.995l-.35 3.507a.552.552 0 0 1-1.1 0L7.1 5.995A.905.905 0 0 1 8
5zm.002 6a1 1 0 1 1 0 2 1 1 0 0 1 0-2z"/>
  </svg>
  <div>
    An example alert with an icon
  </div>
</div>
```

Need more than one icon for your alerts? Consider using more Bootstrap Icons and making a local SVG sprite like so to easily reference the same icons repeatedly.

An example alert with an icon
An example success alert with an icon
An example warning alert with an icon
An example danger alert with an icon

html

```
<svg xmlns="http://www.w3.org/2000/svg" style="display: none;">
  <symbol id="check-circle-fill" fill="currentColor" viewBox="0 0 16 16">
    <path d="M16 8A8 8 0 1 1 0 8a8 8 0 0 1 16 0zm-3.97-3.03a.75.75 0 0 0-
1.08.022L7.477 9.417 5.384 7.323a.75.75 0 0 0-1.06 1.06L6.97 11.03a.75.75 0 0 0
1.079-.02l3.992-4.99a.75.75 0 0 0-.01-1.05z"/>
  </symbol>
  <symbol id="info-fill" fill="currentColor" viewBox="0 0 16 16">
    <path d="M8 16A8 8 0 1 0 8 0a8 8 0 0 0 0 16zm.93-9.412-1
4.705c-.07.34.029.533.304.533.194
0 .487-.07.686-.246l-.088.416c-.287.346-.92.598-1.465.598-.703 0-
1.002-.422-.808-1.319l.738-3.468c.064-.293.006-.399-.287-.47l-.451-.081.082-.381
2.29-.287zM8 5.5a1 1 0 1 1 0-2 1 1 0 0 1 0 2z"/>
  </symbol>
  <symbol id="exclamation-triangle-fill" fill="currentColor" viewBox="0 0 16
16">
    <path d="M8.982 1.566a1.13 1.13 0 0 0-1.96 0L.165 13.233c-.457.778.091
1.767.98 1.767h13.713c.889 0 1.438-.99.98-1.767L8.982 1.566zM8 5c.535
```

```
0 .954.462.9.995l-.35 3.507a.552.552 0 0 1-1.1 0L7.1 5.995A.905.905 0 0 1 8
5zm.002 6a1 1 0 1 1 0 2 1 1 0 0 1 0-2z"/>
  </symbol>
</svg>

<div class="alert alert-primary d-flex align-items-center" role="alert">
  <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-
label="Info:"><use xlink:href="#info-fill"/></svg>
  <div>
    An example alert with an icon
  </div>
</div>
<div class="alert alert-success d-flex align-items-center" role="alert">
  <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-
label="Success:"><use xlink:href="#check-circle-fill"/></svg>
  <div>
    An example success alert with an icon
  </div>
</div>
<div class="alert alert-warning d-flex align-items-center" role="alert">
  <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-
label="Warning:"><use xlink:href="#exclamation-triangle-fill"/></svg>
  <div>
    An example warning alert with an icon
  </div>
</div>
<div class="alert alert-danger d-flex align-items-center" role="alert">
  <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-
label="Danger:"><use xlink:href="#exclamation-triangle-fill"/></svg>
  <div>
    An example danger alert with an icon
  </div>
</div>
```

Dismissing

Using the alert JavaScript plugin, it’s possible to dismiss any alert inline. Here’s how:

- Be sure you’ve loaded the alert plugin, or the compiled Bootstrap JavaScript.
- Add a close button and the `.alert-dismissible` class, which adds extra padding to the right of the alert and positions the close button.
- On the close button, add the `data-bs-dismiss="alert"` attribute, which triggers the JavaScript functionality. Be sure to use the `<button>` element with it for proper behavior across all devices.
- To animate alerts when dismissing them, be sure to add the `.fade` and `.show` classes.

You can see this in action with a live demo:

Holy guacamole! You should check in on some of those fields below.

html

```
<div class="alert alert-warning alert-dismissible fade show" role="alert">
  <strong>Holy guacamole!</strong> You should check in on some of those fields
below.
  <button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>
</div>
```

When an alert is dismissed, the element is completely removed from the page structure. If a keyboard user dismisses the alert using the close button, their focus will suddenly be lost and, depending on the browser, reset to the start of the page/document. For this reason, we recommend including additional JavaScript that listens for the `closed.bs.alert` event and programmatically sets `focus()` to the most appropriate location in the page. If you're planning to move focus to a non-interactive element that normally does not receive focus, make sure to add `tabindex="-1"` to the element.

CSS

Variables

Added in v5.2.0

As part of Bootstrap's evolving CSS variables approach, alerts now use local CSS variables on `.alert` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}alert-bg: transparent;
  --#{$prefix}alert-padding-x: #{$alert-padding-x};
  --#{$prefix}alert-padding-y: #{$alert-padding-y};
  --#{$prefix}alert-margin-bottom: #{$alert-margin-bottom};
  --#{$prefix}alert-color: inherit;
  --#{$prefix}alert-border-color: transparent;
  --#{$prefix}alert-border: #{$alert-border-width} solid var(--#{$prefix}alert-
border-color);
  --#{$prefix}alert-border-radius: #{$alert-border-radius};
```

Sass variables

```
$alert-padding-y:               $spacer;
$alert-padding-x:               $spacer;
$alert-margin-bottom:           1rem;
$alert-border-radius:           $border-radius;
$alert-link-font-weight:        $font-weight-bold;
$alert-border-width:            $border-width;
$alert-bg-scale:                -80%;
$alert-border-scale:            -70%;
$alert-color-scale:             40%;
$alert-dismissible-padding-r:   $alert-padding-x * 3; // 3x covers width of x
plus default padding on either side
```

Sass mixin

Used in combination with `$theme-colors` to create contextual modifier classes for our alerts.

```
@mixin alert-variant($background, $border, $color) {
  --#{$prefix}alert-color: #{$color};
  --#{$prefix}alert-bg: #{$background};
  --#{$prefix}alert-border-color: #{$border};

  @if $enable-gradients {
    background-image: var(--#{$prefix}gradient);
  }

  .alert-link {
    color: shade-color($color, 20%);
  }
}
```

Sass loop

Loop that generates the modifier classes with the `alert-variant()` mixin.

```
// Generate contextual modifier classes for colorizing the alert.

@each $state, $value in $theme-colors {
  $alert-background: shift-color($value, $alert-bg-scale);
  $alert-border: shift-color($value, $alert-border-scale);
  $alert-color: shift-color($value, $alert-color-scale);

  @if (contrast-ratio($alert-background, $alert-color) < $min-contrast-ratio) {
    $alert-color: mix($value, color-contrast($alert-background), abs($alert-
color-scale));
  }
  .alert-#{$state} {
    @include alert-variant($alert-background, $alert-border, $alert-color);
  }
}
```

JavaScript behavior

Initialize

Initialize elements as alerts

```
const alertList = document.querySelectorAll('.alert')
const alerts = [...alertList].map(element => new bootstrap.Alert(element))
```

For the sole purpose of dismissing an alert, it isn't necessary to initialize the component manually via the JS API. By making use of `data-bs-dismiss="alert"`, the component will be initialized automatically and properly dismissed.

See the triggers section for more details.

Triggers

Dismissal can be achieved with the `data` attribute on a button **within the alert** as demonstrated below:

```
<button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>
```

or on a button **outside the alert** using the `data-bs-target` as demonstrated below:

```
<button type="button" class="btn-close" data-bs-dismiss="alert" data-bs-
target="#my-alert" aria-label="Close"></button>
```

Note that closing an alert will remove it from the DOM.

Methods

You can create an alert instance with the alert constructor, for example:

```
const bsAlert = new bootstrap.Alert('#myAlert')
```

This makes an alert listen for click events on descendant elements which have the `data-bs-dismiss="alert"` attribute. (Not necessary when using the data-api's auto-initialization.)

Method	Description
`close`	Closes an alert by removing it from the DOM. If the `.fade` and `.show` classes are present on the element, the alert will fade out before it is removed.
`dispose`	Destroys an element's alert. (Removes stored data on the DOM element)
`getInstance`	Static method which allows you to get the alert instance associated to a DOM element. For example: `bootstrap.Alert.getInstance(alert)`.
`getOrCreateInstance`	Static method which returns an alert instance associated to a DOM element or create a new one in case it wasn't initialized. You can use it like this: `bootstrap.Alert.getOrCreateInstance(element)`.

Basic usage:

```
const alert = bootstrap.Alert.getOrCreateInstance('#myAlert')
alert.close()
```

Events

Bootstrap's alert plugin exposes a few events for hooking into alert functionality.

Event	Description
`close.bs.alert`	Fires immediately when the `close` instance method is called.
`closed.bs.alert`	Fired when the alert has been closed and CSS transitions have completed.

```
const myAlert = document.getElementById('myAlert')
myAlert.addEventListener('closed.bs.alert', event => {
  // do something, for instance, explicitly move focus to the most appropriate
element,
  // so it doesn't get lost/reset to the start of the page
  // document.getElementById('...').focus()
})
```

Breadcrumb

Indicate the current page’s location within a navigational hierarchy that automatically adds separators via CSS.

On this page

Example

Use an ordered or unordered list with linked list items to create a minimally styled breadcrumb. Use our utilities to add additional styles as desired.

1. Home

1. Home
2. Library

1. Home
2. Library
3. Data

html

```
<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item active" aria-current="page">Home</li>
  </ol>
</nav>

<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Library</li>
  </ol>
</nav>

<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item"><a href="#">Library</a></li>
    <li class="breadcrumb-item active" aria-current="page">Data</li>
  </ol>
</nav>
```

Dividers

Dividers are automatically added in CSS through `::before` and `content`. They can be changed by modifying a local CSS custom property `--bs-breadcrumb-divider`, or through the `$breadcrumb-divider` Sass variable — and `$breadcrumb-divider-flipped` for its RTL counterpart, if needed. We default to our Sass variable, which is set as a fallback to the custom property. This way, you get a global divider that you can override without recompiling CSS at any time.

1. Home
2. Library

html

```
<nav style="--bs-breadcrumb-divider: '>';" aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Library</li>
  </ol>
</nav>
```

When modifying via Sass, the quote function is required to generate the quotes around a string. For example, using > as the divider, you can use this:

```
$breadcrumb-divider: quote(">");
```

It's also possible to use an **embedded SVG icon**. Apply it via our CSS custom property, or use the Sass variable.

Using embedded SVG

Inlining SVG as data URI requires to URL escape a few characters, most notably <, > and #. That's why the `$breadcrumb-divider` variable is passed through our `escape-svg()` Sass function. When using the CSS custom property, you need to URL escape your SVG on your own. Read Kevin Weber's explanations on CodePen for detailed information on what to escape.

1. Home
2. Library

html

```
<nav style="--bs-breadcrumb-divider: url(&#34;data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' width='8' height='8'%3E%3Cpath d='M2.5 0L1
1.5 3.5 4 1 6.5 2.5 8l4-4-4-4z' fill='%236c757d'/%3E%3C/svg%3E&#34;);" aria-
label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Library</li>
  </ol>
</nav>
```

```
$breadcrumb-divider: url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' width='8' height='8'><path d='M2.5 0L1 1.5
3.5 4 1 6.5 2.5 8l4-4-4-4z' fill='#{$breadcrumb-divider-color}'/></svg>");
```

You can also remove the divider setting `--bs-breadcrumb-divider: '';` (empty strings in CSS custom properties counts as a value), or setting the Sass variable to `$breadcrumb-divider: none;`.

1. Home
2. Library

html

```
<nav style="--bs-breadcrumb-divider: '';" aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Library</li>
  </ol>
</nav>
```

```
$breadcrumb-divider: none;
```

Accessibility

Since breadcrumbs provide a navigation, it’s a good idea to add a meaningful label such as `aria-label="breadcrumb"` to describe the type of navigation provided in the `<nav>` element, as well as applying an `aria-current="page"` to the last item of the set to indicate that it represents the current page.

For more information, see the WAI-ARIA Authoring Practices for the breadcrumb pattern.

CSS

Variables

Added in v5.2.0

As part of Bootstrap’s evolving CSS variables approach, breadcrumbs now use local CSS variables on `.breadcrumb` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}breadcrumb-padding-x: #{$breadcrumb-padding-x};
  --#{$prefix}breadcrumb-padding-y: #{$breadcrumb-padding-y};
  --#{$prefix}breadcrumb-margin-bottom: #{$breadcrumb-margin-bottom};
  @include rfs($breadcrumb-font-size, --#{$prefix}breadcrumb-font-size);
  --#{$prefix}breadcrumb-bg: #{$breadcrumb-bg};
  --#{$prefix}breadcrumb-border-radius: #{$breadcrumb-border-radius};
  --#{$prefix}breadcrumb-divider-color: #{$breadcrumb-divider-color};
  --#{$prefix}breadcrumb-item-padding-x: #{$breadcrumb-item-padding-x};
  --#{$prefix}breadcrumb-item-active-color: #{$breadcrumb-active-color};
```

Sass variables

```
$breadcrumb-font-size:              null;
$breadcrumb-padding-y:              0;
$breadcrumb-padding-x:              0;
$breadcrumb-item-padding-x:         .5rem;
$breadcrumb-margin-bottom:          1rem;
```

```
$breadcrumb-bg:                    null;
$breadcrumb-divider-color:         $gray-600;
$breadcrumb-active-color:          $gray-600;
$breadcrumb-divider:               quote("/");
$breadcrumb-divider-flipped:       $breadcrumb-divider;
$breadcrumb-border-radius:         null;
```

Buttons

Use Bootstrap’s custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more.

On this page

Examples

Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control.

html

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Link</button>
```

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.

Disable text wrapping

If you don’t want the button text to wrap, you can add the `.text-nowrap` class to the button. In Sass, you can set `$btn-white-space: nowrap` to disable text wrapping for each button.

Button tags

The `.btn` classes are designed to be used with the `<button>` element. However, you can also use these classes on `<a>` or `<input>` elements (though some browsers may apply a slightly different rendering).

When using button classes on `<a>` elements that are used to trigger in-page functionality (like collapsing content), rather than linking to new pages or sections within the current page, these links should be given a `role="button"` to appropriately convey their purpose to assistive technologies such as screen readers.

html

```
<a class="btn btn-primary" href="#" role="button">Link</a>
<button class="btn btn-primary" type="submit">Button</button>
<input class="btn btn-primary" type="button" value="Input">
<input class="btn btn-primary" type="submit" value="Submit">
<input class="btn btn-primary" type="reset" value="Reset">
```

Outline buttons

In need of a button, but not the hefty background colors they bring? Replace the default modifier classes with the `.btn-outline-*` ones to remove all background images and colors on any button.

html

```
<button type="button" class="btn btn-outline-primary">Primary</button>
<button type="button" class="btn btn-outline-secondary">Secondary</button>
<button type="button" class="btn btn-outline-success">Success</button>
<button type="button" class="btn btn-outline-danger">Danger</button>
<button type="button" class="btn btn-outline-warning">Warning</button>
<button type="button" class="btn btn-outline-info">Info</button>
<button type="button" class="btn btn-outline-light">Light</button>
<button type="button" class="btn btn-outline-dark">Dark</button>
```

Some of the button styles use a relatively light foreground color, and should only be used on a dark background in order to have sufficient contrast.

Sizes

Fancy larger or smaller buttons? Add `.btn-lg` or `.btn-sm` for additional sizes.

html

```
<button type="button" class="btn btn-primary btn-lg">Large button</button>
<button type="button" class="btn btn-secondary btn-lg">Large button</button>
```

html

```
<button type="button" class="btn btn-primary btn-sm">Small button</button>
<button type="button" class="btn btn-secondary btn-sm">Small button</button>
```

You can even roll your own custom sizing with CSS variables:

html

```
<button type="button" class="btn btn-primary"
        style="--bs-btn-padding-y: .25rem; --bs-btn-padding-x: .5rem; --bs-btn-
font-size: .75rem;">
  Custom button
</button>
```

Disabled state

Make buttons look inactive by adding the `disabled` boolean attribute to any `<button>` element. Disabled buttons have `pointer-events: none` applied to, preventing hover and active states from triggering.

html

```
<button type="button" class="btn btn-primary" disabled>Primary button</button>
<button type="button" class="btn btn-secondary" disabled>Button</button>
<button type="button" class="btn btn-outline-primary" disabled>Primary
button</button>
<button type="button" class="btn btn-outline-secondary" disabled>Button</button>
```

Disabled buttons using the `<a>` element behave a bit different:

- `<a>`s don't support the `disabled` attribute, so you must add the `.disabled` class to make it visually appear disabled.
- Some future-friendly styles are included to disable all `pointer-events` on anchor buttons.
- Disabled buttons using `<a>` should include the `aria-disabled="true"` attribute to indicate the state of the element to assistive technologies.
- Disabled buttons using `<a>` *should not* include the `href` attribute.

html

```
<a class="btn btn-primary disabled" role="button" aria-disabled="true">Primary
link</a>
<a class="btn btn-secondary disabled" role="button"
aria-disabled="true">Link</a>
```

Link functionality caveat

To cover cases where you have to keep the `href` attribute on a disabled link, the `.disabled` class uses `pointer-events: none` to try to disable the link functionality of `<a>`s. Note that this CSS property is not yet standardized for HTML, but all modern browsers support it. In addition, even in browsers that do support `pointer-events: none`, keyboard navigation remains unaffected, meaning that sighted keyboard users and users of assistive technologies will still be able to activate these links. So to be safe, in addition to `aria-disabled="true"`, also include a `tabindex="-1"` attribute on these links to prevent them from receiving keyboard focus, and use custom JavaScript to disable their functionality altogether.

html

```
<a href="#" class="btn btn-primary disabled" tabindex="-1" role="button" aria-
disabled="true">Primary link</a>
<a href="#" class="btn btn-secondary disabled" tabindex="-1" role="button" aria-
disabled="true">Link</a>
```

Block buttons

Create responsive stacks of full-width, “block buttons” like those in Bootstrap 4 with a mix of our display and gap utilities. By using utilities instead of button specific classes, we have much greater control over spacing, alignment, and responsive behaviors.

html

```
<div class="d-grid gap-2">
  <button class="btn btn-primary" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

Here we create a responsive variation, starting with vertically stacked buttons until the `md` breakpoint, where `.d-md-block` replaces the `.d-grid` class, thus nullifying the `gap-2` utility. Resize your browser to see them change.

html

```
<div class="d-grid gap-2 d-md-block">
  <button class="btn btn-primary" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

You can adjust the width of your block buttons with grid column width classes. For example, for a half-width “block button”, use `.col-6`. Center it horizontally with `.mx-auto`, too.

html

```
<div class="d-grid gap-2 col-6 mx-auto">
  <button class="btn btn-primary" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

Additional utilities can be used to adjust the alignment of buttons when horizontal. Here we’ve taken our previous responsive example and added some flex utilities and a margin utility on the button to right align the buttons when they’re no longer stacked.

html

```
<div class="d-grid gap-2 d-md-flex justify-content-md-end">
  <button class="btn btn-primary me-md-2" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

Button plugin

The button plugin allows you to create simple on/off toggle buttons.

Visually, these toggle buttons are identical to the checkbox toggle buttons. However, they are conveyed differently by assistive technologies: the checkbox toggles will be announced by screen readers as “checked”/“not checked” (since, despite their appearance, they are fundamentally still checkboxes), whereas these toggle buttons will be announced as “button”/“button pressed”. The choice between these two approaches will depend on the type of toggle you are creating, and whether or not the toggle will make sense to users when announced as a checkbox or as an actual button.

Toggle states

Add `data-bs-toggle="button"` to toggle a button’s `active` state. If you’re pre-toggling a button, you must manually add the `.active` class **and** `aria-pressed="true"` to ensure that it is conveyed appropriately to assistive technologies.

html

```
<button type="button" class="btn btn-primary" data-bs-toggle="button">Toggle
button</button>
<button type="button" class="btn btn-primary active" data-bs-toggle="button"
aria-pressed="true">Active toggle button</button>
<button type="button" class="btn btn-primary" disabled data-bs-
toggle="button">Disabled toggle button</button>
```

html

```
<a href="#" class="btn btn-primary" role="button" data-bs-toggle="button">Toggle
link</a>
<a href="#" class="btn btn-primary active" role="button" data-bs-toggle="button"
aria-pressed="true">Active toggle link</a>
<a class="btn btn-primary disabled" aria-disabled="true" role="button" data-bs-
toggle="button">Disabled toggle link</a>
```

Methods

You can create a button instance with the button constructor, for example:

```
const bsButton = new bootstrap.Button('#myButton')
```

Method	Description
`toggle`	Toggles push state. Gives the button the appearance that it has been activated.
`dispose`	Destroys an element’s button. (Removes stored data on the DOM element)
`getInstance`	Static method which allows you to get the button instance associated to a DOM element, you can use it like this: `bootstrap.Button.getInstance(element)`
`getOrCreateInstance`	Static method which returns a button instance associated to a DOM element or create a new one in case it wasn’t initialized. You can use it like this: `bootstrap.Button.getOrCreateInstance(element)`

For example, to toggle all buttons

```
document.querySelectorAll('.btn').forEach(buttonElement => {
  const button = bootstrap.Button.getOrCreateInstance(buttonElement)
  button.toggle()
})
```

CSS

Variables

Added in v5.2.0

As part of Bootstrap’s evolving CSS variables approach, buttons now use local CSS variables on `.btn` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}btn-padding-x: #{$btn-padding-x};
  --#{$prefix}btn-padding-y: #{$btn-padding-y};
  --#{$prefix}btn-font-family: #{$btn-font-family};
  @include rfs($btn-font-size, --#{$prefix}btn-font-size);
  --#{$prefix}btn-font-weight: #{$btn-font-weight};
  --#{$prefix}btn-line-height: #{$btn-line-height};
  --#{$prefix}btn-color: #{$body-color};
  --#{$prefix}btn-bg: transparent;
  --#{$prefix}btn-border-width: #{$btn-border-width};
  --#{$prefix}btn-border-color: transparent;
  --#{$prefix}btn-border-radius: #{$btn-border-radius};
  --#{$prefix}btn-box-shadow: #{$btn-box-shadow};
  --#{$prefix}btn-disabled-opacity: #{$btn-disabled-opacity};
  --#{$prefix}btn-focus-box-shadow: 0 0 0 #{$btn-focus-width} rgba(var(--
#{$prefix}btn-focus-shadow-rgb), .5);
```

Each `.btn-*` modifier class updates the appropriate CSS variables to minimize additional CSS rules with our `button-variant()`, `button-outline-variant()`, and `button-size()` mixins.

Here’s an example of building a custom `.btn-*` modifier class like we do for the buttons unique to our docs by reassigning Bootstrap’s CSS variables with a mixture of our own CSS and Sass variables.

```
.btn-bd-primary {
  --bs-btn-font-weight: 600;
  --bs-btn-color: var(--bs-white);
  --bs-btn-bg: var(--bd-violet);
  --bs-btn-border-color: var(--bd-violet);
  --bs-btn-border-radius: .5rem;
  --bs-btn-hover-color: var(--bs-white);
  --bs-btn-hover-bg: #{shade-color($bd-violet, 10%)};
  --bs-btn-hover-border-color: #{shade-color($bd-violet, 10%)};
  --bs-btn-focus-shadow-rgb: var(--bd-violet-rgb);
  --bs-btn-active-color: var(--bs-btn-hover-color);
  --bs-btn-active-bg: #{shade-color($bd-violet, 20%)};
  --bs-btn-active-border-color: #{shade-color($bd-violet, 20%)};
}
```

Sass variables

```
$btn-padding-y:               $input-btn-padding-y;
$btn-padding-x:               $input-btn-padding-x;
$btn-font-family:             $input-btn-font-family;
$btn-font-size:               $input-btn-font-size;
```

```
$btn-line-height:              $input-btn-line-height;
$btn-white-space:              null; // Set to `nowrap` to prevent text wrapping

$btn-padding-y-sm:             $input-btn-padding-y-sm;
$btn-padding-x-sm:             $input-btn-padding-x-sm;
$btn-font-size-sm:             $input-btn-font-size-sm;

$btn-padding-y-lg:             $input-btn-padding-y-lg;
$btn-padding-x-lg:             $input-btn-padding-x-lg;
$btn-font-size-lg:             $input-btn-font-size-lg;

$btn-border-width:             $input-btn-border-width;

$btn-font-weight:              $font-weight-normal;
$btn-box-shadow:               inset 0 1px 0 rgba($white, .15), 0 1px 1px
rgba($black, .075);
$btn-focus-width:              $input-btn-focus-width;
$btn-focus-box-shadow:         $input-btn-focus-box-shadow;
$btn-disabled-opacity:         .65;
$btn-active-box-shadow:        inset 0 3px 5px rgba($black, .125);

$btn-link-color:               var(--#{$prefix}link-color);
$btn-link-hover-color:         var(--#{$prefix}link-hover-color);
$btn-link-disabled-color:      $gray-600;

// Allows for customizing button radius independently from global border radius
$btn-border-radius:            $border-radius;
$btn-border-radius-sm:         $border-radius-sm;
$btn-border-radius-lg:         $border-radius-lg;

$btn-transition:               color .15s ease-in-out, background-color .15s
ease-in-out, border-color .15s ease-in-out, box-shadow .15s ease-in-out;

$btn-hover-bg-shade-amount:       15%;
$btn-hover-bg-tint-amount:        15%;
$btn-hover-border-shade-amount:   20%;
$btn-hover-border-tint-amount:    10%;
$btn-active-bg-shade-amount:      20%;
$btn-active-bg-tint-amount:       20%;
$btn-active-border-shade-amount:  25%;
$btn-active-border-tint-amount:   10%;
```

Sass mixins

There are three mixins for buttons: button and button outline variant mixins (both based on `$theme-colors`), plus a button size mixin.

```
@mixin button-variant(
  $background,
  $border,
  $color: color-contrast($background),
  $hover-background: if($color == $color-contrast-light, shade-
color($background, $btn-hover-bg-shade-amount), tint-color($background, $btn-
hover-bg-tint-amount)),
  $hover-border: if($color == $color-contrast-light, shade-color($border, $btn-
hover-border-shade-amount), tint-color($border, $btn-hover-border-tint-amount)),
  $hover-color: color-contrast($hover-background),
  $active-background: if($color == $color-contrast-light, shade-
color($background, $btn-active-bg-shade-amount), tint-color($background, $btn-
active-bg-tint-amount)),
  $active-border: if($color == $color-contrast-light, shade-color($border, $btn-
active-border-shade-amount), tint-color($border, $btn-active-border-tint-
amount)),
  $active-color: color-contrast($active-background),
  $disabled-background: $background,
  $disabled-border: $border,
  $disabled-color: color-contrast($disabled-background)
) {
  --#{$prefix}btn-color: #{$color};
  --#{$prefix}btn-bg: #{$background};
  --#{$prefix}btn-border-color: #{$border};
  --#{$prefix}btn-hover-color: #{$hover-color};
  --#{$prefix}btn-hover-bg: #{$hover-background};
  --#{$prefix}btn-hover-border-color: #{$hover-border};
  --#{$prefix}btn-focus-shadow-rgb: #{to-rgb(mix($color, $border, 15%))};
  --#{$prefix}btn-active-color: #{$active-color};
  --#{$prefix}btn-active-bg: #{$active-background};
  --#{$prefix}btn-active-border-color: #{$active-border};
  --#{$prefix}btn-active-shadow: #{$btn-active-box-shadow};
  --#{$prefix}btn-disabled-color: #{$disabled-color};
  --#{$prefix}btn-disabled-bg: #{$disabled-background};
  --#{$prefix}btn-disabled-border-color: #{$disabled-border};
}

@mixin button-outline-variant(
  $color,
  $color-hover: color-contrast($color),
  $active-background: $color,
  $active-border: $color,
  $active-color: color-contrast($active-background)
) {
  --#{$prefix}btn-color: #{$color};
  --#{$prefix}btn-border-color: #{$color};
  --#{$prefix}btn-hover-color: #{$color-hover};
  --#{$prefix}btn-hover-bg: #{$active-background};
  --#{$prefix}btn-hover-border-color: #{$active-border};
  --#{$prefix}btn-focus-shadow-rgb: #{to-rgb($color)};
  --#{$prefix}btn-active-color: #{$active-color};
  --#{$prefix}btn-active-bg: #{$active-background};
  --#{$prefix}btn-active-border-color: #{$active-border};
  --#{$prefix}btn-active-shadow: #{$btn-active-box-shadow};
  --#{$prefix}btn-disabled-color: #{$color};
  --#{$prefix}btn-disabled-bg: transparent;
  --#{$prefix}gradient: none;
}

@mixin button-size($padding-y, $padding-x, $font-size, $border-radius) {
  --#{$prefix}btn-padding-y: #{$padding-y};
  --#{$prefix}btn-padding-x: #{$padding-x};
  @include rfs($font-size, --#{$prefix}btn-font-size);
  --#{$prefix}btn-border-radius: #{$border-radius};
}
```

Sass loops

Button variants (for regular and outline buttons) use their respective mixins with our `$theme-colors` map to generate the modifier classes in `scss/_buttons.scss`.

```
@each $color, $value in $theme-colors {
  .btn-#{$color} {
    @include button-variant($value, $value);
  }
}

@each $color, $value in $theme-colors {
  .btn-outline-#{$color} {
    @include button-outline-variant($value);
  }
}
```

Button group

Group a series of buttons together on a single line or stack them in a vertical column.

On this page

Basic example

Wrap a series of buttons with `.btn` in `.btn-group`.

html

```
<div class="btn-group" role="group" aria-label="Basic example">
  <button type="button" class="btn btn-primary">Left</button>
  <button type="button" class="btn btn-primary">Middle</button>
  <button type="button" class="btn btn-primary">Right</button>
</div>
```

Ensure correct `role` and provide a label

In order for assistive technologies (such as screen readers) to convey that a series of buttons is grouped, an appropriate `role` attribute needs to be provided. For button groups, this would be `role="group"`, while toolbars should have a `role="toolbar"`.

In addition, groups and toolbars should be given an explicit label, as most assistive technologies will otherwise not announce them, despite the presence of the correct role attribute. In the examples provided here, we use `aria-label`, but alternatives such as `aria-labelledby` can also be used.

These classes can also be added to groups of links, as an alternative to the `.nav` navigation components.

html

```
<div class="btn-group">
  <a href="#" class="btn btn-primary active" aria-current="page">Active link</a>
  <a href="#" class="btn btn-primary">Link</a>
  <a href="#" class="btn btn-primary">Link</a>
</div>
```

Mixed styles

html

```
<div class="btn-group" role="group" aria-label="Basic mixed styles example">
  <button type="button" class="btn btn-danger">Left</button>
  <button type="button" class="btn btn-warning">Middle</button>
  <button type="button" class="btn btn-success">Right</button>
</div>
```

Outlined styles

html

```
<div class="btn-group" role="group" aria-label="Basic outlined example">
  <button type="button" class="btn btn-outline-primary">Left</button>
  <button type="button" class="btn btn-outline-primary">Middle</button>
  <button type="button" class="btn btn-outline-primary">Right</button>
</div>
```

Checkbox and radio button groups

Combine button-like checkbox and radio toggle buttons into a seamless looking button group.

html

```
<div class="btn-group" role="group" aria-label="Basic checkbox toggle button
group">
  <input type="checkbox" class="btn-check" id="btncheck1" autocomplete="off">
  <label class="btn btn-outline-primary" for="btncheck1">Checkbox 1</label>

  <input type="checkbox" class="btn-check" id="btncheck2" autocomplete="off">
  <label class="btn btn-outline-primary" for="btncheck2">Checkbox 2</label>

  <input type="checkbox" class="btn-check" id="btncheck3" autocomplete="off">
  <label class="btn btn-outline-primary" for="btncheck3">Checkbox 3</label>
</div>
```

html

```
<div class="btn-group" role="group" aria-label="Basic radio toggle button
group">
  <input type="radio" class="btn-check" name="btnradio" id="btnradio1"
autocomplete="off" checked>
  <label class="btn btn-outline-primary" for="btnradio1">Radio 1</label>

  <input type="radio" class="btn-check" name="btnradio" id="btnradio2"
autocomplete="off">
  <label class="btn btn-outline-primary" for="btnradio2">Radio 2</label>

  <input type="radio" class="btn-check" name="btnradio" id="btnradio3"
autocomplete="off">
  <label class="btn btn-outline-primary" for="btnradio3">Radio 3</label>
</div>
```

Button toolbar

Combine sets of button groups into button toolbars for more complex components. Use utility classes as needed to space out groups, buttons, and more.

html

```
<div class="btn-toolbar" role="toolbar" aria-label="Toolbar with button groups">
```

```
  <div class="btn-group me-2" role="group" aria-label="First group">
    <button type="button" class="btn btn-primary">1</button>
    <button type="button" class="btn btn-primary">2</button>
    <button type="button" class="btn btn-primary">3</button>
    <button type="button" class="btn btn-primary">4</button>
  </div>
  <div class="btn-group me-2" role="group" aria-label="Second group">
    <button type="button" class="btn btn-secondary">5</button>
    <button type="button" class="btn btn-secondary">6</button>
    <button type="button" class="btn btn-secondary">7</button>
  </div>
  <div class="btn-group" role="group" aria-label="Third group">
    <button type="button" class="btn btn-info">8</button>
  </div>
</div>
```

Feel free to mix input groups with button groups in your toolbars. Similar to the example above, you’ll likely need some utilities though to space things properly.

@

@

html

```
<div class="btn-toolbar mb-3" role="toolbar" aria-label="Toolbar with button
groups">
  <div class="btn-group me-2" role="group" aria-label="First group">
    <button type="button" class="btn btn-outline-secondary">1</button>
    <button type="button" class="btn btn-outline-secondary">2</button>
    <button type="button" class="btn btn-outline-secondary">3</button>
    <button type="button" class="btn btn-outline-secondary">4</button>
  </div>
  <div class="input-group">
    <div class="input-group-text" id="btnGroupAddon">@</div>
    <input type="text" class="form-control" placeholder="Input group example"
aria-label="Input group example" aria-describedby="btnGroupAddon">
  </div>
</div>

<div class="btn-toolbar justify-content-between" role="toolbar" aria-
label="Toolbar with button groups">
  <div class="btn-group" role="group" aria-label="First group">
    <button type="button" class="btn btn-outline-secondary">1</button>
    <button type="button" class="btn btn-outline-secondary">2</button>
    <button type="button" class="btn btn-outline-secondary">3</button>
    <button type="button" class="btn btn-outline-secondary">4</button>
  </div>
  <div class="input-group">
    <div class="input-group-text" id="btnGroupAddon2">@</div>
    <input type="text" class="form-control" placeholder="Input group example"
aria-label="Input group example" aria-describedby="btnGroupAddon2">
  </div>
</div>
```

Sizing

Instead of applying button sizing classes to every button in a group, just add `.btn-group-*` to each `.btn-group`, including each one when nesting multiple groups.

```
<div class="btn-group btn-group-lg" role="group" aria-label="...">...</div>
<div class="btn-group" role="group" aria-label="...">...</div>
<div class="btn-group btn-group-sm" role="group" aria-label="...">...</div>
```

Nesting

Place a `.btn-group` within another `.btn-group` when you want dropdown menus mixed with a series of buttons.

html

```
<div class="btn-group" role="group" aria-label="Button group with nested
dropdown">
  <button type="button" class="btn btn-primary">1</button>
  <button type="button" class="btn btn-primary">2</button>

  <div class="btn-group" role="group">
    <button id="btnGroupDrop1" type="button" class="btn btn-primary dropdown-
toggle" data-bs-toggle="dropdown" aria-expanded="false">
      Dropdown
    </button>
    <ul class="dropdown-menu" aria-labelledby="btnGroupDrop1">
      <li><a class="dropdown-item" href="#">Dropdown link</a></li>
      <li><a class="dropdown-item" href="#">Dropdown link</a></li>
    </ul>
  </div>
</div>
```

Vertical variation

Make a set of buttons appear vertically stacked rather than horizontally. **Split button dropdowns are not supported here.**

```
<div class="btn-group-vertical">
  ...
</div>
```

Cards

Bootstrap’s cards provide a flexible and extensible content container with multiple variants and options.

On this page

About

A **card** is a flexible and extensible content container. It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options. If you’re familiar with Bootstrap 3, cards replace our old panels, wells, and thumbnails. Similar functionality to those components is available as modifier classes for cards.

Example

Cards are built with as little markup and styles as possible, but still manage to deliver a ton of control and customization. Built with flexbox, they offer easy alignment and mix well with other Bootstrap components. They have no `margin` by default, so use spacing utilities as needed.

Below is an example of a basic card with mixed content and a fixed width. Cards have no fixed width to start, so they'll naturally fill the full width of its parent element. This is easily customized with our various sizing options.

Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

html

```
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Content types

Cards support a wide variety of content, including images, text, list groups, links, and more. Below are examples of what's supported.

Body

The building block of a card is the `.card-body`. Use it whenever you need a padded section within a card.

This is some text within a card body.

html

```
<div class="card">
  <div class="card-body">
    This is some text within a card body.
  </div>
</div>
```

Titles, text, and links

Card titles are used by adding `.card-title` to a `<h*>` tag. In the same way, links are added and placed next to each other by adding `.card-link` to an `<a>` tag.

Subtitles are used by adding a `.card-subtitle` to a `<h*>` tag. If the `.card-title` and the `.card-subtitle` items are placed in a `.card-body` item, the card title and subtitle are aligned nicely.

Card title

Card subtitle

Some quick example text to build on the card title and make up the bulk of the card's content.

Card link Another link

html

```
<div class="card" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <h6 class="card-subtitle mb-2 text-muted">Card subtitle</h6>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
    <a href="#" class="card-link">Card link</a>
    <a href="#" class="card-link">Another link</a>
  </div>
</div>
```

Images

`.card-img-top` places an image to the top of the card. With `.card-text`, text can be added to the card. Text within `.card-text` can also be styled with the standard HTML tags.

Some quick example text to build on the card title and make up the bulk of the card's content.

html

```
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
```

List groups

Create lists of content in a card with a flush list group.

- An item
- A second item
- A third item

html

```
<div class="card" style="width: 18rem;">
  <ul class="list-group list-group-flush">
    <li class="list-group-item">An item</li>
    <li class="list-group-item">A second item</li>
    <li class="list-group-item">A third item</li>
  </ul>
</div>
```

Featured

- An item
- A second item
- A third item

html

```html
<div class="card" style="width: 18rem;">
  <div class="card-header">
    Featured
  </div>
  <ul class="list-group list-group-flush">
    <li class="list-group-item">An item</li>
    <li class="list-group-item">A second item</li>
    <li class="list-group-item">A third item</li>
  </ul>
</div>
```

- An item
- A second item
- A third item

Card footer

html

```html
<div class="card" style="width: 18rem;">
  <ul class="list-group list-group-flush">
    <li class="list-group-item">An item</li>
    <li class="list-group-item">A second item</li>
    <li class="list-group-item">A third item</li>
  </ul>
  <div class="card-footer">
    Card footer
  </div>
</div>
```

Kitchen sink

Mix and match multiple content types to create the card you need, or throw everything in there. Shown below are image styles, blocks, text styles, and a list group—all wrapped in a fixed-width card.

Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

- An item
- A second item
- A third item

Card link Another link

html

```html
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
  <ul class="list-group list-group-flush">
    <li class="list-group-item">An item</li>
    <li class="list-group-item">A second item</li>
    <li class="list-group-item">A third item</li>
  </ul>
  <div class="card-body">
    <a href="#" class="card-link">Card link</a>
    <a href="#" class="card-link">Another link</a>
  </div>
</div>
```

Header and footer

Add an optional header and/or footer within a card.

Featured

Special title treatment

With supporting text below as a natural lead-in to additional content.

html

```
<div class="card">
  <div class="card-header">
    Featured
  </div>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Card headers can be styled by adding `.card-header` to `<h*>` elements.

Featured

Special title treatment

With supporting text below as a natural lead-in to additional content.

html

```
<div class="card">
  <h5 class="card-header">Featured</h5>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Quote

> A well-known quote, contained in a blockquote element.
>
> *Source Title*

html

```
<div class="card">
  <div class="card-header">
    Quote
  </div>
  <div class="card-body">
    <blockquote class="blockquote mb-0">
      <p>A well-known quote, contained in a blockquote element.</p>
      <footer class="blockquote-footer">Someone famous in <cite title="Source
Title">Source Title</cite></footer>
    </blockquote>
  </div>
</div>
```

Featured

Special title treatment

With supporting text below as a natural lead-in to additional content.

2 days ago

html

```
<div class="card text-center">
  <div class="card-header">
    Featured
  </div>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
  <div class="card-footer text-muted">
    2 days ago
  </div>
</div>
```

Sizing

Cards assume no specific `width` to start, so they'll be 100% wide unless otherwise stated. You can change this as needed with custom CSS, grid classes, grid Sass mixins, or utilities.

Using grid markup

Using the grid, wrap cards in columns and rows as needed.

Special title treatment

With supporting text below as a natural lead-in to additional content.

Special title treatment

With supporting text below as a natural lead-in to additional content.

html

```
<div class="row">
  <div class="col-sm-6">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Special title treatment</h5>
        <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
    </div>
  </div>
  <div class="col-sm-6">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Special title treatment</h5>
        <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
    </div>
  </div>
</div>
```

Using utilities

Use our handful of available sizing utilities to quickly set a card’s width.

Card title

With supporting text below as a natural lead-in to additional content.

Card title

With supporting text below as a natural lead-in to additional content.

html

```
<div class="card w-75">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Button</a>
  </div>
</div>

<div class="card w-50">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Button</a>
  </div>
</div>
```

Using custom CSS

Use custom CSS in your stylesheets or as inline styles to set a width.

Special title treatment

With supporting text below as a natural lead-in to additional content.

html

```
<div class="card" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
```

```
</div>
```

Text alignment

You can quickly change the text alignment of any card—in its entirety or specific parts—with our text align classes.

Special title treatment

With supporting text below as a natural lead-in to additional content.

Special title treatment

With supporting text below as a natural lead-in to additional content.

Special title treatment

With supporting text below as a natural lead-in to additional content.

html

```
<div class="card" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>

<div class="card text-center" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>

<div class="card text-end" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Navigation

Add some navigation to a card’s header (or block) with Bootstrap’s nav components.

- Active
- Link
- Disabled

Special title treatment

With supporting text below as a natural lead-in to additional content.

html

```
<div class="card text-center">
  <div class="card-header">
    <ul class="nav nav-tabs card-header-tabs">
      <li class="nav-item">
        <a class="nav-link active" aria-current="true" href="#">Active</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled">Disabled</a>
      </li>
    </ul>
  </div>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

- Active
- Link
- Disabled

Special title treatment

With supporting text below as a natural lead-in to additional content.

html

```
<div class="card text-center">
  <div class="card-header">
    <ul class="nav nav-pills card-header-pills">
      <li class="nav-item">
        <a class="nav-link active" href="#">Active</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled">Disabled</a>
      </li>
    </ul>
  </div>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to
additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Images

Cards include a few options for working with images. Choose from appending “image caps” at either end of a card, overlaying images with card content, or simply embedding the image in a card.

Image caps

Similar to headers and footers, cards can include top and bottom “image caps”—images at the top or bottom of a card.

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

html

```
<div class="card mb-3">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">This is a wider card with supporting text below as a
natural lead-in to additional content. This content is a little bit longer.</p>
    <p class="card-text"><small class="text-muted">Last updated 3 mins
ago</small></p>
  </div>
</div>
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">This is a wider card with supporting text below as a
natural lead-in to additional content. This content is a little bit longer.</p>
    <p class="card-text"><small class="text-muted">Last updated 3 mins
ago</small></p>
  </div>
  <img src="..." class="card-img-bottom" alt="...">
</div>
```

Image overlays

Turn an image into a card background and overlay your card’s text. Depending on the image, you may or may not need additional styles or utilities.

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

html

```
<div class="card bg-dark text-white">
  <img src="..." class="card-img" alt="...">
  <div class="card-img-overlay">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">This is a wider card with supporting text below as a
natural lead-in to additional content. This content is a little bit longer.</p>
```

```
    <p class="card-text">Last updated 3 mins ago</p>
  </div>
</div>
```

Note that content should not be larger than the height of the image. If content is larger than the image the content will be displayed outside the image.

Horizontal

Using a combination of grid and utility classes, cards can be made horizontal in a mobile-friendly and responsive way. In the example below, we remove the grid gutters with `.g-0` and use `.col-md-*` classes to make the card horizontal at the `md` breakpoint. Further adjustments may be needed depending on your card content.

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

html

```
<div class="card mb-3" style="max-width: 540px;">
  <div class="row g-0">
    <div class="col-md-4">
      <img src="..." class="img-fluid rounded-start" alt="...">
    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a wider card with supporting text below as
a natural lead-in to additional content. This content is a little bit
longer.</p>
        <p class="card-text"><small class="text-muted">Last updated 3 mins
ago</small></p>
      </div>
    </div>
  </div>
</div>
```

Card styles

Cards include various options for customizing their backgrounds, borders, and color.

Background and color

Added in v5.2.0

Set a `background-color` with contrasting foreground `color` with [our `.text-bg-{color}` helpers](). Previously it was required to manually pair your choice of [`.text-{color}`]() and [`.bg-{color}`]() utilities for styling, which you still may use if you prefer.

Header

Primary card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Secondary card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Success card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Danger card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Warning card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Info card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Light card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Dark card title

Some quick example text to build on the card title and make up the bulk of the card's content.

html

```
<div class="card text-bg-primary mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Primary card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-bg-secondary mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Secondary card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-bg-success mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Success card title</h5>
    <p class="card-text">Some quick example text to build on the card title and 
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-bg-danger mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Danger card title</h5>
    <p class="card-text">Some quick example text to build on the card title and 
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-bg-warning mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Warning card title</h5>
    <p class="card-text">Some quick example text to build on the card title and 
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-bg-info mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Info card title</h5>
    <p class="card-text">Some quick example text to build on the card title and 
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-bg-light mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Light card title</h5>
    <p class="card-text">Some quick example text to build on the card title and 
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-bg-dark mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Dark card title</h5>
    <p class="card-text">Some quick example text to build on the card title and 
make up the bulk of the card's content.</p>
  </div>
</div>
```

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.

Border

Use border utilities to change just the `border-color` of a card. Note that you can put `.text-{color}` classes on the parent `.card` or a subset of the card’s contents as shown below.

Header

Primary card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Secondary card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Success card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Danger card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Warning card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Info card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Light card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Dark card title

Some quick example text to build on the card title and make up the bulk of the card's content.

html

```
<div class="card border-primary mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body text-primary">
    <h5 class="card-title">Primary card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card border-secondary mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body text-secondary">
    <h5 class="card-title">Secondary card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card border-success mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body text-success">
    <h5 class="card-title">Success card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card border-danger mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body text-danger">
    <h5 class="card-title">Danger card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card border-warning mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Warning card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card border-info mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Info card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card border-light mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Light card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card border-dark mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body text-dark">
    <h5 class="card-title">Dark card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
```

Mixins utilities

You can also change the borders on the card header and footer as needed, and even remove their `background-color` with `.bg-transparent`.

Header

Success card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Footer

html

```
<div class="card border-success mb-3" style="max-width: 18rem;">
```

```
  <div class="card-header bg-transparent border-success">Header</div>
  <div class="card-body text-success">
    <h5 class="card-title">Success card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
  <div class="card-footer bg-transparent border-success">Footer</div>
</div>
```

Card layout

In addition to styling the content within cards, Bootstrap includes a few options for laying out series of cards. For the time being, **these layout options are not yet responsive**.

Card groups

Use card groups to render cards as a single, attached element with equal width and height columns. Card groups start off stacked and use `display: flex;` to become attached with uniform dimensions starting at the `sm` breakpoint.

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

Card title

This card has supporting text below as a natural lead-in to additional content.

Last updated 3 mins ago

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This card has even longer content than the first to show that equal height action.

Last updated 3 mins ago

html

```
<div class="card-group">
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This is a wider card with supporting text below as a
natural lead-in to additional content. This content is a little bit longer.</p>
      <p class="card-text"><small class="text-muted">Last updated 3 mins
ago</small></p>
    </div>
  </div>
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This card has supporting text below as a natural
lead-in to additional content.</p>
      <p class="card-text"><small class="text-muted">Last updated 3 mins
ago</small></p>
```

```
    </div>
  </div>
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This is a wider card with supporting text below as a
natural lead-in to additional content. This card has even longer content than
the first to show that equal height action.</p>
      <p class="card-text"><small class="text-muted">Last updated 3 mins
ago</small></p>
    </div>
  </div>
</div>
```

When using card groups with footers, their content will automatically line up.

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

Card title

This card has supporting text below as a natural lead-in to additional content.

Last updated 3 mins ago

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This card has even longer content than the first to show that equal height action.

Last updated 3 mins ago

html

```
<div class="card-group">
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This is a wider card with supporting text below as a
natural lead-in to additional content. This content is a little bit longer.</p>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 3 mins ago</small>
    </div>
  </div>
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This card has supporting text below as a natural
lead-in to additional content.</p>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 3 mins ago</small>
    </div>
  </div>
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This is a wider card with supporting text below as a 
natural lead-in to additional content. This card has even longer content than 
the first to show that equal height action.</p>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 3 mins ago</small>
    </div>
  </div>
</div>
```

Grid cards

Use the Bootstrap grid system and its `.row-cols` classes to control how many grid columns (wrapped around your cards) you show per row. For example, here's `.row-cols-1` laying out the cards on one column, and `.row-cols-md-2` splitting four cards to equal width across multiple rows, from the medium breakpoint up.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

html

```
<div class="row row-cols-1 row-cols-md-2 g-4">
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as 
a natural lead-in to additional content. This content is a little bit 
longer.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as 
a natural lead-in to additional content. This content is a little bit 
longer.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as
a natural lead-in to additional content.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as
a natural lead-in to additional content. This content is a little bit
longer.</p>
      </div>
    </div>
  </div>
</div>
```

Change it to `.row-cols-3` and you'll see the fourth card wrap.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

html

```
<div class="row row-cols-1 row-cols-md-3 g-4">
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as
a natural lead-in to additional content. This content is a little bit
longer.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as
a natural lead-in to additional content. This content is a little bit
longer.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as
a natural lead-in to additional content.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as
a natural lead-in to additional content. This content is a little bit
longer.</p>
      </div>
    </div>
  </div>
</div>
```

When you need equal height, add `.h-100` to the cards. If you want equal heights by default, you can set `$card-height: 100%` in Sass.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Card title

This is a short card.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content.

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

html

```
<div class="row row-cols-1 row-cols-md-3 g-4">
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as
a natural lead-in to additional content. This content is a little bit
longer.</p>
```

```
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a short card.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as
a natural lead-in to additional content.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as
a natural lead-in to additional content. This content is a little bit
longer.</p>
      </div>
    </div>
  </div>
</div>
```

Just like with card groups, card footers will automatically line up.

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

Card title

This card has supporting text below as a natural lead-in to additional content.

Last updated 3 mins ago

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This card has even longer content than the first to show that equal height action.

Last updated 3 mins ago

html

```
<div class="row row-cols-1 row-cols-md-3 g-4">
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
```

```
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a wider card with supporting text below as
a natural lead-in to additional content. This content is a little bit
longer.</p>
      </div>
      <div class="card-footer">
        <small class="text-muted">Last updated 3 mins ago</small>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This card has supporting text below as a natural
lead-in to additional content.</p>
      </div>
      <div class="card-footer">
        <small class="text-muted">Last updated 3 mins ago</small>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a wider card with supporting text below as
a natural lead-in to additional content. This card has even longer content than
the first to show that equal height action.</p>
      </div>
      <div class="card-footer">
        <small class="text-muted">Last updated 3 mins ago</small>
      </div>
    </div>
  </div>
</div>
```

Masonry

In `v4` we used a CSS-only technique to mimic the behavior of Masonry-like columns, but this technique came with lots of unpleasant side effects. If you want to have this type of layout in `v5`, you can just make use of Masonry plugin. **Masonry is not included in Bootstrap**, but we've made a demo example to help you get started.

CSS

Variables

Added in v5.2.0

As part of Bootstrap's evolving CSS variables approach, cards now use local CSS variables on `.card` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}card-spacer-y: #{$card-spacer-y};
  --#{$prefix}card-spacer-x: #{$card-spacer-x};
  --#{$prefix}card-title-spacer-y: #{$card-title-spacer-y};
  --#{$prefix}card-border-width: #{$card-border-width};
  --#{$prefix}card-border-color: #{$card-border-color};
  --#{$prefix}card-border-radius: #{$card-border-radius};
  --#{$prefix}card-box-shadow: #{$card-box-shadow};
  --#{$prefix}card-inner-border-radius: #{$card-inner-border-radius};
  --#{$prefix}card-cap-padding-y: #{$card-cap-padding-y};
  --#{$prefix}card-cap-padding-x: #{$card-cap-padding-x};
  --#{$prefix}card-cap-bg: #{$card-cap-bg};
  --#{$prefix}card-cap-color: #{$card-cap-color};
  --#{$prefix}card-height: #{$card-height};
  --#{$prefix}card-color: #{$card-color};
  --#{$prefix}card-bg: #{$card-bg};
  --#{$prefix}card-img-overlay-padding: #{$card-img-overlay-padding};
  --#{$prefix}card-group-margin: #{$card-group-margin};
```

Sass variables

```
$card-spacer-y:                     $spacer;
$card-spacer-x:                     $spacer;
$card-title-spacer-y:               $spacer * .5;
$card-border-width:                 $border-width;
$card-border-color:                 var(--#{$prefix}border-color-translucent);
$card-border-radius:                $border-radius;
$card-box-shadow:                   null;
$card-inner-border-radius:          subtract($card-border-radius, $card-border-
width);
$card-cap-padding-y:                $card-spacer-y * .5;
$card-cap-padding-x:                $card-spacer-x;
$card-cap-bg:                       rgba($black, .03);
$card-cap-color:                    null;
$card-height:                       null;
$card-color:                        null;
$card-bg:                           $white;
$card-img-overlay-padding:          $spacer;
$card-group-margin:                 $grid-gutter-width * .5;
```

Carousel

A slideshow component for cycling through elements—images or slides of text—like a carousel.

On this page

How it works

The carousel is a slideshow for cycling through a series of content, built with CSS 3D transforms and a bit of JavaScript. It works with a series of images, text, or custom markup. It also includes support for previous/next controls and indicators.

In browsers where the Page Visibility API is supported, the carousel will avoid sliding when the webpage is not visible to the user (such as when the browser tab is inactive, the browser window is minimized, etc.).

The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the reduced motion section of our accessibility documentation.

Please be aware that nested carousels are not supported, and carousels are generally not compliant with accessibility standards.

Example

Carousels don’t automatically normalize slide dimensions. As such, you may need to use additional utilities or custom styles to appropriately size content. While carousels support previous/next controls and indicators, they’re not explicitly required. Add and customize as you see fit.

The `.active` class needs to be added to one of the slides otherwise the carousel will not be visible. Also be sure to set a unique `id` on the `.carousel` for optional controls, especially if you're using multiple carousels on a single page. Control and indicator elements must have a `data-bs-target` attribute (or `href` for links) that matches the `id` of the `.carousel` element.

Slides only

Here's a carousel with slides only. Note the presence of the `.d-block` and `.w-100` on carousel images to prevent browser default image alignment.

html

```
<div id="carouselExampleSlidesOnly" class="carousel slide" data-bs-
ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
</div>
```

With controls

Adding in the previous and next controls. We recommend using `<button>` elements, but you can also use `<a>` elements with `role="button"`.

html

```
<div id="carouselExampleControls" class="carousel slide" data-bs-
ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleControls" data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleControls" data-bs-slide="next">
```

```
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

With indicators

You can also add the indicators to the carousel, alongside the controls, too.

html

```
<div id="carouselExampleIndicators" class="carousel slide" data-bs-ride="true">
  <div class="carousel-indicators">
    <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-
slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></button>
    <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-
slide-to="1" aria-label="Slide 2"></button>
    <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-
slide-to="2" aria-label="Slide 3"></button>
  </div>
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleIndicators" data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleIndicators" data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

With captions

Add captions to your slides easily with the `.carousel-caption` element within any `.carousel-item`. They can be easily hidden on smaller viewports, as shown below, with optional display utilities. We hide them initially with `.d-none` and bring them back on medium-sized devices with `.d-md-block`.

First slide label

Some representative placeholder content for the first slide.

html

```
<div id="carouselExampleCaptions" class="carousel slide" data-bs-ride="false">
```

```
  <div class="carousel-indicators">
    <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-
slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></button>
    <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-
slide-to="1" aria-label="Slide 2"></button>
    <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-
slide-to="2" aria-label="Slide 3"></button>
  </div>
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>First slide label</h5>
        <p>Some representative placeholder content for the first slide.</p>
      </div>
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>Second slide label</h5>
        <p>Some representative placeholder content for the second slide.</p>
      </div>
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>Third slide label</h5>
        <p>Some representative placeholder content for the third slide.</p>
      </div>
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleCaptions" data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleCaptions" data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

Crossfade

Add `.carousel-fade` to your carousel to animate slides with a fade transition instead of a slide. Depending on your carousel content (e.g., text only slides), you may want to add `.bg-body` or some custom CSS to the `.carousel-item`s for proper crossfading.

html

```
<div id="carouselExampleFade" class="carousel slide carousel-fade" data-bs-
ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleFade" data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleFade" data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

Individual `.carousel-item` interval

Add `data-bs-interval=""` to a `.carousel-item` to change the amount of time to delay between automatically cycling to the next item.

html

```
<div id="carouselExampleInterval" class="carousel slide" data-bs-
ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active" data-bs-interval="10000">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item" data-bs-interval="2000">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleInterval" data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleInterval" data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

Disable touch swiping

Carousels support swiping left/right on touchscreen devices to move between slides. This can be disabled using the `data-bs-touch` attribute. The example below also does not include the `data-bs-ride` attribute and has `data-bs-interval="false"` so it doesn’t autoplay.

html

```
<div id="carouselExampleControlsNoTouching" class="carousel slide" data-bs-
touch="false" data-bs-interval="false">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleControlsNoTouching" data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleControlsNoTouching" data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

Dark variant

Add `.carousel-dark` to the `.carousel` for darker controls, indicators, and captions. Controls have been inverted from their default white fill with the `filter` CSS property. Captions and controls have additional Sass variables that customize the `color` and `background-color`.

First slide label

Some representative placeholder content for the first slide.

html

```
<div id="carouselExampleDark" class="carousel carousel-dark slide" data-bs-
ride="carousel">
  <div class="carousel-indicators">
    <button type="button" data-bs-target="#carouselExampleDark" data-bs-slide-
to="0" class="active" aria-current="true" aria-label="Slide 1"></button>
    <button type="button" data-bs-target="#carouselExampleDark" data-bs-slide-
to="1" aria-label="Slide 2"></button>
    <button type="button" data-bs-target="#carouselExampleDark" data-bs-slide-
to="2" aria-label="Slide 3"></button>
  </div>
  <div class="carousel-inner">
    <div class="carousel-item active" data-bs-interval="10000">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>First slide label</h5>
        <p>Some representative placeholder content for the first slide.</p>
      </div>
    </div>
    <div class="carousel-item" data-bs-interval="2000">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>Second slide label</h5>
        <p>Some representative placeholder content for the second slide.</p>
      </div>
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>Third slide label</h5>
        <p>Some representative placeholder content for the third slide.</p>
      </div>
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleDark" data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleDark" data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

Custom transition

The transition duration of `.carousel-item` can be changed with the `$carousel-transition-duration` Sass variable before compiling or custom styles if you're using the compiled CSS. If multiple transitions are applied, make sure the transform transition is defined first (eg. `transition: transform 2s ease, opacity .5s ease-out`).

Sass

Variables

```
$carousel-control-color:             $white;
$carousel-control-width:             15%;
$carousel-control-opacity:           .5;
$carousel-control-hover-opacity:     .9;
$carousel-control-transition:        opacity .15s ease;

$carousel-indicator-width:           30px;
$carousel-indicator-height:          3px;
$carousel-indicator-hit-area-height: 10px;
$carousel-indicator-spacer:          3px;
$carousel-indicator-opacity:         .5;
$carousel-indicator-active-bg:       $white;
$carousel-indicator-active-opacity:  1;
$carousel-indicator-transition:      opacity .6s ease;

$carousel-caption-width:             70%;
$carousel-caption-color:             $white;
$carousel-caption-padding-y:         1.25rem;
$carousel-caption-spacer:            1.25rem;

$carousel-control-icon-width:        2rem;

$carousel-control-prev-icon-bg:      url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 16 16' fill='#{$carousel-
control-color}'><path d='M11.354 1.646a.5.5 0 0 1 0 .708L5.707 8l5.647
5.646a.5.5 0 0 1-.708.708l-6-6a.5.5 0 0 1 0-.708l6-6a.5.5 0 0 1 .708
0z'/></svg>");
$carousel-control-next-icon-bg:      url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 16 16' fill='#{$carousel-
control-color}'><path d='M4.646 1.646a.5.5 0 0 1 .708 0l6 6a.5.5 0 0 1 0 .708l-6
6a.5.5 0 0 1-.708-.708L10.293 8 4.646 2.354a.5.5 0 0 1 0-.708z'/></svg>");

$carousel-transition-duration:       .6s;
$carousel-transition:                transform $carousel-transition-duration
ease-in-out; // Define transform transition first if using multiple transitions
(e.g., `transform 2s ease, opacity .5s ease-out`)

$carousel-dark-indicator-active-bg:  $black;
$carousel-dark-caption-color:        $black;
$carousel-dark-control-icon-filter:  invert(1) grayscale(100);
```

Usage

Via data attributes

Use data attributes to easily control the position of the carousel. `data-bs-slide` accepts the keywords `prev` or `next`, which alters the slide position relative to its current position. Alternatively, use `data-bs-slide-to` to pass a raw slide index to the carousel `data-bs-slide-to="2"`, which shifts the slide position to a particular index beginning with `0`.

The `data-bs-ride="carousel"` attribute is used to mark a carousel as animating starting at page load. If you don't use `data-bs-ride="carousel"` to initialize your carousel, you have to initialize it yourself. **It cannot be used in combination with (redundant and unnecessary) explicit JavaScript initialization of the same carousel.**

Via JavaScript

Call carousel manually with:

```
const carousel = new bootstrap.Carousel('#myCarousel')
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-bs-`, as in `data-bs-animation=""`. Make sure to change the case type of the option name from camelCase to kebab-case when passing the options via data attributes. For example, use `data-bs-custom-class="beautifier"` instead of `data-bs-customClass="beautifier"`.

As of Bootstrap 5.2.0, all components support an **experimental** reserved data attribute `data-bs-config` that can house simple component configuration as a JSON string. When an element has `data-bs-config='{"delay":0, "title":123}'` and `data-bs-title="456"` attributes, the final `title` value will be `456` and the separate data attributes will override values given on `data-bs-config`. In addition, existing data attributes are able to house JSON values like `data-bs-delay='{"show":0,"hide":150}'`.

Name	Type	Default	Description
`interval`	number	`5000`	The amount of time to delay between automatically cycling an item. If `false`, carousel will not automatically cycle.
`keyboard`	boolean	`true`	Whether the carousel should react to keyboard events.
`pause`	string, boolean	`"hover"`	If set to `"hover"`, pauses the cycling of the carousel on `mouseenter` and resumes the cycling of the carousel on `mouseleave`. If set to `false`, hovering over the carousel won't pause it. On touch-enabled devices, when set to `"hover"`, cycling will pause on `touchend` (once the user finished interacting with the carousel) for two intervals, before automatically resuming. This is in addition to the mouse behavior.
`ride`	string, boolean	`false`	If set to `true`, autoplays the carousel after the user manually cycles the first item. If set to `"carousel"`, autoplays the carousel on load.
`touch`	boolean	`true`	Whether the carousel should support left/right swipe interactions on touchscreen devices.
`wrap`	boolean	`true`	Whether the carousel should cycle continuously or have hard stops.

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.

You can create a carousel instance with the carousel constructor, for example, to initialize with additional options and start cycling through items:

```
const myCarouselElement = document.querySelector('#myCarousel')
const carousel = new bootstrap.Carousel(myCarouselElement, {
  interval: 2000,
  wrap: false
})
```

Method	Description
`cycle`	Cycles through the carousel items from left to right.
`pause`	Stops the carousel from cycling through items.
`prev`	Cycles to the previous item. **Returns to the caller before the previous item has been shown** (e.g., before the `slid.bs.carousel` event occurs).
`next`	Cycles to the next item. **Returns to the caller before the next item has been shown** (e.g., before the `slid.bs.carousel` event occurs).
`nextWhenVisible`	Don't cycle carousel to next when the page isn't visible or the carousel or its parent isn't visible. **Returns to the caller before the target item has been shown**
`to`	Cycles the carousel to a particular frame (0 based, similar to an

Method	Description
	array). **Returns to the caller before the target item has been shown** (e.g., before the `slid.bs.carousel` event occurs).
`dispose`	Destroys an element’s carousel. (Removes stored data on the DOM element)
`getInstance`	Static method which allows you to get the carousel instance associated to a DOM element, you can use it like this: `bootstrap.Carousel.getInstance(element)`
`getOrCreateInstance`	Static method which returns a carousel instance associated to a DOM element or create a new one in case it wasn’t initialized. You can use it like this: `bootstrap.Carousel.getOrCreateInstance(element)`

Events

Bootstrap’s carousel class exposes two events for hooking into carousel functionality. Both events have the following additional properties:

- `direction`: The direction in which the carousel is sliding (either `"left"` or `"right"`).
- `relatedTarget`: The DOM element that is being slid into place as the active item.
- `from`: The index of the current item
- `to`: The index of the next item

All carousel events are fired at the carousel itself (i.e. at the `<div class="carousel">`).

Event type	Description
`slide.bs.carousel`	Fires immediately when the `slide` instance method is invoked.
`slid.bs.carousel`	Fired when the carousel has completed its slide transition.

```
const myCarousel = document.getElementById('myCarousel')

myCarousel.addEventListener('slide.bs.carousel', event => {
  // do something...
})
```

Close button

A generic close button for dismissing content like modals and alerts.

On this page

Example

Provide an option to dismiss or close a component with `.btn-close`. Default styling is limited, but highly customizable. Modify the Sass variables to replace the default `background-image`. **Be sure to include text for screen readers**, as we've done with `aria-label`.

html

```
<button type="button" class="btn-close" aria-label="Close"></button>
```

Disabled state

Disabled close buttons change their `opacity`. We've also applied `pointer-events: none` and `user-select: none` to preventing hover and active states from triggering.

html

```
<button type="button" class="btn-close" disabled aria-label="Close"></button>
```

White variant

Change the default `.btn-close` to be white with the `.btn-close-white` class. This class uses the `filter` property to invert the `background-image`.

html

```
<button type="button" class="btn-close btn-close-white"
aria-label="Close"></button>
<button type="button" class="btn-close btn-close-white" disabled aria-
label="Close"></button>
```

Sass

Variables

```
$btn-close-width:            1em;
$btn-close-height:           $btn-close-width;
$btn-close-padding-x:        .25em;
```

```
$btn-close-padding-y:        $btn-close-padding-x;
$btn-close-color:            $black;
$btn-close-bg:               url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 16 16' fill='#{$btn-close-
color}'><path d='M.293.293a1 1 0 0 1 1.414 0L8 6.586 14.293.293a1 1 0 1 1 1.414
1.414L9.414 8l6.293 6.293a1 1 0 0 1-1.414 1.414L8 9.414l-6.293 6.293a1 1 0 0 1-
1.414-1.414L6.586 8 .293 1.707a1 1 0 0 1 0-1.414z'/></svg>");
$btn-close-focus-shadow:     $input-btn-focus-box-shadow;
$btn-close-opacity:          .5;
$btn-close-hover-opacity:    .75;
$btn-close-focus-opacity:    1;
$btn-close-disabled-opacity: .25;
$btn-close-white-filter:     invert(1) grayscale(100%) brightness(200%);
```

Collapse

Toggle the visibility of content across your project with a few classes and our JavaScript plugins.

On this page

How it works

The collapse JavaScript plugin is used to show and hide content. Buttons or anchors are used as triggers that are mapped to specific elements you toggle. Collapsing an element will animate the `height` from its current value to `0`. Given how CSS handles animations, you cannot use `padding` on a `.collapse` element. Instead, use the class as an independent wrapping element.

The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the reduced motion section of our accessibility documentation.

Example

Click the buttons below to show and hide another element via class changes:

- `.collapse` hides content
- `.collapsing` is applied during transitions
- `.collapse.show` shows content

Generally, we recommend using a button with the `data-bs-target` attribute. While not recommended from a semantic point of view, you can also use a link with the `href` attribute (and a `role="button"`). In both cases, the `data-bs-toggle="collapse"` is required.

html

```
<p>
  <a class="btn btn-primary" data-bs-toggle="collapse" href="#collapseExample"
role="button" aria-expanded="false" aria-controls="collapseExample">
    Link with href
  </a>
  <button class="btn btn-primary" type="button" data-bs-toggle="collapse" data-
bs-target="#collapseExample" aria-expanded="false" aria-
controls="collapseExample">
    Button with data-bs-target
  </button>
</p>
<div class="collapse" id="collapseExample">
  <div class="card card-body">
    Some placeholder content for the collapse component. This panel is hidden by
default but revealed when the user activates the relevant trigger.
  </div>
</div>
```

Horizontal

The collapse plugin also supports horizontal collapsing. Add the `.collapse-horizontal` modifier class to transition the `width` instead of `height` and set a `width` on the immediate child element. Feel free to write your own custom Sass, use inline styles, or use our width utilities.

Please note that while the example below has a `min-height` set to avoid excessive repaints in our docs, this is not explicitly required. **Only the `width` on the child element is required.**

html

```
<p>
  <button class="btn btn-primary" type="button" data-bs-toggle="collapse" data-
bs-target="#collapseWidthExample" aria-expanded="false" aria-
controls="collapseWidthExample">
    Toggle width collapse
  </button>
</p>
<div style="min-height: 120px;">
  <div class="collapse collapse-horizontal" id="collapseWidthExample">
    <div class="card card-body" style="width: 300px;">
      This is some placeholder content for a horizontal collapse. It's hidden by
default and shown when triggered.
    </div>
  </div>
</div>
```

Multiple targets

A `<button>` or `<a>` can show and hide multiple elements by referencing them with a selector in its `href` or `data-bs-target` attribute. Multiple `<button>` or `<a>` can show and hide an element if they each reference it with their `href` or `data-bs-target` attribute

html

```
<p>
  <a class="btn btn-primary" data-bs-toggle="collapse"
href="#multiCollapseExample1" role="button" aria-expanded="false" aria-
controls="multiCollapseExample1">Toggle first element</a>
  <button class="btn btn-primary" type="button" data-bs-toggle="collapse" data-
bs-target="#multiCollapseExample2" aria-expanded="false" aria-
controls="multiCollapseExample2">Toggle second element</button>
  <button class="btn btn-primary" type="button" data-bs-toggle="collapse" data-
bs-target=".multi-collapse" aria-expanded="false" aria-
```

```
controls="multiCollapseExample1 multiCollapseExample2">Toggle both
elements</button>
</p>
<div class="row">
  <div class="col">
    <div class="collapse multi-collapse" id="multiCollapseExample1">
      <div class="card card-body">
        Some placeholder content for the first collapse component of this multi-
collapse example. This panel is hidden by default but revealed when the user
activates the relevant trigger.
      </div>
    </div>
  </div>
  <div class="col">
    <div class="collapse multi-collapse" id="multiCollapseExample2">
      <div class="card card-body">
        Some placeholder content for the second collapse component of this
multi-collapse example. This panel is hidden by default but revealed when the
user activates the relevant trigger.
      </div>
    </div>
  </div>
</div>
```

Accessibility

Be sure to add `aria-expanded` to the control element. This attribute explicitly conveys the current state of the collapsible element tied to the control to screen readers and similar assistive technologies. If the collapsible element is closed by default, the attribute on the control element should have a value of `aria-expanded="false"`. If you've set the collapsible element to be open by default using the `show` class, set `aria-expanded="true"` on the control instead. The plugin will automatically toggle this attribute on the control based on whether or not the collapsible element has been opened or closed (via JavaScript, or because the user triggered another control element also tied to the same collapsible element). If the control element's HTML element is not a button (e.g., an `<a>` or `<div>`), the attribute `role="button"` should be added to the element.

If your control element is targeting a single collapsible element – i.e. the `data-bs-target` attribute is pointing to an `id` selector – you should add the `aria-controls` attribute to the control element, containing the `id` of the collapsible element. Modern screen readers and similar assistive technologies make use of this attribute to provide users with additional shortcuts to navigate directly to the collapsible element itself.

Note that Bootstrap's current implementation does not cover the various *optional* keyboard interactions described in the WAI-ARIA Authoring Practices 1.1 accordion pattern - you will need to include these yourself with custom JavaScript.

Sass

Variables

```
$transition-collapse:         height .35s ease;
$transition-collapse-width:   width .35s ease;
```

Classes

Collapse transition classes can be found in `scss/_transitions.scss` as these are shared across multiple components (collapse and accordion).

```
.collapse {
  &:not(.show) {
    display: none;
  }
}

.collapsing {
  height: 0;
  overflow: hidden;
  @include transition($transition-collapse);

  &.collapse-horizontal {
    width: 0;
    height: auto;
    @include transition($transition-collapse-width);
  }
}
```

Usage

The collapse plugin utilizes a few classes to handle the heavy lifting:

- `.collapse` hides the content
- `.collapse.show` shows the content
- `.collapsing` is added when the transition starts, and removed when it finishes

These classes can be found in `_transitions.scss`.

Via data attributes

Just add `data-bs-toggle="collapse"` and a `data-bs-target` to the element to automatically assign control of one or more collapsible elements. The `data-bs-target` attribute accepts a CSS selector to apply the collapse to. Be sure to add the class `collapse` to the collapsible element. If you'd like it to default open, add the additional class `show`.

To add accordion-like group management to a collapsible area, add the data attribute `data-bs-parent="#selector"`. Refer to the accordion page for more information.

Via JavaScript

Enable manually with:

```
const collapseElementList = document.querySelectorAll('.collapse')
const collapseList = [...collapseElementList].map(collapseEl => new
bootstrap.Collapse(collapseEl))
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-bs-`, as in `data-bs-animation=""`. Make sure to change the case type of the option name from camelCase to kebab-case when passing the options via data attributes. For example, use `data-bs-custom-class="beautifier"` instead of `data-bs-customClass="beautifier"`.

As of Bootstrap 5.2.0, all components support an **experimental** reserved data attribute `data-bs-config` that can house simple component configuration as a JSON string. When an element has `data-bs-config='{"delay":0, "title":123}'` and `data-bs-title="456"` attributes, the final `title` value will be `456` and the separate data attributes will override values given on `data-bs-config`. In addition, existing data attributes are able to house JSON values like `data-bs-delay='{"show":0,"hide":150}'`.

Name	Type	Default	Description
`parent`	selector, jQuery object, DOM element	`false`	If parent is provided, then all collapsible elements under the specified parent will be closed when this collapsible item is shown. (similar to traditional accordion behavior - this is dependent on the `card` class). The attribute has to be set on the target collapsible area.
`toggle`	boolean	`true`	Toggles the collapsible element on invocation

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.

Activates your content as a collapsible element. Accepts an optional options `object`.

You can create a collapse instance with the constructor, for example:

```
const bsCollapse = new bootstrap.Collapse('#myCollapse', {
  toggle: false
})
```

Method	Description
`toggle`	Toggles a collapsible element to shown or hidden. **Returns to the caller before the collapsible element has actually been shown or hidden** (i.e. before the `shown.bs.collapse` or `hidden.bs.collapse` event occurs).
`show`	Shows a collapsible element. **Returns to the caller before the collapsible element has actually been shown** (e.g., before the `shown.bs.collapse` event occurs).
`hide`	Hides a collapsible element. **Returns to the caller before the collapsible element has actually been hidden** (e.g., before the `hidden.bs.collapse` event occurs).
`dispose`	Destroys an element’s collapse. (Removes stored data on the DOM element)
`getInstance`	Static method which allows you to get the collapse instance associated to a DOM element, you can use it like this:

Method	Description
	`bootstrap.Collapse.getInstance(element)`
`getOrCreateInstance`	Static method which returns a collapse instance associated to a DOM element or create a new one in case it wasn’t initialized. You can use it like this: `bootstrap.Collapse.getOrCreateInstance(element)`

Events

Bootstrap’s collapse class exposes a few events for hooking into collapse functionality.

Event type	Description
`show.bs.collapse`	This event fires immediately when the `show` instance method is called.
`shown.bs.collapse`	This event is fired when a collapse element has been made visible to the user (will wait for CSS transitions to complete).
`hide.bs.collapse`	This event is fired immediately when the `hide` method has been called.
`hidden.bs.collapse`	This event is fired when a collapse element has been hidden from the user (will wait for CSS transitions to complete).

```
const myCollapsible = document.getElementById('myCollapsible')
myCollapsible.addEventListener('hidden.bs.collapse', event => {
  // do something...
})
```

Dropdowns

Toggle contextual overlays for displaying lists of links and more with the Bootstrap dropdown plugin.

On this page

Overview

Dropdowns are toggleable, contextual overlays for displaying lists of links and more. They're made interactive with the included Bootstrap dropdown JavaScript plugin. They're toggled by clicking, not by hovering; this is an intentional design decision.

Dropdowns are built on a third party library, Popper, which provides dynamic positioning and viewport detection. Be sure to include popper.min.js before Bootstrap's JavaScript or use `bootstrap.bundle.min.js` / `bootstrap.bundle.js` which contains Popper. Popper isn't used to position dropdowns in navbars though as dynamic positioning isn't required.

Accessibility

The standard defines an actual `role="menu"` widget, but this is specific to application-like menus which trigger actions or functions. menus can only contain menu items, checkbox menu items, radio button menu items, radio button groups, and sub-menus.

Bootstrap's dropdowns, on the other hand, are designed to be generic and applicable to a variety of situations and markup structures. For instance, it is possible to create dropdowns that contain additional inputs and form controls, such as search fields or login forms. For this reason, Bootstrap does not expect (nor automatically add) any of the `role` and `aria-` attributes required for true menus. Authors will have to include these more specific attributes themselves.

However, Bootstrap does add built-in support for most standard keyboard menu interactions, such as the ability to move through individual `.dropdown-item` elements using the cursor keys and close the menu with the `ESC` key.

Examples

Wrap the dropdown's toggle (your button or link) and the dropdown menu within `.dropdown`, or another element that declares `position: relative;`. Dropdowns can be triggered from `<a>` or `<button>` elements to better fit your potential needs. The examples shown here use semantic `<ul>` elements where appropriate, but custom markup is supported.

Single button

Any single `.btn` can be turned into a dropdown toggle with some markup changes. Here's how you can put them to work with either `<button>` elements:

html

```html
<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-expanded="false">
    Dropdown button
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
  </ul>
</div>
```

And with `<a>` elements:

html

```
<div class="dropdown">
  <a class="btn btn-secondary dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-bs-toggle="dropdown" aria-expanded="false">
    Dropdown link
  </a>

  <ul class="dropdown-menu" aria-labelledby="dropdownMenuLink">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
  </ul>
</div>
```

The best part is you can do this with any button variant, too:

```
<!-- Example single danger button -->
<div class="btn-group">
  <button type="button" class="btn btn-danger dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Action
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>
```

Split button

Similarly, create split button dropdowns with virtually the same markup as single button dropdowns, but with the addition of `.dropdown-toggle-split` for proper spacing around the dropdown caret.

We use this extra class to reduce the horizontal `padding` on either side of the caret by 25% and remove the `margin-left` that's added for regular button dropdowns. Those extra changes keep the caret centered in the split button and provide a more appropriately sized hit area next to the main button.

```
<!-- Example split danger button -->
<div class="btn-group">
  <button type="button" class="btn btn-danger">Action</button>
  <button type="button" class="btn btn-danger dropdown-toggle dropdown-toggle-
split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>
```

Sizing

Button dropdowns work with buttons of all sizes, including default and split dropdown buttons.

```
<!-- Large button groups (default and split) -->
<div class="btn-group">
  <button class="btn btn-secondary btn-lg dropdown-toggle" type="button" data-
bs-toggle="dropdown" aria-expanded="false">
    Large button
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
<div class="btn-group">
  <button class="btn btn-secondary btn-lg" type="button">
    Large split button
  </button>
  <button type="button" class="btn btn-lg btn-secondary dropdown-toggle
dropdown-toggle-split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>

<div class="btn-group">
  <button class="btn btn-secondary btn-sm dropdown-toggle" type="button" data-
bs-toggle="dropdown" aria-expanded="false">
    Small button
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
<div class="btn-group">
  <button class="btn btn-secondary btn-sm" type="button">
    Small split button
  </button>
  <button type="button" class="btn btn-sm btn-secondary dropdown-toggle
dropdown-toggle-split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
```

Dark dropdowns

Opt into darker dropdowns to match a dark navbar or custom style by adding `.dropdown-menu-dark` onto an existing `.dropdown-menu`. No changes are required to the dropdown items.

html

```
<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenuButton2" data-bs-toggle="dropdown" aria-expanded="false">
    Dropdown button
  </button>
  <ul class="dropdown-menu dropdown-menu-dark" aria-
labelledby="dropdownMenuButton2">
    <li><a class="dropdown-item active" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>
```

And putting it to use in a navbar:

Navbar

- Dropdown

html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarNavDarkDropdown" aria-controls="navbarNavDarkDropdown" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavDarkDropdown">
      <ul class="navbar-nav">
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#"
id="navbarDarkDropdownMenuLink" role="button" data-bs-toggle="dropdown" aria-
expanded="false">
            Dropdown
          </a>
          <ul class="dropdown-menu dropdown-menu-dark" aria-
labelledby="navbarDarkDropdownMenuLink">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Directions

RTL

Directions are mirrored when using Bootstrap in RTL, meaning `.dropstart` will appear on the right side.

Centered

Make the dropdown menu centered below the toggle with `.dropdown-center` on the parent element.

html

```
<div class="dropdown-center">
  <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownCenterBtn" data-bs-toggle="dropdown" aria-expanded="false">
    Centered dropdown
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownCenterBtn">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Action two</a></li>
    <li><a class="dropdown-item" href="#">Action three</a></li>
  </ul>
</div>
```

Dropup

Trigger dropdown menus above elements by adding `.dropup` to the parent element.

```
<!-- Default dropup button -->
<div class="btn-group dropup">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropup
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>

<!-- Split dropup button -->
<div class="btn-group dropup">
  <button type="button" class="btn btn-secondary">
    Split dropup
  </button>
  <button type="button" class="btn btn-secondary dropdown-toggle dropdown-
toggle-split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>
```

Dropup centered

Make the dropup menu centered above the toggle with `.dropup-center` on the parent element.

html

```
<div class="dropup-center dropup">
  <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropupCenterBtn" data-bs-toggle="dropdown" aria-expanded="false">
    Centered dropup
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropupCenterBtn">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Action two</a></li>
    <li><a class="dropdown-item" href="#">Action three</a></li>
  </ul>
</div>
```

Dropend

Trigger dropdown menus at the right of the elements by adding `.dropend` to the parent element.

```
<!-- Default dropend button -->
<div class="btn-group dropend">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropend
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>

<!-- Split dropend button -->
<div class="btn-group dropend">
  <button type="button" class="btn btn-secondary">
    Split dropend
  </button>
  <button type="button" class="btn btn-secondary dropdown-toggle dropdown-
toggle-split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropend</span>
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>
```

Dropstart

Trigger dropdown menus at the left of the elements by adding `.dropstart` to the parent element.

```
<!-- Default dropstart button -->
<div class="btn-group dropstart">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropstart
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>

<!-- Split dropstart button -->
<div class="btn-group dropstart">
  <button type="button" class="btn btn-secondary dropdown-toggle dropdown-
toggle-split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropstart</span>
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
  <button type="button" class="btn btn-secondary">
    Split dropstart
  </button>
</div>
```

Menu items

You can use `<a>` or `<button>` elements as dropdown items.

html

```
<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenu2" data-bs-toggle="dropdown" aria-expanded="false">
    Dropdown
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenu2">
    <li><button class="dropdown-item" type="button">Action</button></li>
    <li><button class="dropdown-item" type="button">Another action</button></li>
    <li><button class="dropdown-item" type="button">Something else
here</button></li>
  </ul>
</div>
```

You can also create non-interactive dropdown items with `.dropdown-item-text`. Feel free to style further with custom CSS or text utilities.

- Dropdown item text
- Action
- Another action
- Something else here

html

```
<ul class="dropdown-menu">
  <li><span class="dropdown-item-text">Dropdown item text</span></li>
  <li><a class="dropdown-item" href="#">Action</a></li>
  <li><a class="dropdown-item" href="#">Another action</a></li>
  <li><a class="dropdown-item" href="#">Something else here</a></li>
</ul>
```

Active

Add `.active` to items in the dropdown to **style them as active**. To convey the active state to assistive technologies, use the `aria-current` attribute — using the `page` value for the current page, or `true` for the current item in a set.

- Regular link
- Active link
- Another link

html

```
<ul class="dropdown-menu">
  <li><a class="dropdown-item" href="#">Regular link</a></li>
  <li><a class="dropdown-item active" href="#" aria-current="true">Active
link</a></li>
  <li><a class="dropdown-item" href="#">Another link</a></li>
</ul>
```

Disabled

Add `.disabled` to items in the dropdown to **style them as disabled**.

- Regular link
- Disabled link

- Another link

html

```
<ul class="dropdown-menu">
  <li><a class="dropdown-item" href="#">Regular link</a></li>
  <li><a class="dropdown-item disabled">Disabled link</a></li>
  <li><a class="dropdown-item" href="#">Another link</a></li>
</ul>
```

Menu alignment

By default, a dropdown menu is automatically positioned 100% from the top and along the left side of its parent. You can change this with the directional `.drop*` classes, but you can also control them with additional modifier classes.

Add `.dropdown-menu-end` to a `.dropdown-menu` to right align the dropdown menu. Directions are mirrored when using Bootstrap in RTL, meaning `.dropdown-menu-end` will appear on the left side.

Heads up! Dropdowns are positioned thanks to Popper except when they are contained in a navbar.
html

```
<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Right-aligned menu example
  </button>
  <ul class="dropdown-menu dropdown-menu-end">
    <li><button class="dropdown-item" type="button">Action</button></li>
    <li><button class="dropdown-item" type="button">Another action</button></li>
    <li><button class="dropdown-item" type="button">Something else
here</button></li>
  </ul>
</div>
```

Responsive alignment

If you want to use responsive alignment, disable dynamic positioning by adding the `data-bs-display="static"` attribute and use the responsive variation classes.

To align **right** the dropdown menu with the given breakpoint or larger, add `.dropdown-menu{-sm|-md|-lg|-xl|-xxl}-end`.

html

```
<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" data-bs-display="static" aria-expanded="false">
    Left-aligned but right aligned when large screen
  </button>
  <ul class="dropdown-menu dropdown-menu-lg-end">
    <li><button class="dropdown-item" type="button">Action</button></li>
    <li><button class="dropdown-item" type="button">Another action</button></li>
    <li><button class="dropdown-item" type="button">Something else
here</button></li>
  </ul>
</div>
```

To align **left** the dropdown menu with the given breakpoint or larger, add `.dropdown-menu-end` and `.dropdown-menu{-sm|-md|-lg|-xl|-xxl}-start`.

html

```
<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" data-bs-display="static" aria-expanded="false">
    Right-aligned but left aligned when large screen
  </button>
  <ul class="dropdown-menu dropdown-menu-end dropdown-menu-lg-start">
    <li><button class="dropdown-item" type="button">Action</button></li>
    <li><button class="dropdown-item" type="button">Another action</button></li>
    <li><button class="dropdown-item" type="button">Something else
here</button></li>
  </ul>
</div>
```

Note that you don’t need to add a `data-bs-display="static"` attribute to dropdown buttons in navbars, since Popper isn’t used in navbars.

Alignment options

Taking most of the options shown above, here’s a small kitchen sink demo of various dropdown alignment options in one place.

html

```
<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenuButton" data-bs-toggle="dropdown" aria-expanded="false">
    Dropdown
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Right-aligned menu
  </button>
  <ul class="dropdown-menu dropdown-menu-end">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" data-bs-display="static" aria-expanded="false">
    Left-aligned, right-aligned lg
  </button>
  <ul class="dropdown-menu dropdown-menu-lg-end">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" data-bs-display="static" aria-expanded="false">
    Right-aligned, left-aligned lg
  </button>
  <ul class="dropdown-menu dropdown-menu-end dropdown-menu-lg-start">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group dropstart">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropstart
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group dropend">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropend
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group dropup">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropup
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>
```

Menu content

Headers

Add a header to label sections of actions in any dropdown menu.

- **Dropdown header**

- Action
- Another action

html

```
<ul class="dropdown-menu">
  <li><h6 class="dropdown-header">Dropdown header</h6></li>
  <li><a class="dropdown-item" href="#">Action</a></li>
  <li><a class="dropdown-item" href="#">Another action</a></li>
</ul>
```

Dividers

Separate groups of related menu items with a divider.

- Action
- Another action
- Something else here
-

- Separated link

html

```
<ul class="dropdown-menu">
  <li><a class="dropdown-item" href="#">Action</a></li>
  <li><a class="dropdown-item" href="#">Another action</a></li>
  <li><a class="dropdown-item" href="#">Something else here</a></li>
  <li><hr class="dropdown-divider"></li>
  <li><a class="dropdown-item" href="#">Separated link</a></li>
</ul>
```

Text

Place any freeform text within a dropdown menu with text and use spacing utilities. Note that you'll likely need additional sizing styles to constrain the menu width.

Some example text that's free-flowing within the dropdown menu.

And this is more example text.

html

```
<div class="dropdown-menu p-4 text-muted" style="max-width: 200px;">
  <p>
    Some example text that's free-flowing within the dropdown menu.
  </p>
  <p class="mb-0">
    And this is more example text.
  </p>
</div>
```

Forms

Put a form within a dropdown menu, or make it into a dropdown menu, and use margin or padding utilities to give it the negative space you require.

☐

New around here? Sign up Forgot password?

html

```
<div class="dropdown-menu">
  <form class="px-4 py-3">
    <div class="mb-3">
      <label for="exampleDropdownFormEmail1" class="form-label">Email
address</label>
      <input type="email" class="form-control" id="exampleDropdownFormEmail1"
placeholder="email@example.com">
    </div>
    <div class="mb-3">
      <label for="exampleDropdownFormPassword1"
class="form-label">Password</label>
      <input type="password" class="form-control"
id="exampleDropdownFormPassword1" placeholder="Password">
    </div>
    <div class="mb-3">
      <div class="form-check">
        <input type="checkbox" class="form-check-input" id="dropdownCheck">
        <label class="form-check-label" for="dropdownCheck">
          Remember me
        </label>
      </div>
    </div>
    <button type="submit" class="btn btn-primary">Sign in</button>
  </form>
  <div class="dropdown-divider"></div>
  <a class="dropdown-item" href="#">New around here? Sign up</a>
  <a class="dropdown-item" href="#">Forgot password?</a>
</div>
```

html

```
<div class="dropdown">
  <button type="button" class="btn btn-primary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false" data-bs-auto-close="outside">
    Dropdown form
  </button>
  <form class="dropdown-menu p-4">
    <div class="mb-3">
      <label for="exampleDropdownFormEmail2" class="form-label">Email
address</label>
      <input type="email" class="form-control" id="exampleDropdownFormEmail2"
placeholder="email@example.com">
    </div>
    <div class="mb-3">
      <label for="exampleDropdownFormPassword2"
class="form-label">Password</label>
      <input type="password" class="form-control"
id="exampleDropdownFormPassword2" placeholder="Password">
    </div>
    <div class="mb-3">
      <div class="form-check">
        <input type="checkbox" class="form-check-input" id="dropdownCheck2">
        <label class="form-check-label" for="dropdownCheck2">
          Remember me
        </label>
      </div>
    </div>
    <button type="submit" class="btn btn-primary">Sign in</button>
  </form>
</div>
```

Dropdown options

Use `data-bs-offset` or `data-bs-reference` to change the location of the dropdown.

html

```
<div class="d-flex">
  <div class="dropdown me-1">
    <button type="button" class="btn btn-secondary dropdown-toggle"
id="dropdownMenuOffset" data-bs-toggle="dropdown" aria-expanded="false" data-bs-
offset="10,20">
      Offset
    </button>
    <ul class="dropdown-menu" aria-labelledby="dropdownMenuOffset">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
    </ul>
  </div>
  <div class="btn-group">
    <button type="button" class="btn btn-secondary">Reference</button>
    <button type="button" class="btn btn-secondary dropdown-toggle dropdown-
toggle-split" id="dropdownMenuReference" data-bs-toggle="dropdown" aria-
expanded="false" data-bs-reference="parent">
      <span class="visually-hidden">Toggle Dropdown</span>
    </button>
    <ul class="dropdown-menu" aria-labelledby="dropdownMenuReference">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
      <li><hr class="dropdown-divider"></li>
      <li><a class="dropdown-item" href="#">Separated link</a></li>
    </ul>
  </div>
</div>
```

Auto close behavior

By default, the dropdown menu is closed when clicking inside or outside the dropdown menu. You can use the `autoClose` option to change this behavior of the dropdown.

html

```
<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle" type="button"
id="defaultDropdown" data-bs-toggle="dropdown" data-bs-auto-close="true" aria-
expanded="false">
    Default dropdown
  </button>
  <ul class="dropdown-menu" aria-labelledby="defaultDropdown">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenuClickableOutside" data-bs-toggle="dropdown" data-bs-auto-
close="inside" aria-expanded="false">
    Clickable outside
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuClickableOutside">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenuClickableInside" data-bs-toggle="dropdown" data-bs-auto-
close="outside" aria-expanded="false">
    Clickable inside
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuClickableInside">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenuClickable" data-bs-toggle="dropdown" data-bs-auto-close="false"
aria-expanded="false">
    Manual close
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuClickable">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>
```

CSS

Variables

Added in v5.2.0

As part of Bootstrap’s evolving CSS variables approach, dropdowns now use local CSS variables on `.dropdown-menu` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}dropdown-min-width: #{$dropdown-min-width};
  --#{$prefix}dropdown-padding-x: #{$dropdown-padding-x};
  --#{$prefix}dropdown-padding-y: #{$dropdown-padding-y};
  --#{$prefix}dropdown-spacer: #{$dropdown-spacer};
  @include rfs($dropdown-font-size, --#{$prefix}dropdown-font-size);
  --#{$prefix}dropdown-color: #{$dropdown-color}; // stylelint-disable-line
custom-property-empty-line-before
  --#{$prefix}dropdown-bg: #{$dropdown-bg};
  --#{$prefix}dropdown-border-color: #{$dropdown-border-color};
  --#{$prefix}dropdown-border-radius: #{$dropdown-border-radius};
  --#{$prefix}dropdown-border-width: #{$dropdown-border-width};
  --#{$prefix}dropdown-inner-border-radius: #{$dropdown-inner-border-radius};
  --#{$prefix}dropdown-divider-bg: #{$dropdown-divider-bg};
  --#{$prefix}dropdown-divider-margin-y: #{$dropdown-divider-margin-y};
  --#{$prefix}dropdown-box-shadow: #{$dropdown-box-shadow};
  --#{$prefix}dropdown-link-color: #{$dropdown-link-color};
  --#{$prefix}dropdown-link-hover-color: #{$dropdown-link-hover-color};
  --#{$prefix}dropdown-link-hover-bg: #{$dropdown-link-hover-bg};
  --#{$prefix}dropdown-link-active-color: #{$dropdown-link-active-color};
  --#{$prefix}dropdown-link-active-bg: #{$dropdown-link-active-bg};
  --#{$prefix}dropdown-link-disabled-color: #{$dropdown-link-disabled-color};
  --#{$prefix}dropdown-item-padding-x: #{$dropdown-item-padding-x};
  --#{$prefix}dropdown-item-padding-y: #{$dropdown-item-padding-y};
  --#{$prefix}dropdown-header-color: #{$dropdown-header-color};
  --#{$prefix}dropdown-header-padding-x: #{$dropdown-header-padding-x};
  --#{$prefix}dropdown-header-padding-y: #{$dropdown-header-padding-y};
```

Customization through CSS variables can be seen on the `.dropdown-menu-dark` class where we override specific values without adding duplicate CSS selectors.

```
  --#{$prefix}dropdown-color: #{$dropdown-dark-color};
  --#{$prefix}dropdown-bg: #{$dropdown-dark-bg};
  --#{$prefix}dropdown-border-color: #{$dropdown-dark-border-color};
  --#{$prefix}dropdown-box-shadow: #{$dropdown-dark-box-shadow};
  --#{$prefix}dropdown-link-color: #{$dropdown-dark-link-color};
  --#{$prefix}dropdown-link-hover-color: #{$dropdown-dark-link-hover-color};
  --#{$prefix}dropdown-divider-bg: #{$dropdown-dark-divider-bg};
  --#{$prefix}dropdown-link-hover-bg: #{$dropdown-dark-link-hover-bg};
  --#{$prefix}dropdown-link-active-color: #{$dropdown-dark-link-active-color};
  --#{$prefix}dropdown-link-active-bg: #{$dropdown-dark-link-active-bg};
  --#{$prefix}dropdown-link-disabled-color: #{$dropdown-dark-link-disabled-
color};
  --#{$prefix}dropdown-header-color: #{$dropdown-dark-header-color};
```

Sass variables

Variables for all dropdowns:

```
$dropdown-min-width:                10rem;
$dropdown-padding-x:                0;
$dropdown-padding-y:                .5rem;
$dropdown-spacer:                   .125rem;
$dropdown-font-size:                $font-size-base;
$dropdown-color:                    $body-color;
$dropdown-bg:                       $white;
$dropdown-border-color:             var(--#{$prefix}border-color-translucent);
$dropdown-border-radius:            $border-radius;
$dropdown-border-width:             $border-width;
$dropdown-inner-border-radius:      subtract($dropdown-border-radius, $dropdown-
border-width);
$dropdown-divider-bg:               $dropdown-border-color;
$dropdown-divider-margin-y:         $spacer * .5;
$dropdown-box-shadow:               $box-shadow;

$dropdown-link-color:               $gray-900;
$dropdown-link-hover-color:         shade-color($dropdown-link-color, 10%);
$dropdown-link-hover-bg:            $gray-200;

$dropdown-link-active-color:        $component-active-color;
$dropdown-link-active-bg:           $component-active-bg;

$dropdown-link-disabled-color:      $gray-500;

$dropdown-item-padding-y:           $spacer * .25;
$dropdown-item-padding-x:           $spacer;

$dropdown-header-color:             $gray-600;
$dropdown-header-padding-x:         $dropdown-item-padding-x;
```

```
$dropdown-header-padding-y:         $dropdown-padding-y;
// fusv-disable
$dropdown-header-padding:           $dropdown-header-padding-y $dropdown-header-padding-x; // Deprecated in v5.2.0
// fusv-enable
```

Variables for the dark dropdown:

```
$dropdown-dark-color:               $gray-300;
$dropdown-dark-bg:                  $gray-800;
$dropdown-dark-border-color:        $dropdown-border-color;
$dropdown-dark-divider-bg:          $dropdown-divider-bg;
$dropdown-dark-box-shadow:          null;
$dropdown-dark-link-color:          $dropdown-dark-color;
$dropdown-dark-link-hover-color:    $white;
$dropdown-dark-link-hover-bg:       rgba($white, .15);
$dropdown-dark-link-active-color:   $dropdown-link-active-color;
$dropdown-dark-link-active-bg:      $dropdown-link-active-bg;
$dropdown-dark-link-disabled-color: $gray-500;
$dropdown-dark-header-color:        $gray-500;
```

Variables for the CSS-based carets that indicate a dropdown's interactivity:

```
$caret-width:                 .3em;
$caret-vertical-align:        $caret-width * .85;
$caret-spacing:               $caret-width * .85;
```

Mixins

Mixins are used to generate the CSS-based carets and can be found in `scss/mixins/_caret.scss`.

```
@mixin caret-down {
  border-top: $caret-width solid;
  border-right: $caret-width solid transparent;
  border-bottom: 0;
  border-left: $caret-width solid transparent;
}

@mixin caret-up {
  border-top: 0;
  border-right: $caret-width solid transparent;
  border-bottom: $caret-width solid;
  border-left: $caret-width solid transparent;
}

@mixin caret-end {
  border-top: $caret-width solid transparent;
  border-right: 0;
  border-bottom: $caret-width solid transparent;
  border-left: $caret-width solid;
}

@mixin caret-start {
  border-top: $caret-width solid transparent;
  border-right: $caret-width solid;
  border-bottom: $caret-width solid transparent;
}

@mixin caret($direction: down) {
  @if $enable-caret {
    &::after {
      display: inline-block;
      margin-left: $caret-spacing;
      vertical-align: $caret-vertical-align;
      content: "";
      @if $direction == down {
        @include caret-down();
      } @else if $direction == up {
        @include caret-up();
      } @else if $direction == end {
        @include caret-end();
      }
    }

    @if $direction == start {
      &::after {
        display: none;
      }

      &::before {
        display: inline-block;
        margin-right: $caret-spacing;
        vertical-align: $caret-vertical-align;
        content: "";
        @include caret-start();
      }
    }

    &:empty::after {
      margin-left: 0;
    }
  }
}
```

Usage

Via data attributes or JavaScript, the dropdown plugin toggles hidden content (dropdown menus) by toggling the `.show` class on the parent `.dropdown-menu`. The `data-bs-toggle="dropdown"` attribute is relied on for closing dropdown menus at an application level, so it’s a good idea to always use it.

On touch-enabled devices, opening a dropdown adds empty `mouseover` handlers to the immediate children of the `<body>` element. This admittedly ugly hack is necessary to work around a quirk in iOS’ event delegation, which would otherwise prevent a tap anywhere outside of the dropdown from triggering the code that closes the dropdown. Once the dropdown is closed, these additional empty `mouseover` handlers are removed.

Via data attributes

Add `data-bs-toggle="dropdown"` to a link or button to toggle a dropdown.

```
<div class="dropdown">
  <button id="dLabel" type="button" data-bs-toggle="dropdown" aria-
expanded="false">
    Dropdown trigger
  </button>
  <ul class="dropdown-menu" aria-labelledby="dLabel">
    ...
  </ul>
```

```
</div>
```

Via JavaScript

Call the dropdowns via JavaScript:

```
const dropdownElementList = document.querySelectorAll('.dropdown-toggle')
const dropdownList = [...dropdownElementList].map(dropdownToggleEl => new
bootstrap.Dropdown(dropdownToggleEl))
```

`data-bs-toggle="dropdown"` still required

Regardless of whether you call your dropdown via JavaScript or instead use the data-api, `data-bs-toggle="dropdown"` is always required to be present on the dropdown's trigger element.

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-bs-`, as in `data-bs-animation=""`. Make sure to change the case type of the option name from camelCase to kebab-case when passing the options via data attributes. For example, use `data-bs-custom-class="beautifier"` instead of `data-bs-customClass="beautifier"`.

As of Bootstrap 5.2.0, all components support an **experimental** reserved data attribute `data-bs-config` that can house simple component configuration as a JSON string. When an element has `data-bs-config='{"delay":0, "title":123}'` and `data-bs-title="456"` attributes, the final `title` value will be `456` and the separate data attributes will override values given on `data-bs-config`. In addition, existing data attributes are able to house JSON values like `data-bs-delay='{"show":0,"hide":150}'`.

Name	Type	Default	Description
`boundary`	string, element	`'scrollParent'`	Overflow constraint boundary of the dropdown menu (applies only to Popper's preventOverflow modifier). By default it's `clippingParents` and can accept an HTMLElement reference (via JavaScript only). For more information refer to Popper's detectOverflow docs.
`reference`	string, element	`'toggle'`	Reference element of the dropdown menu. Accepts the values of `'toggle'`, `'parent'`, an HTMLElement reference or an object providing `getBoundingClientRect`. For more information refer to Popper's constructor docs and virtual element docs.
`display`	string	`'dynamic'`	By default, we use Popper for dynamic positioning. Disable this with `static`.
`offset`	number, string, function	`[0, 2]`	Offset of the dropdown relative to its target. You can pass a string in data attributes with comma separated values like: `data-bs-offset="10,20"`. When a function is used to determine the offset, it is called with an object containing the popper placement, the reference,

Name	Type	Default	Description
			and popper rects as its first argument. The triggering element DOM node is passed as the second argument. The function must return an array with two numbers: skidding, distance. For more information refer to Popper's offset docs.
`autoClose`	boolean, string	`true`	Configure the auto close behavior of the dropdown: • `true` - the dropdown will be closed by clicking outside or inside the dropdown menu. • `false` - the dropdown will be closed by clicking the toggle button and manually calling `hide` or `toggle` method. (Also will not be closed by pressing `esc` key) • `'inside'` - the dropdown will be closed (only) by clicking inside the dropdown menu. • `'outside'` - the dropdown will be closed (only) by clicking outside the dropdown menu. Note: the dropdown can always be closed with the `ESC` key
`popperConfig`	null, object, function	`null`	To change Bootstrap's default Popper config, see Popper's configuration. When a function is used to create the Popper configuration, it's called with an object that contains the Bootstrap's default Popper configuration. It helps you use and merge the default with your own configuration. The function must return a configuration object for Popper.

Using function with `popperConfig`

```
const dropdown = new bootstrap.Dropdown(element, {
  popperConfig(defaultBsPopperConfig) {
    // const newPopperConfig = {...}
    // use defaultBsPopperConfig if needed...
    // return newPopperConfig
  }
})
```

Methods

Method	Description
`toggle`	Toggles the dropdown menu of a given navbar or tabbed navigation.
`show`	Shows the dropdown menu of a given navbar or tabbed navigation.
`hide`	Hides the dropdown menu of a given navbar or tabbed navigation.
`update`	Updates the position of an element's dropdown.
`dispose`	Destroys an element's dropdown. (Removes stored data on the DOM

Method	Description
	element)
`getInstance`	Static method which allows you to get the dropdown instance associated to a DOM element, you can use it like this: `bootstrap.Dropdown.getInstance(element)`
`getOrCreateInstance`	Static method which returns a dropdown instance associated to a DOM element or create a new one in case it wasn’t initialized. You can use it like this: `bootstrap.Dropdown.getOrCreateInstance(element)`.

Events

All dropdown events are fired at the toggling element and then bubbled up. So you can also add event listeners on the `.dropdown-menu`’s parent element. `hide.bs.dropdown` and `hidden.bs.dropdown` events have a `clickEvent` property (only when the original Event type is `click`) that contains an Event Object for the click event.

Event type	Description
`show.bs.dropdown`	Fires immediately when the `show` instance method is called.
`shown.bs.dropdown`	Fired when the dropdown has been made visible to the user and CSS transitions have completed.
`hide.bs.dropdown`	Fires immediately when the `hide` instance method has been called.
`hidden.bs.dropdown`	Fired when the dropdown has finished being hidden from the user and CSS transitions have completed.

```
const myDropdown = document.getElementById('myDropdown')
myDropdown.addEventListener('show.bs.dropdown', event => {
  // do something...
})
```

List group

List groups are a flexible and powerful component for displaying a series of content. Modify and extend them to support just about any content within.

On this page

Basic example

The most basic list group is an unordered list with list items and the proper classes. Build upon it with the options that follow, or with your own CSS as needed.

- An item
- A second item
- A third item
- A fourth item
- And a fifth one

html

```
<ul class="list-group">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
</ul>
```

Active items

Add `.active` to a `.list-group-item` to indicate the current active selection.

- An active item
- A second item
- A third item
- A fourth item
- And a fifth one

html

```
<ul class="list-group">
  <li class="list-group-item active" aria-current="true">An active item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
</ul>
```

Disabled items

Add `.disabled` to a `.list-group-item` to make it *appear* disabled. Note that some elements with `.disabled` will also require custom JavaScript to fully disable their click events (e.g., links).

- A disabled item
- A second item
- A third item
- A fourth item
- And a fifth one

html

```
<ul class="list-group">
  <li class="list-group-item disabled" aria-disabled="true">A disabled item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
</ul>
```

Links and buttons

Use `<a>`s or `<button>`s to create *actionable* list group items with hover, disabled, and active states by adding `.list-group-item-action`. We separate these pseudo-classes to ensure list groups made of non-interactive elements (like `<li>`s or `<div>`s) don't provide a click or tap affordance.

Be sure to **not use the standard `.btn` classes here**.

The current link item A second link item A third link item A fourth link item A disabled link item

html

```
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action active" aria-
current="true">
    The current link item
  </a>
  <a href="#" class="list-group-item list-group-item-action">A second link
item</a>
  <a href="#" class="list-group-item list-group-item-action">A third link
item</a>
  <a href="#" class="list-group-item list-group-item-action">A fourth link
item</a>
  <a class="list-group-item list-group-item-action disabled">A disabled link
item</a>
</div>
```

With `<button>`s, you can also make use of the `disabled` attribute instead of the `.disabled` class. Sadly, `<a>`s don't support the disabled attribute.

html

```
<div class="list-group">
  <button type="button" class="list-group-item list-group-item-action active"
aria-current="true">
    The current button
  </button>
  <button type="button" class="list-group-item list-group-item-action">A second
button item</button>
  <button type="button" class="list-group-item list-group-item-action">A third
button item</button>
  <button type="button" class="list-group-item list-group-item-action">A fourth
button item</button>
  <button type="button" class="list-group-item list-group-item-action"
disabled>A disabled button item</button>
</div>
```

Flush

Add `.list-group-flush` to remove some borders and rounded corners to render list group items edge-to-edge in a parent container (e.g., cards).

- An item
- A second item
- A third item
- A fourth item
- And a fifth one

html

```
<ul class="list-group list-group-flush">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
</ul>
```

Numbered

Add the `.list-group-numbered` modifier class (and optionally use an `<ol>` element) to opt into numbered list group items. Numbers are generated via CSS (as opposed to a `<ol>`s default browser styling) for better placement inside list group items and to allow for better customization.

Numbers are generated by `counter-reset` on the `<ol>`, and then styled and placed with a `::before` pseudo-element on the `<li>` with `counter-increment` and `content`.

1. A list item
2. A list item
3. A list item

html

```
<ol class="list-group list-group-numbered">
  <li class="list-group-item">A list item</li>
  <li class="list-group-item">A list item</li>
  <li class="list-group-item">A list item</li>
</ol>
```

These work great with custom content as well.

1. Subheading
   Content for list item
   14
2. Subheading
   Content for list item
   14
3. Subheading
   Content for list item
   14

html

```
<ol class="list-group list-group-numbered">
  <li class="list-group-item d-flex justify-content-between align-items-start">
    <div class="ms-2 me-auto">
      <div class="fw-bold">Subheading</div>
      Content for list item
    </div>
    <span class="badge bg-primary rounded-pill">14</span>
  </li>
  <li class="list-group-item d-flex justify-content-between align-items-start">
    <div class="ms-2 me-auto">
      <div class="fw-bold">Subheading</div>
      Content for list item
    </div>
    <span class="badge bg-primary rounded-pill">14</span>
  </li>
  <li class="list-group-item d-flex justify-content-between align-items-start">
    <div class="ms-2 me-auto">
      <div class="fw-bold">Subheading</div>
      Content for list item
    </div>
    <span class="badge bg-primary rounded-pill">14</span>
  </li>
</ol>
```

Horizontal

Add `.list-group-horizontal` to change the layout of list group items from vertical to horizontal across all breakpoints. Alternatively, choose a responsive variant `.list-group-horizontal-{sm|md|lg|xl|xxl}` to make a list group horizontal starting at that breakpoint's `min-width`. Currently **horizontal list groups cannot be combined with flush list groups.**

ProTip: Want equal-width list group items when horizontal? Add `.flex-fill` to each list group item.

- An item
- A second item
- A third item

- An item
- A second item
- A third item

- An item
- A second item
- A third item

- An item
- A second item
- A third item

- An item
- A second item
- A third item

- An item
- A second item
- A third item

html

```
<ul class="list-group list-group-horizontal">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
```

```
<ul class="list-group list-group-horizontal-sm">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
<ul class="list-group list-group-horizontal-md">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
<ul class="list-group list-group-horizontal-lg">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
<ul class="list-group list-group-horizontal-xl">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
<ul class="list-group list-group-horizontal-xxl">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
```

Contextual classes

Use contextual classes to style list items with a stateful background and color.

- A simple default list group item
- A simple primary list group item
- A simple secondary list group item
- A simple success list group item
- A simple danger list group item
- A simple warning list group item
- A simple info list group item
- A simple light list group item
- A simple dark list group item

html

```
<ul class="list-group">
  <li class="list-group-item">A simple default list group item</li>

  <li class="list-group-item list-group-item-primary">A simple primary list
group item</li>
  <li class="list-group-item list-group-item-secondary">A simple secondary list
group item</li>
  <li class="list-group-item list-group-item-success">A simple success list
group item</li>
  <li class="list-group-item list-group-item-danger">A simple danger list group
item</li>
  <li class="list-group-item list-group-item-warning">A simple warning list
group item</li>
  <li class="list-group-item list-group-item-info">A simple info list group
item</li>
  <li class="list-group-item list-group-item-light">A simple light list group
item</li>
```

```
  <li class="list-group-item list-group-item-dark">A simple dark list group
item</li>
</ul>
```

Contextual classes also work with `.list-group-item-action`. Note the addition of the hover styles here not present in the previous example. Also supported is the `.active` state; apply it to indicate an active selection on a contextual list group item.

A simple default list group item A simple primary list group item A simple secondary list group item A simple success list group item A simple danger list group item A simple warning list group item A simple info list group item A simple light list group item A simple dark list group item

html

```
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action">A simple default
list group item</a>

  <a href="#" class="list-group-item list-group-item-action list-group-item-
primary">A simple primary list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-
secondary">A simple secondary list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-
success">A simple success list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-
danger">A simple danger list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-
warning">A simple warning list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-
info">A simple info list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-
light">A simple light list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-
dark">A simple dark list group item</a>
</div>
```

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.

With badges

Add badges to any list group item to show unread counts, activity, and more with the help of some utilities.

- A list item 14
- A second list item 2
- A third list item 1

html

```
<ul class="list-group">
  <li class="list-group-item d-flex justify-content-between align-items-center">
    A list item
    <span class="badge bg-primary rounded-pill">14</span>
  </li>
  <li class="list-group-item d-flex justify-content-between align-items-center">
    A second list item
    <span class="badge bg-primary rounded-pill">2</span>
  </li>
  <li class="list-group-item d-flex justify-content-between align-items-center">
    A third list item
    <span class="badge bg-primary rounded-pill">1</span>
  </li>
</ul>
```

Custom content

Add nearly any HTML within, even for linked list groups like the one below, with the help of [flexbox utilities](#).

List group item heading

3 days ago

Some placeholder content in a paragraph.

And some small print.

List group item heading

3 days ago

Some placeholder content in a paragraph.

And some muted small print.

List group item heading

3 days ago

Some placeholder content in a paragraph.

And some muted small print.

html

```
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action active" aria-
current="true">
    <div class="d-flex w-100 justify-content-between">
      <h5 class="mb-1">List group item heading</h5>
      <small>3 days ago</small>
    </div>
    <p class="mb-1">Some placeholder content in a paragraph.</p>
    <small>And some small print.</small>
  </a>
  <a href="#" class="list-group-item list-group-item-action">
    <div class="d-flex w-100 justify-content-between">
      <h5 class="mb-1">List group item heading</h5>
      <small class="text-muted">3 days ago</small>
    </div>
    <p class="mb-1">Some placeholder content in a paragraph.</p>
    <small class="text-muted">And some muted small print.</small>
  </a>
  <a href="#" class="list-group-item list-group-item-action">
    <div class="d-flex w-100 justify-content-between">
      <h5 class="mb-1">List group item heading</h5>
      <small class="text-muted">3 days ago</small>
    </div>
    <p class="mb-1">Some placeholder content in a paragraph.</p>
    <small class="text-muted">And some muted small print.</small>
  </a>
</div>
```

Checkboxes and radios

Place Bootstrap’s checkboxes and radios within list group items and customize as needed. You can use them without `<label>`s, but please remember to include an `aria-label` attribute and value for accessibility.

- First checkbox
- Second checkbox
- Third checkbox
- Fourth checkbox
- Fifth checkbox

html

```
<ul class="list-group">
  <li class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="" aria-
label="...">
    First checkbox
  </li>
  <li class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="" aria-
label="...">
    Second checkbox
  </li>
  <li class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="" aria-
label="...">
    Third checkbox
  </li>
  <li class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="" aria-
label="...">
    Fourth checkbox
  </li>
  <li class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="" aria-
label="...">
    Fifth checkbox
  </li>
</ul>
```

And if you want `<label>`s as the `.list-group-item` for large hit areas, you can do that, too.

First checkbox Second checkbox Third checkbox Fourth checkbox Fifth checkbox

html

```
<div class="list-group">
  <label class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="">
    First checkbox
  </label>
  <label class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="">
    Second checkbox
  </label>
  <label class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="">
    Third checkbox
  </label>
  <label class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="">
    Fourth checkbox
  </label>
  <label class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="">
    Fifth checkbox
  </label>
</div>
```

CSS

Variables

Added in v5.2.0

As part of Bootstrap's evolving CSS variables approach, list groups now use local CSS variables on `.list-group` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}list-group-color: #{$list-group-color};
  --#{$prefix}list-group-bg: #{$list-group-bg};
  --#{$prefix}list-group-border-color: #{$list-group-border-color};
  --#{$prefix}list-group-border-width: #{$list-group-border-width};
  --#{$prefix}list-group-border-radius: #{$list-group-border-radius};
  --#{$prefix}list-group-item-padding-x: #{$list-group-item-padding-x};
  --#{$prefix}list-group-item-padding-y: #{$list-group-item-padding-y};
  --#{$prefix}list-group-action-color: #{$list-group-action-color};
  --#{$prefix}list-group-action-hover-color: #{$list-group-action-hover-color};
  --#{$prefix}list-group-action-hover-bg: #{$list-group-hover-bg};
  --#{$prefix}list-group-action-active-color: #{$list-group-action-active-
color};
  --#{$prefix}list-group-action-active-bg: #{$list-group-action-active-bg};
  --#{$prefix}list-group-disabled-color: #{$list-group-disabled-color};
  --#{$prefix}list-group-disabled-bg: #{$list-group-disabled-bg};
  --#{$prefix}list-group-active-color: #{$list-group-active-color};
  --#{$prefix}list-group-active-bg: #{$list-group-active-bg};
  --#{$prefix}list-group-active-border-color: #{$list-group-active-border-
color};
```

Sass variables

```
$list-group-color:                  $gray-900;
$list-group-bg:                     $white;
$list-group-border-color:           rgba($black, .125);
$list-group-border-width:           $border-width;
$list-group-border-radius:          $border-radius;

$list-group-item-padding-y:         $spacer * .5;
$list-group-item-padding-x:         $spacer;
$list-group-item-bg-scale:          -80%;
```

```
$list-group-item-color-scale:       40%;

$list-group-hover-bg:               $gray-100;
$list-group-active-color:           $component-active-color;
$list-group-active-bg:              $component-active-bg;
$list-group-active-border-color:    $list-group-active-bg;

$list-group-disabled-color:         $gray-600;
$list-group-disabled-bg:            $list-group-bg;

$list-group-action-color:           $gray-700;
$list-group-action-hover-color:     $list-group-action-color;

$list-group-action-active-color:    $body-color;
$list-group-action-active-bg:       $gray-200;
```

Mixins

Used in combination with `$theme-colors` to generate the contextual variant classes for `.list-group-item`s.

```
@mixin list-group-item-variant($state, $background, $color) {
  .list-group-item-#{$state} {
    color: $color;
    background-color: $background;

    &.list-group-item-action {
      &:hover,
      &:focus {
        color: $color;
        background-color: shade-color($background, 10%);
      }

      &.active {
        color: $white;
        background-color: $color;
        border-color: $color;
      }
    }
  }
}
```

Loop

Loop that generates the modifier classes with the `list-group-item-variant()` mixin.

```
// List group contextual variants
//
// Add modifier classes to change text and background color on individual items.
// Organizationally, this must come after the `:hover` states.

@each $state, $value in $theme-colors {
  $list-group-variant-bg: shift-color($value, $list-group-item-bg-scale);
  $list-group-variant-color: shift-color($value, $list-group-item-color-scale);
  @if (contrast-ratio($list-group-variant-bg, $list-group-variant-color) < $min-
contrast-ratio) {
    $list-group-variant-color: mix($value, color-contrast($list-group-variant-
bg), abs($list-group-item-color-scale));
  }
  @include list-group-item-variant($state, $list-group-variant-bg, $list-group-
variant-color);
}
```

JavaScript behavior

Use the tab JavaScript plugin—include it individually or through the compiled `bootstrap.js` file—to extend our list group to create tabbable panes of local content.

Home Profile Messages Settings

Some placeholder content in a paragraph relating to "Home". And some more content, used here just to pad out and fill this tab panel. In production, you would obviously have more real content here. And not just text. It could be anything, really. Text, images, forms.

```
<div class="row">
  <div class="col-4">
    <div class="list-group" id="list-tab" role="tablist">
      <a class="list-group-item list-group-item-action active" id="list-home-
list" data-bs-toggle="list" href="#list-home" role="tab" aria-controls="list-
home">Home</a>
      <a class="list-group-item list-group-item-action" id="list-profile-list"
data-bs-toggle="list" href="#list-profile" role="tab" aria-controls="list-
profile">Profile</a>
      <a class="list-group-item list-group-item-action" id="list-messages-list"
data-bs-toggle="list" href="#list-messages" role="tab" aria-controls="list-
messages">Messages</a>
      <a class="list-group-item list-group-item-action" id="list-settings-list"
data-bs-toggle="list" href="#list-settings" role="tab" aria-controls="list-
settings">Settings</a>
    </div>
  </div>
  <div class="col-8">
    <div class="tab-content" id="nav-tabContent">
      <div class="tab-pane fade show active" id="list-home" role="tabpanel"
aria-labelledby="list-home-list">...</div>
      <div class="tab-pane fade" id="list-profile" role="tabpanel" aria-
labelledby="list-profile-list">...</div>
      <div class="tab-pane fade" id="list-messages" role="tabpanel" aria-
labelledby="list-messages-list">...</div>
      <div class="tab-pane fade" id="list-settings" role="tabpanel" aria-
labelledby="list-settings-list">...</div>
    </div>
  </div>
</div>
```

Using data attributes

You can activate a list group navigation without writing any JavaScript by simply specifying `data-bs-toggle="list"` or on an element. Use these data attributes on `.list-group-item`.

```
<div role="tabpanel">
  <!-- List group -->
  <div class="list-group" id="myList" role="tablist">
    <a class="list-group-item list-group-item-action active" data-bs-
toggle="list" href="#home" role="tab">Home</a>
```

```
    <a class="list-group-item list-group-item-action" data-bs-toggle="list"
href="#profile" role="tab">Profile</a>
    <a class="list-group-item list-group-item-action" data-bs-toggle="list"
href="#messages" role="tab">Messages</a>
    <a class="list-group-item list-group-item-action" data-bs-toggle="list"
href="#settings" role="tab">Settings</a>
  </div>

  <!-- Tab panes -->
  <div class="tab-content">
    <div class="tab-pane active" id="home" role="tabpanel">...</div>
    <div class="tab-pane" id="profile" role="tabpanel">...</div>
    <div class="tab-pane" id="messages" role="tabpanel">...</div>
    <div class="tab-pane" id="settings" role="tabpanel">...</div>
  </div>
</div>
```

Via JavaScript

Enable tabbable list item via JavaScript (each list item needs to be activated individually):

```
const triggerTabList = document.querySelectorAll('#myTab a')
triggerTabList.forEach(triggerEl => {
  const tabTrigger = new bootstrap.Tab(triggerEl)

  triggerEl.addEventListener('click', event => {
    event.preventDefault()
    tabTrigger.show()
  })
})
```

You can activate individual list item in several ways:

```
const triggerEl = document.querySelector('#myTab a[href="#profile"]')
bootstrap.Tab.getInstance(triggerEl).show() // Select tab by name

const triggerFirstTabEl = document.querySelector('#myTab li:first-child a')
bootstrap.Tab.getInstance(triggerFirstTabEl).show() // Select first tab
```

Fade effect

To make tabs panel fade in, add `.fade` to each `.tab-pane`. The first tab pane must also have `.show` to make the initial content visible.

```
<div class="tab-content">
  <div class="tab-pane fade show active" id="home" role="tabpanel">...</div>
  <div class="tab-pane fade" id="profile" role="tabpanel">...</div>
  <div class="tab-pane fade" id="messages" role="tabpanel">...</div>
  <div class="tab-pane fade" id="settings" role="tabpanel">...</div>
</div>
```

Methods

constructor

Activates a list item element and content container. Tab should have either a `data-bs-target` or an `href` targeting a container node in the DOM.

```
<div class="list-group" id="myList" role="tablist">
```

```
  <a class="list-group-item list-group-item-action active" data-bs-toggle="list"
href="#home" role="tab">Home</a>
  <a class="list-group-item list-group-item-action" data-bs-toggle="list"
href="#profile" role="tab">Profile</a>
  <a class="list-group-item list-group-item-action" data-bs-toggle="list"
href="#messages" role="tab">Messages</a>
  <a class="list-group-item list-group-item-action" data-bs-toggle="list"
href="#settings" role="tab">Settings</a>
</div>

<div class="tab-content">
  <div class="tab-pane active" id="home" role="tabpanel">...</div>
  <div class="tab-pane" id="profile" role="tabpanel">...</div>
  <div class="tab-pane" id="messages" role="tabpanel">...</div>
  <div class="tab-pane" id="settings" role="tabpanel">...</div>
</div>

<script>
  const firstTabEl = document.querySelector('#myTab a:last-child')
  const firstTab = new bootstrap.Tab(firstTabEl)

  firstTab.show()
</script>
```

show

Selects the given list item and shows its associated pane. Any other list item that was previously selected becomes unselected and its associated pane is hidden. **Returns to the caller before the tab pane has actually been shown** (for example, before the `shown.bs.tab` event occurs).

```
const tab = new bootstrap.Tab('#someListItem')

tab.show()
```

dispose

Destroys an element’s tab.

getInstance

Static method which allows you to get the tab instance associated with a DOM element

```
const tab = bootstrap.Tab.getInstance('#trigger') // Returns a Bootstrap tab
instance
```

getOrCreateInstance

Static method which allows you to get the tab instance associated with a DOM element, or create a new one in case it wasn’t initialized

```
const tab = bootstrap.Tab.getOrCreateInstance('#trigger') // Returns a Bootstrap
tab instance
```

Events

When showing a new tab, the events fire in the following order:

1. `hide.bs.tab` (on the current active tab)
2. `show.bs.tab` (on the to-be-shown tab)

3. `hidden.bs.tab` (on the previous active tab, the same one as for the `hide.bs.tab` event)
4. `shown.bs.tab` (on the newly-active just-shown tab, the same one as for the `show.bs.tab` event)

If no tab was already active, the `hide.bs.tab` and `hidden.bs.tab` events will not be fired.

Event type	Description
`show.bs.tab`	This event fires on tab show, but before the new tab has been shown. Use `event.target` and `event.relatedTarget` to target the active tab and the previous active tab (if available) respectively.
`shown.bs.tab`	This event fires on tab show after a tab has been shown. Use `event.target` and `event.relatedTarget` to target the active tab and the previous active tab (if available) respectively.
`hide.bs.tab`	This event fires when a new tab is to be shown (and thus the previous active tab is to be hidden). Use `event.target` and `event.relatedTarget` to target the current active tab and the new soon-to-be-active tab, respectively.
`hidden.bs.tab`	This event fires after a new tab is shown (and thus the previous active tab is hidden). Use `event.target` and `event.relatedTarget` to target the previous active tab and the new active tab, respectively.

```
const tabElms = document.querySelectorAll('a[data-bs-toggle="list"]')
tabElms.forEach(tabElm => {
  tabElm.addEventListener('shown.bs.tab', event => {
    event.target // newly activated tab
    event.relatedTarget // previous active tab
  })
})
```

Modal

Use Bootstrap’s JavaScript modal plugin to add dialogs to your site for lightboxes, user notifications, or completely custom content.

On this page

How it works

Before getting started with Bootstrap’s modal component, be sure to read the following as our menu options have recently changed.

- Modals are built with HTML, CSS, and JavaScript. They’re positioned over everything else in the document and remove scroll from the `<body>` so that modal content scrolls instead.

- Clicking on the modal “backdrop” will automatically close the modal.
- Bootstrap only supports one modal window at a time. Nested modals aren’t supported as we believe them to be poor user experiences.
- Modals use `position: fixed`, which can sometimes be a bit particular about its rendering. Whenever possible, place your modal HTML in a top-level position to avoid potential interference from other elements. You’ll likely run into issues when nesting a `.modal` within another fixed element.
- Once again, due to `position: fixed`, there are some caveats with using modals on mobile devices. See our browser support docs for details.
- Due to how HTML5 defines its semantics, the `autofocus` HTML attribute has no effect in Bootstrap modals. To achieve the same effect, use some custom JavaScript:

```
const myModal = document.getElementById('myModal')
const myInput = document.getElementById('myInput')

myModal.addEventListener('shown.bs.modal', () => {
  myInput.focus()
})
```

The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the reduced motion section of our accessibility documentation.

Keep reading for demos and usage guidelines.

Examples

Modal components

Below is a *static* modal example (meaning its `position` and `display` have been overridden). Included are the modal header, modal body (required for `padding`), and modal footer (optional). We ask that you include modal headers with dismiss actions whenever possible, or provide another explicit dismiss action.

Modal title

Modal body text goes here.

```
<div class="modal" tabindex="-1">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <div class="modal-body">
        <p>Modal body text goes here.</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

Live demo

Toggle a working modal demo by clicking the button below. It will slide down and fade in from the top of the page.

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-
target="#exampleModal">
  Launch demo modal
</button>

<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

Static backdrop

When backdrop is set to static, the modal will not close when clicking outside of it. Click the button below to try it.

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-
target="#staticBackdrop">
  Launch static backdrop modal
</button>

<!-- Modal -->
<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-
keyboard="false" tabindex="-1" aria-labelledby="staticBackdropLabel" aria-
hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="staticBackdropLabel">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Understood</button>
      </div>
    </div>
  </div>
</div>
```

Scrolling long content

When modals become too long for the user’s viewport or device, they scroll independent of the page itself. Try the demo below to see what we mean.

You can also create a scrollable modal that allows scroll the modal body by adding `.modal-dialog-scrollable` to `.modal-dialog`.

```
<!-- Scrollable modal -->
<div class="modal-dialog modal-dialog-scrollable">
  ...
</div>
```

Vertically centered

Add `.modal-dialog-centered` to `.modal-dialog` to vertically center the modal.

```
<!-- Vertically centered modal -->
<div class="modal-dialog modal-dialog-centered">
  ...
</div>

<!-- Vertically centered scrollable modal -->
<div class="modal-dialog modal-dialog-centered modal-dialog-scrollable">
  ...
</div>
```

Tooltips and popovers

Tooltips and popovers can be placed within modals as needed. When modals are closed, any tooltips and popovers within are also automatically dismissed.

```
<div class="modal-body">
  <h5>Popover in a modal</h5>
  <p>This <a href="#" role="button" class="btn btn-secondary" data-bs-
toggle="popover" title="Popover title" data-bs-content="Popover body content is
set in this attribute.">button</a> triggers a popover on click.</p>
  <hr>
  <h5>Tooltips in a modal</h5>
  <p><a href="#" data-bs-toggle="tooltip" title="Tooltip">This link</a> and <a
href="#" data-bs-toggle="tooltip" title="Tooltip">that link</a> have tooltips on
hover.</p>
</div>
```

Using the grid

Utilize the Bootstrap grid system within a modal by nesting `.container-fluid` within the `.modal-body`. Then, use the normal grid system classes as you would anywhere else.

```
<div class="modal-body">
  <div class="container-fluid">
    <div class="row">
      <div class="col-md-4">.col-md-4</div>
      <div class="col-md-4 ms-auto">.col-md-4 .ms-auto</div>
    </div>
    <div class="row">
      <div class="col-md-3 ms-auto">.col-md-3 .ms-auto</div>
      <div class="col-md-2 ms-auto">.col-md-2 .ms-auto</div>
    </div>
    <div class="row">
      <div class="col-md-6 ms-auto">.col-md-6 .ms-auto</div>
    </div>
    <div class="row">
      <div class="col-sm-9">
        Level 1: .col-sm-9
        <div class="row">
          <div class="col-8 col-sm-6">
            Level 2: .col-8 .col-sm-6
          </div>
          <div class="col-4 col-sm-6">
            Level 2: .col-4 .col-sm-6
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Varying modal content

Have a bunch of buttons that all trigger the same modal with slightly different contents? Use `event.relatedTarget` and HTML `data-bs-*` attributes to vary the contents of the modal depending on which button was clicked.

Below is a live demo followed by example HTML and JavaScript. For more information, read the modal events docs for details on `relatedTarget`.

html

```
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-
target="#exampleModal" data-bs-whatever="@mdo">Open modal for @mdo</button>
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-
target="#exampleModal" data-bs-whatever="@fat">Open modal for @fat</button>
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-
target="#exampleModal" data-bs-whatever="@getbootstrap">Open modal for
@getbootstrap</button>

<div class="modal fade" id="exampleModal" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">New message</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <div class="modal-body">
        <form>
          <div class="mb-3">
            <label for="recipient-name"
class="col-form-label">Recipient:</label>
            <input type="text" class="form-control" id="recipient-name">
          </div>
          <div class="mb-3">
```

```
            <label for="message-text" class="col-form-label">Message:</label>
            <textarea class="form-control" id="message-text"></textarea>
          </div>
        </form>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Send message</button>
      </div>
    </div>
  </div>
</div>
```

```
const exampleModal = document.getElementById('exampleModal')
exampleModal.addEventListener('show.bs.modal', event => {
  // Button that triggered the modal
  const button = event.relatedTarget
  // Extract info from data-bs-* attributes
  const recipient = button.getAttribute('data-bs-whatever')
  // If necessary, you could initiate an AJAX request here
  // and then do the updating in a callback.
  //
  // Update the modal's content.
  const modalTitle = exampleModal.querySelector('.modal-title')
  const modalBodyInput = exampleModal.querySelector('.modal-body input')

  modalTitle.textContent = `New message to ${recipient}`
  modalBodyInput.value = recipient
})
```

Toggle between modals

Toggle between multiple modals with some clever placement of the `data-bs-target` and `data-bs-toggle` attributes. For example, you could toggle a password reset modal from within an already open sign in modal. **Please note multiple modals cannot be open at the same time**—this method simply toggles between two separate modals.

html

```
<div class="modal fade" id="exampleModalToggle" aria-hidden="true" aria-
labelledby="exampleModalToggleLabel" tabindex="-1">
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalToggleLabel">Modal 1</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <div class="modal-body">
        Show a second modal and hide this one with the button below.
      </div>
      <div class="modal-footer">
        <button class="btn btn-primary" data-bs-target="#exampleModalToggle2"
data-bs-toggle="modal">Open second modal</button>
      </div>
    </div>
  </div>
</div>
<div class="modal fade" id="exampleModalToggle2" aria-hidden="true" aria-
labelledby="exampleModalToggleLabel2" tabindex="-1">
```

```
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalToggleLabel2">Modal 2</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <div class="modal-body">
        Hide this modal and show the first with the button below.
      </div>
      <div class="modal-footer">
        <button class="btn btn-primary" data-bs-target="#exampleModalToggle"
data-bs-toggle="modal">Back to first</button>
      </div>
    </div>
  </div>
</div>
<a class="btn btn-primary" data-bs-toggle="modal" href="#exampleModalToggle"
role="button">Open first modal</a>
```

Change animation

The `$modal-fade-transform` variable determines the transform state of `.modal-dialog` before the modal fade-in animation, the `$modal-show-transform` variable determines the transform of `.modal-dialog` at the end of the modal fade-in animation.

If you want for example a zoom-in animation, you can set `$modal-fade-transform: scale(.8)`.

Remove animation

For modals that simply appear rather than fade in to view, remove the `.fade` class from your modal markup.

```
<div class="modal" tabindex="-1" aria-labelledby="..." aria-hidden="true">
  ...
</div>
```

Dynamic heights

If the height of a modal changes while it is open, you should call `myModal.handleUpdate()` to readjust the modal's position in case a scrollbar appears.

Accessibility

Be sure to add `aria-labelledby="..."`, referencing the modal title, to `.modal`. Additionally, you may give a description of your modal dialog with `aria-describedby` on `.modal`. Note that you don't need to add `role="dialog"` since we already add it via JavaScript.

Embedding YouTube videos

Embedding YouTube videos in modals requires additional JavaScript not in Bootstrap to automatically stop playback and more. See this helpful Stack Overflow post for more information.

Optional sizes

Modals have three optional sizes, available via modifier classes to be placed on a `.modal-dialog`. These sizes kick in at certain breakpoints to avoid horizontal scrollbars on narrower viewports.

Size	Class	Modal max-width
Small	`.modal-sm`	`300px`
Default	None	`500px`
Large	`.modal-lg`	`800px`
Extra large	`.modal-xl`	`1140px`

Our default modal without modifier class constitutes the “medium” size modal.

```
<div class="modal-dialog modal-xl">...</div>
<div class="modal-dialog modal-lg">...</div>
<div class="modal-dialog modal-sm">...</div>
```

Fullscreen Modal

Another override is the option to pop up a modal that covers the user viewport, available via modifier classes that are placed on a `.modal-dialog`.

Class	Availability
`.modal-fullscreen`	Always
`.modal-fullscreen-sm-down`	`576px`
`.modal-fullscreen-md-down`	`768px`
`.modal-fullscreen-lg-down`	`992px`
`.modal-fullscreen-xl-down`	`1200px`
`.modal-fullscreen-xxl-down`	`1400px`

```
<!-- Full screen modal -->
<div class="modal-dialog modal-fullscreen-sm-down">
  ...
</div>
```

CSS

Variables

Added in v5.2.0

As part of Bootstrap’s evolving CSS variables approach, modals now use local CSS variables on `.modal` and `.modal-backdrop` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}modal-zindex: #{$zindex-modal};
  --#{$prefix}modal-width: #{$modal-md};
  --#{$prefix}modal-padding: #{$modal-inner-padding};
  --#{$prefix}modal-margin: #{$modal-dialog-margin};
  --#{$prefix}modal-color: #{$modal-content-color};
  --#{$prefix}modal-bg: #{$modal-content-bg};
  --#{$prefix}modal-border-color: #{$modal-content-border-color};
  --#{$prefix}modal-border-width: #{$modal-content-border-width};
  --#{$prefix}modal-border-radius: #{$modal-content-border-radius};
  --#{$prefix}modal-box-shadow: #{$modal-content-box-shadow-xs};
  --#{$prefix}modal-inner-border-radius: #{$modal-content-inner-border-radius};
  --#{$prefix}modal-header-padding-x: #{$modal-header-padding-x};
  --#{$prefix}modal-header-padding-y: #{$modal-header-padding-y};
  --#{$prefix}modal-header-padding: #{$modal-header-padding}; // Todo in v6:
Split this padding into x and y
  --#{$prefix}modal-header-border-color: #{$modal-header-border-color};
  --#{$prefix}modal-header-border-width: #{$modal-header-border-width};
  --#{$prefix}modal-title-line-height: #{$modal-title-line-height};
  --#{$prefix}modal-footer-gap: #{$modal-footer-margin-between};
  --#{$prefix}modal-footer-bg: #{$modal-footer-bg};
  --#{$prefix}modal-footer-border-color: #{$modal-footer-border-color};
  --#{$prefix}modal-footer-border-width: #{$modal-footer-border-width};


  --#{$prefix}backdrop-zindex: #{$zindex-modal-backdrop};
  --#{$prefix}backdrop-bg: #{$modal-backdrop-bg};
  --#{$prefix}backdrop-opacity: #{$modal-backdrop-opacity};
```

Sass variables

```
$modal-inner-padding:               $spacer;

$modal-footer-margin-between:       .5rem;

$modal-dialog-margin:               .5rem;
$modal-dialog-margin-y-sm-up:       1.75rem;

$modal-title-line-height:           $line-height-base;

$modal-content-color:               null;
$modal-content-bg:                  $white;
$modal-content-border-color:        var(--#{$prefix}border-color-translucent);
$modal-content-border-width:        $border-width;
$modal-content-border-radius:       $border-radius-lg;
$modal-content-inner-border-radius: subtract($modal-content-border-radius,
$modal-content-border-width);
$modal-content-box-shadow-xs:       $box-shadow-sm;
$modal-content-box-shadow-sm-up:    $box-shadow;

$modal-backdrop-bg:                 $black;
$modal-backdrop-opacity:            .5;

$modal-header-border-color:         var(--#{$prefix}border-color);
$modal-header-border-width:         $modal-content-border-width;
$modal-header-padding-y:            $modal-inner-padding;
$modal-header-padding-x:            $modal-inner-padding;
$modal-header-padding:              $modal-header-padding-y $modal-header-
padding-x; // Keep this for backwards compatibility

$modal-footer-bg:                   null;
$modal-footer-border-color:         $modal-header-border-color;
$modal-footer-border-width:         $modal-header-border-width;

$modal-sm:                          300px;
$modal-md:                          500px;
$modal-lg:                          800px;
$modal-xl:                          1140px;

$modal-fade-transform:              translate(0, -50px);
$modal-show-transform:              none;
```

```
$modal-transition:                    transform .3s ease-out;
$modal-scale-transform:               scale(1.02);
```

Loop

Responsive fullscreen modals are generated via the `$breakpoints` map and a loop in `scss/_modal.scss`.

```
@each $breakpoint in map-keys($grid-breakpoints) {
  $infix: breakpoint-infix($breakpoint, $grid-breakpoints);
  $postfix: if($infix != "", $infix + "-down", "");

  @include media-breakpoint-down($breakpoint) {
    .modal-fullscreen#{$postfix} {
      width: 100vw;
      max-width: none;
      height: 100%;
      margin: 0;

      .modal-content {
        height: 100%;
        border: 0;
        @include border-radius(0);
      }

      .modal-header,
      .modal-footer {
        @include border-radius(0);
      }

      .modal-body {
        overflow-y: auto;
      }
    }
  }
}
```

Usage

The modal plugin toggles your hidden content on demand, via data attributes or JavaScript. It also overrides default scrolling behavior and generates a `.modal-backdrop` to provide a click area for dismissing shown modals when clicking outside the modal.

Via data attributes

Toggle

Activate a modal without writing JavaScript. Set `data-bs-toggle="modal"` on a controller element, like a button, along with a `data-bs-target="#foo"` or `href="#foo"` to target a specific modal to toggle.

```
<button type="button" data-bs-toggle="modal" data-bs-target="#myModal">Launch
modal</button>
```

Dismiss

Dismissal can be achieved with the `data` attribute on a button **within the modal** as demonstrated below:

```
<button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
```

or on a button **outside the modal** using the `data-bs-target` as demonstrated below:

```
<button type="button" class="btn-close" data-bs-dismiss="modal" data-bs-
target="#my-modal" aria-label="Close"></button>
```

While both ways to dismiss a modal are supported, keep in mind that dismissing from outside a modal does not match the WAI-ARIA modal dialog design pattern. Do this at your own risk.

Via JavaScript

Create a modal with a single line of JavaScript:

```
const myModal = new bootstrap.Modal(document.getElementById('myModal'), options)
// or
const myModalAlternative = new bootstrap.Modal('#myModal', options)
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-bs-`, as in `data-bs-animation=""`. Make sure to change the case type of the option name from camelCase to kebab-case when passing the options via data attributes. For example, use `data-bs-custom-class="beautifier"` instead of `data-bs-customClass="beautifier"`.

As of Bootstrap 5.2.0, all components support an **experimental** reserved data attribute `data-bs-config` that can house simple component configuration as a JSON string. When an element has `data-bs-config='{"delay":0, "title":123}'` and `data-bs-title="456"` attributes, the final `title` value will be `456` and the separate data attributes will override values given on `data-bs-config`. In addition, existing data attributes are able to house JSON values like `data-bs-delay='{"show":0,"hide":150}'`.

Name	Type	Default	Description
`backdrop`	boolean, `'static'`	`true`	Includes a modal-backdrop element. Alternatively, specify `static` for a backdrop which doesn't close the modal when clicked.
`keyboard`	boolean	`true`	Closes the modal when escape key is pressed.
`focus`	boolean	`true`	Puts the focus on the modal when initialized.

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.

Passing options

Activates your content as a modal. Accepts an optional options `object`.

```
const myModal = new bootstrap.Modal('#myModal', {
  keyboard: false
})
```

Method	Description
`toggle`	Manually toggles a modal. **Returns to the caller before the modal has actually been shown or hidden** (i.e. before the `shown.bs.modal` or `hidden.bs.modal` event occurs).
`show`	Manually opens a modal. **Returns to the caller before the modal has actually been shown** (i.e. before the `shown.bs.modal` event occurs). Also, you can pass a DOM element as an argument that can be received in the modal events (as the `relatedTarget` property). (i.e. `const modalToggle = document.getElementById('toggleMyModal'); myModal.show(modalToggle)`
`hide`	Manually hides a modal. **Returns to the caller before the modal has actually been hidden** (i.e. before the `hidden.bs.modal` event occurs).
`handleUpdate`	Manually readjust the modal’s position if the height of a modal changes while it is open (i.e. in case a scrollbar appears).
`dispose`	Destroys an element’s modal. (Removes stored data on the DOM element)
`getInstance`	*Static* method which allows you to get the modal instance associated with a DOM element.
`getOrCreateInstance`	*Static* method which allows you to get the modal instance associated with a DOM element, or create a new one in case it wasn’t initialized.

Events

Bootstrap’s modal class exposes a few events for hooking into modal functionality. All modal events are fired at the modal itself (i.e. at the `<div class="modal">`).

Event	Description
`show.bs.modal`	This event fires immediately when the `show` instance method is called. If caused by a click, the clicked element is available as the `relatedTarget` property of the event.
`shown.bs.modal`	This event is fired when the modal has been made visible to the user (will wait for CSS transitions to complete). If caused by a click, the clicked element is available as the `relatedTarget` property of the event.
`hide.bs.modal`	This event is fired immediately when the `hide` instance method has been called.
`hidden.bs.modal`	This event is fired when the modal has finished being hidden from the user (will wait for CSS transitions to complete).
`hidePrevented.bs.modal`	This event is fired when the modal is shown, its backdrop is

Event	Description
	`static` and a click outside of the modal is performed. The event is also fired when the escape key is pressed and the `keyboard` option is set to `false`.

```
const myModalEl = document.getElementById('myModal')
myModalEl.addEventListener('hidden.bs.modal', event => {
  // do something...
})
```

Navbar

Documentation and examples for Bootstrap’s powerful, responsive navigation header, the navbar. Includes support for branding, navigation, and more, including support for our collapse plugin.

On this page

How it works

Here’s what you need to know before getting started with the navbar:

- Navbars require a wrapping `.navbar` with `.navbar-expand{-sm|-md|-lg|-xl|-xxl}` for responsive collapsing and color scheme classes.
- Navbars and their contents are fluid by default. Change the container to limit their horizontal width in different ways.
- Use our spacing and flex utility classes for controlling spacing and alignment within navbars.
- Navbars are responsive by default, but you can easily modify them to change that. Responsive behavior depends on our Collapse JavaScript plugin.
- Ensure accessibility by using a `<nav>` element or, if using a more generic element such as a `<div>`, add a `role="navigation"` to every navbar to explicitly identify it as a landmark region for users of assistive technologies.

- Indicate the current item by using `aria-current="page"` for the current page or `aria-current="true"` for the current item in a set.
- **New in v5.2.0:** Navbars can be themed with CSS variables that are scoped to the `.navbar` base class. `.navbar-light` has been deprecated and `.navbar-dark` has been rewritten to override CSS variables instead of adding additional styles.

The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the reduced motion section of our accessibility documentation.

Supported content

Navbars come with built-in support for a handful of sub-components. Choose from the following as needed:

- `.navbar-brand` for your company, product, or project name.
- `.navbar-nav` for a full-height and lightweight navigation (including support for dropdowns).
- `.navbar-toggler` for use with our collapse plugin and other navigation toggling behaviors.
- Flex and spacing utilities for any form controls and actions.
- `.navbar-text` for adding vertically centered strings of text.
- `.collapse.navbar-collapse` for grouping and hiding navbar contents by a parent breakpoint.
- Add an optional `.navbar-scroll` to set a `max-height` and scroll expanded navbar content.

Here’s an example of all the sub-components included in a responsive light-themed navbar that automatically collapses at the `lg` (large) breakpoint.

Navbar

- Home
- Link
- Dropdown
- Disabled

html

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Dropdown
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><hr class="dropdown-divider"></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled">Disabled</a>
        </li>
      </ul>
      <form class="d-flex" role="search">
        <input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

This example uses background (`bg-light`) and spacing (`me-auto`, `mb-2`, `mb-lg-0`, `me-2`) utility classes.

Brand

The `.navbar-brand` can be applied to most elements, but an anchor works best, as some elements might require utility classes or custom styles.

Text

Add your text within an element with the `.navbar-brand` class.

Navbar

Navbar

html

```
<!-- As a link -->
<nav class="navbar bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
  </div>
</nav>

<!-- As a heading -->
<nav class="navbar bg-light">
  <div class="container-fluid">
    <span class="navbar-brand mb-0 h1">Navbar</span>
  </div>
</nav>
```

Image

You can replace the text within the `.navbar-brand` with an `<img>`.

html

```
<nav class="navbar bg-light">
  <div class="container">
    <a class="navbar-brand" href="#">
      <img src="/docs/5.2/assets/brand/bootstrap-logo.svg" alt="" width="30"
height="24">
    </a>
  </div>
</nav>
```

Image and text

You can also make use of some additional utilities to add an image and text at the same time. Note the addition of `.d-inline-block` and `.align-text-top` on the `<img>`.

Bootstrap

html

```
<nav class="navbar bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">
      <img src="/docs/5.2/assets/brand/bootstrap-logo.svg" alt="" width="30"
height="24" class="d-inline-block align-text-top">
      Bootstrap
    </a>
  </div>
</nav>
```

Nav

Navbar navigation links build on our `.nav` options with their own modifier class and require the use of toggler classes for proper responsive styling. **Navigation in navbars will also grow to occupy as much horizontal space as possible** to keep your navbar contents securely aligned.

Add the `.active` class on `.nav-link` to indicate the current page.

Please note that you should also add the `aria-current` attribute on the active `.nav-link`.

Navbar

- Home
- Features
- Pricing
- Disabled

html

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Features</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Pricing</a>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled">Disabled</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

And because we use classes for our navs, you can avoid the list-based approach entirely if you like.

Navbar

Home Features Pricing Disabled

html

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
      <div class="navbar-nav">
        <a class="nav-link active" aria-current="page" href="#">Home</a>
        <a class="nav-link" href="#">Features</a>
        <a class="nav-link" href="#">Pricing</a>
        <a class="nav-link disabled">Disabled</a>
      </div>
    </div>
  </div>
</nav>
```

You can also use dropdowns in your navbar. Dropdown menus require a wrapping element for positioning, so be sure to use separate and nested elements for `.nav-item` and `.nav-link` as shown below.

Navbar

- Home
- Features
- Pricing
- Dropdown link

html

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavDropdown">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Features</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Pricing</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#"
id="navbarDropdownMenuLink" role="button" data-bs-toggle="dropdown" aria-
expanded="false">
            Dropdown link
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Forms

Place various form controls and components within a navbar:

html

```
<nav class="navbar bg-light">
  <div class="container-fluid">
    <form class="d-flex" role="search">
      <input class="form-control me-2" type="search" placeholder="Search" aria-
label="Search">
      <button class="btn btn-outline-success" type="submit">Search</button>
    </form>
  </div>
</nav>
```

Immediate child elements of `.navbar` use flex layout and will default to `justify-content: space-between`. Use additional flex utilities as needed to adjust this behavior.

Navbar

html

```
<nav class="navbar bg-light">
  <div class="container-fluid">
    <a class="navbar-brand">Navbar</a>
    <form class="d-flex" role="search">
      <input class="form-control me-2" type="search" placeholder="Search" aria-
label="Search">
      <button class="btn btn-outline-success" type="submit">Search</button>
    </form>
  </div>
</nav>
```

Input groups work, too. If your navbar is an entire form, or mostly a form, you can use the `<form>` element as the container and save some HTML.

@

html

```
<nav class="navbar bg-light">
  <form class="container-fluid">
    <div class="input-group">
      <span class="input-group-text" id="basic-addon1">@</span>
      <input type="text" class="form-control" placeholder="Username" aria-
label="Username" aria-describedby="basic-addon1">
    </div>
  </form>
</nav>
```

Various buttons are supported as part of these navbar forms, too. This is also a great reminder that vertical alignment utilities can be used to align different sized elements.

html

```
<nav class="navbar bg-light">
  <form class="container-fluid justify-content-start">
    <button class="btn btn-outline-success me-2" type="button">Main
button</button>
    <button class="btn btn-sm btn-outline-secondary" type="button">Smaller
button</button>
  </form>
</nav>
```

Text

Navbars may contain bits of text with the help of `.navbar-text`. This class adjusts vertical alignment and horizontal spacing for strings of text.

Navbar text with an inline element

html

```
<nav class="navbar bg-light">
  <div class="container-fluid">
    <span class="navbar-text">
      Navbar text with an inline element
    </span>
  </div>
</nav>
```

Mix and match with other components and utilities as needed.

Navbar w/ text

- Home
- Features
- Pricing

Navbar text with an inline element

html

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar w/ text</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarText" aria-controls="navbarText" aria-expanded="false" aria-
label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarText">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Features</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Pricing</a>
        </li>
      </ul>
      <span class="navbar-text">
        Navbar text with an inline element
      </span>
    </div>
  </div>
</nav>
```

Color schemes

New in v5.2.0: Navbar theming is now powered by CSS variables and `.navbar-light` has been deprecated. CSS variables are applied to `.navbar`, defaulting to the "light" appearance, and can be overridden with `.navbar-dark`.

Navbar themes are easier than ever thanks to Bootstrap's combination of Sass and CSS variables. The default is our "light navbar" for use with light background colors, but you can also apply `.navbar-dark` for dark background colors. Then, customize with `.bg-*` utilities.

Navbar

- Home
- Features
- Pricing
- About

Navbar

- Home
- Features
- Pricing
- About

Navbar

- Home
- Features
- Pricing
- About

```
<nav class="navbar navbar-dark bg-dark">
  <!-- Navbar content -->
</nav>

<nav class="navbar navbar-dark bg-primary">
  <!-- Navbar content -->
</nav>

<nav class="navbar" style="background-color: #e3f2fd;">
  <!-- Navbar content -->
</nav>
```

Containers

Although it's not required, you can wrap a navbar in a `.container` to center it on a page–though note that an inner container is still required. Or you can add a container inside the `.navbar` to only center the contents of a fixed or static top navbar.

Navbar

html

```
<div class="container">
  <nav class="navbar navbar-expand-lg bg-light">
    <div class="container-fluid">
      <a class="navbar-brand" href="#">Navbar</a>
    </div>
  </nav>
</div>
```

Use any of the responsive containers to change how wide the content in your navbar is presented.

Navbar

html

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-md">
```

```
    <a class="navbar-brand" href="#">Navbar</a>
  </div>
</nav>
```

Placement

Use our position utilities to place navbars in non-static positions. Choose from fixed to the top, fixed to the bottom, stickied to the top (scrolls with the page until it reaches the top, then stays there), or stickied to the bottom (scrolls with the page until it reaches the bottom, then stays there).

Fixed navbars use `position: fixed`, meaning they’re pulled from the normal flow of the DOM and may require custom CSS (e.g., `padding-top` on the `<body>`) to prevent overlap with other elements.

Default

html

```
<nav class="navbar bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Default</a>
  </div>
</nav>
```

Fixed top

html

```
<nav class="navbar fixed-top bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Fixed top</a>
  </div>
</nav>
```

Fixed bottom

html

```
<nav class="navbar fixed-bottom bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Fixed bottom</a>
  </div>
</nav>
```

Sticky top

html

```
<nav class="navbar sticky-top bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Sticky top</a>
  </div>
</nav>
```

Sticky bottom

html

```
<nav class="navbar sticky-bottom bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Sticky bottom</a>
  </div>
```

```
</nav>
```

Scrolling

Add `.navbar-nav-scroll` to a `.navbar-nav` (or other navbar sub-component) to enable vertical scrolling within the toggleable contents of a collapsed navbar. By default, scrolling kicks in at `75vh` (or 75% of the viewport height), but you can override that with the local CSS custom property `--bs-navbar-height` or custom styles. At larger viewports when the navbar is expanded, content will appear as it does in a default navbar.

Please note that this behavior comes with a potential drawback of `overflow`—when setting `overflow-y: auto` (required to scroll the content here), `overflow-x` is the equivalent of `auto`, which will crop some horizontal content.

Here's an example navbar using `.navbar-nav-scroll` with `style="--bs-scroll-height: 100px;"`, with some extra margin utilities for optimum spacing.

Navbar scroll

- Home
- Link
- Link
- Link

html

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar scroll</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarScroll" aria-controls="navbarScroll" aria-expanded="false"
aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarScroll">
      <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-
scroll-height: 100px;">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#"
id="navbarScrollingDropdown" role="button" data-bs-toggle="dropdown" aria-
expanded="false">
            Link
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarScrollingDropdown">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><hr class="dropdown-divider"></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled">Link</a>
        </li>
      </ul>
      <form class="d-flex" role="search">
        <input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

Responsive behaviors

Navbars can use `.navbar-toggler`, `.navbar-collapse`, and `.navbar-expand{-sm|-md|-lg|-xl|-xxl}` classes to determine when their content collapses behind a button. In combination with other utilities, you can easily choose when to show or hide particular elements.

For navbars that never collapse, add the `.navbar-expand` class on the navbar. For navbars that always collapse, don't add any `.navbar-expand` class.

Toggler

Navbar togglers are left-aligned by default, but should they follow a sibling element like a `.navbar-brand`, they'll automatically be aligned to the far right. Reversing your markup will reverse the placement of the toggler. Below are examples of different toggle styles.

With no `.navbar-brand` shown at the smallest breakpoint:

Hidden brand

- Home
- Link
- Disabled

html

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarTogglerDemo01" aria-controls="navbarTogglerDemo01" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="#">Hidden brand</a>
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
```

```
          <a class="nav-link disabled">Disabled</a>
        </li>
      </ul>
      <form class="d-flex" role="search">
        <input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

With a brand name shown on the left and toggler on the right:

Navbar

- Home
- Link
- Disabled

html

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarTogglerDemo02" aria-controls="navbarTogglerDemo02" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarTogglerDemo02">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled">Disabled</a>
        </li>
      </ul>
      <form class="d-flex" role="search">
        <input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

With a toggler on the left and brand name on the right:

Navbar

- Home
- Link
- Disabled

html

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarTogglerDemo03" aria-controls="navbarTogglerDemo03" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <a class="navbar-brand" href="#">Navbar</a>
    <div class="collapse navbar-collapse" id="navbarTogglerDemo03">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled">Disabled</a>
        </li>
      </ul>
      <form class="d-flex" role="search">
        <input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

External content

Sometimes you want to use the collapse plugin to trigger a container element for content that structurally sits outside of the `.navbar` . Because our plugin works on the `id` and `data-bs-target` matching, that’s easily done!

html

```
<div class="collapse" id="navbarToggleExternalContent">
  <div class="bg-dark p-4">
    <h5 class="text-white h4">Collapsed content</h5>
    <span class="text-muted">Toggleable via the navbar brand.</span>
  </div>
</div>
<nav class="navbar navbar-dark bg-dark">
  <div class="container-fluid">
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarToggleExternalContent" aria-
controls="navbarToggleExternalContent" aria-expanded="false" aria-label="Toggle
navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
  </div>
</nav>
```

When you do this, we recommend including additional JavaScript to move the focus programmatically to the container when it is opened. Otherwise, keyboard users and users of assistive technologies will likely have a hard time finding the newly revealed content - particularly if the container that was opened comes *before* the toggler in the document’s structure. We also recommend making sure that the toggler has the `aria-controls` attribute, pointing to the `id` of the content container. In theory, this allows assistive technology users to jump directly from the toggler to the container it controls–but support for this is currently quite patchy.

Offcanvas

Transform your expanding and collapsing navbar into an offcanvas drawer with the offcanvas plugin. We extend both the offcanvas default styles and use our `.navbar-expand-*` classes to create a dynamic and flexible navigation sidebar.

In the example below, to create an offcanvas navbar that is always collapsed across all breakpoints, omit the `.navbar-expand-*` class entirely.

[Offcanvas navbar]

•

html

```
<nav class="navbar bg-light fixed-top">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Offcanvas navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="offcanvas"
data-bs-target="#offcanvasNavbar" aria-controls="offcanvasNavbar">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="offcanvas offcanvas-end" tabindex="-1" id="offcanvasNavbar"
aria-labelledby="offcanvasNavbarLabel">
      <div class="offcanvas-header">
        <h5 class="offcanvas-title" id="offcanvasNavbarLabel">Offcanvas</h5>
        <button type="button" class="btn-close" data-bs-dismiss="offcanvas"
aria-label="Close"></button>
      </div>
      <div class="offcanvas-body">
        <ul class="navbar-nav justify-content-end flex-grow-1 pe-3">
          <li class="nav-item">
            <a class="nav-link active" aria-current="page" href="#">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Link</a>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#"
id="offcanvasNavbarDropdown" role="button" data-bs-toggle="dropdown" aria-
expanded="false">
              Dropdown
            </a>
            <ul class="dropdown-menu" aria-labelledby="offcanvasNavbarDropdown">
              <li><a class="dropdown-item" href="#">Action</a></li>
              <li><a class="dropdown-item" href="#">Another action</a></li>
              <li>
                <hr class="dropdown-divider">
              </li>
              <li><a class="dropdown-item" href="#">Something else here</a></li>
            </ul>
          </li>
        </ul>
        <form class="d-flex" role="search">
          <input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search">
          <button class="btn btn-outline-success" type="submit">Search</button>
        </form>
      </div>
    </div>
  </div>
</nav>
```

To create an offcanvas navbar that expands into a normal navbar at a specific breakpoint like `lg`, use `.navbar-expand-lg`.

```
<nav class="navbar navbar-expand-lg bg-light fixed-top">
  <a class="navbar-brand" href="#">Offcanvas navbar</a>
  <button class="navbar-toggler" type="button" data-bs-toggle="offcanvas" data-
bs-target="#navbarOffcanvasLg" aria-controls="navbarOffcanvasLg">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="offcanvas offcanvas-end" tabindex="-1" id="navbarOffcanvasLg"
aria-labelledby="navbarOffcanvasLgLabel">
    ...
  </div>
</nav>
```

CSS

Variables

Added in v5.2.0

As part of Bootstrap’s evolving CSS variables approach, navbars now use local CSS variables on `.navbar` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}navbar-padding-x: #{if($navbar-padding-x == null, 0, $navbar-
padding-x)};
  --#{$prefix}navbar-padding-y: #{$navbar-padding-y};
  --#{$prefix}navbar-color: #{$navbar-light-color};
  --#{$prefix}navbar-hover-color: #{$navbar-light-hover-color};
  --#{$prefix}navbar-disabled-color: #{$navbar-light-disabled-color};
  --#{$prefix}navbar-active-color: #{$navbar-light-active-color};
  --#{$prefix}navbar-brand-padding-y: #{$navbar-brand-padding-y};
  --#{$prefix}navbar-brand-margin-end: #{$navbar-brand-margin-end};
  --#{$prefix}navbar-brand-font-size: #{$navbar-brand-font-size};
  --#{$prefix}navbar-brand-color: #{$navbar-light-brand-color};
  --#{$prefix}navbar-brand-hover-color: #{$navbar-light-brand-hover-color};
  --#{$prefix}navbar-nav-link-padding-x: #{$navbar-nav-link-padding-x};
  --#{$prefix}navbar-toggler-padding-y: #{$navbar-toggler-padding-y};
  --#{$prefix}navbar-toggler-padding-x: #{$navbar-toggler-padding-x};
  --#{$prefix}navbar-toggler-font-size: #{$navbar-toggler-font-size};
  --#{$prefix}navbar-toggler-icon-bg: #{escape-svg($navbar-light-toggler-icon-
bg)};
  --#{$prefix}navbar-toggler-border-color: #{$navbar-light-toggler-border-
color};
  --#{$prefix}navbar-toggler-border-radius: #{$navbar-toggler-border-radius};
  --#{$prefix}navbar-toggler-focus-width: #{$navbar-toggler-focus-width};
  --#{$prefix}navbar-toggler-transition: #{$navbar-toggler-transition};
```

Some additional CSS variables are also present on `.navbar-nav`:

```
  --#{$prefix}nav-link-padding-x: 0;
  --#{$prefix}nav-link-padding-y: #{$nav-link-padding-y};
  --#{$prefix}nav-link-color: var(--#{$prefix}navbar-color);
  --#{$prefix}nav-link-hover-color: var(--#{$prefix}navbar-hover-color);
  --#{$prefix}nav-link-disabled-color: var(--#{$prefix}navbar-disabled-color);
```

Sass variables

```
$navbar-padding-y:                  $spacer * .5;
$navbar-padding-x:                  null;

$navbar-nav-link-padding-x:         .5rem;

$navbar-brand-font-size:            $font-size-lg;
// Compute the navbar-brand padding-y so the navbar-brand will have the same
height as navbar-text and nav-link
$nav-link-height:                   $font-size-base * $line-height-base + $nav-
link-padding-y * 2;
$navbar-brand-height:               $navbar-brand-font-size * $line-height-base;
$navbar-brand-padding-y:            ($nav-link-height - $navbar-brand-height)
* .5;
$navbar-brand-margin-end:           1rem;

$navbar-toggler-padding-y:          .25rem;
$navbar-toggler-padding-x:          .75rem;
$navbar-toggler-font-size:          $font-size-lg;
$navbar-toggler-border-radius:      $btn-border-radius;
$navbar-toggler-focus-width:        $btn-focus-width;
$navbar-toggler-transition:         box-shadow .15s ease-in-out;

$navbar-dark-color:                 rgba($white, .55);
$navbar-dark-hover-color:           rgba($white, .75);
$navbar-dark-active-color:          $white;
$navbar-dark-disabled-color:        rgba($white, .25);
$navbar-dark-toggler-icon-bg:       url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 30 30'><path stroke='#{$navbar-
dark-color}' stroke-linecap='round' stroke-miterlimit='10' stroke-width='2'
d='M4 7h22M4 15h22M4 23h22'/></svg>");
$navbar-dark-toggler-border-color:  rgba($white, .1);

$navbar-light-color:                rgba($black, .55);
$navbar-light-hover-color:          rgba($black, .7);
$navbar-light-active-color:         rgba($black, .9);
$navbar-light-disabled-color:       rgba($black, .3);
$navbar-light-toggler-icon-bg:      url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 30 30'><path stroke='#{$navbar-
light-color}' stroke-linecap='round' stroke-miterlimit='10' stroke-width='2'
d='M4 7h22M4 15h22M4 23h22'/></svg>");
$navbar-light-toggler-border-color: rgba($black, .1);

$navbar-light-brand-color:                $navbar-light-active-color;
$navbar-light-brand-hover-color:          $navbar-light-active-color;
$navbar-dark-brand-color:                 $navbar-dark-active-color;
$navbar-dark-brand-hover-color:           $navbar-dark-active-color;
```

Sass loop

Responsive navbar expand/collapse classes (e.g., `.navbar-expand-lg`) are combined with the `$breakpoints` map and generated through a loop in `scss/_navbar.scss`.

```
// Generate series of `.navbar-expand-*` responsive classes for configuring
// where your navbar collapses.
.navbar-expand {
  @each $breakpoint in map-keys($grid-breakpoints) {
    $next: breakpoint-next($breakpoint, $grid-breakpoints);
    $infix: breakpoint-infix($next, $grid-breakpoints);

    // stylelint-disable-next-line scss/selector-no-union-class-name
    &#{$infix} {
      @include media-breakpoint-up($next) {
        flex-wrap: nowrap;
        justify-content: flex-start;

        .navbar-nav {
          flex-direction: row;

          .dropdown-menu {
            position: absolute;
          }

          .nav-link {
            padding-right: var(--#{$prefix}navbar-nav-link-padding-x);
            padding-left: var(--#{$prefix}navbar-nav-link-padding-x);
          }
        }

        .navbar-nav-scroll {
          overflow: visible;
        }

        .navbar-collapse {
          display: flex !important; // stylelint-disable-line declaration-no-
important
          flex-basis: auto;
        }

        .navbar-toggler {
          display: none;
        }

        .offcanvas {
          // stylelint-disable declaration-no-important
          position: static;
          z-index: auto;
          flex-grow: 1;
          width: auto !important;
          height: auto !important;
          visibility: visible !important;
          background-color: transparent !important;
          border: 0 !important;
          transform: none !important;
          @include box-shadow(none);
          @include transition(none);
          // stylelint-enable declaration-no-important

          .offcanvas-header {
            display: none;
          }
          .offcanvas-body {
            display: flex;
            flex-grow: 0;
            padding: 0;
            overflow-y: visible;
          }
        }
      }
    }
  }
}
```

Navs and tabs

Documentation and examples for how to use Bootstrap’s included navigation components.

On this page

Base nav

Navigation available in Bootstrap share general markup and styles, from the base `.nav` class to the active and disabled states. Swap modifier classes to switch between each style.

The base `.nav` component is built with flexbox and provide a strong foundation for building all types of navigation components. It includes some style overrides (for working with lists), some link padding for larger hit areas, and basic disabled styling.

The base `.nav` component does not include any `.active` state. The following examples include the class, mainly to demonstrate that this particular class does not trigger any special styling.

To convey the active state to assistive technologies, use the `aria-current` attribute — using the `page` value for current page, or `true` for the current item in a set.

- Active
- Link
- Link
- Disabled

html

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

Classes are used throughout, so your markup can be super flexible. Use `<ul>`s like above, `<ol>` if the order of your items is important, or roll your own with a `<nav>` element. Because the `.nav` uses `display: flex`, the nav links behave the same as nav items would, but without the extra markup.

Active Link Link Disabled

html

```
<nav class="nav">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled">Disabled</a>
</nav>
```

Available styles

Change the style of `.nav`s component with modifiers and utilities. Mix and match as needed, or build your own.

Horizontal alignment

Change the horizontal alignment of your nav with flexbox utilities. By default, navs are left-aligned, but you can easily change them to center or right aligned.

Centered with `.justify-content-center`:

- Active
- Link
- Link

- Disabled

html

```
<ul class="nav justify-content-center">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

Right-aligned with `.justify-content-end`:

- Active
- Link
- Link
- Disabled

html

```
<ul class="nav justify-content-end">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

Vertical

Stack your navigation by changing the flex item direction with the `.flex-column` utility. Need to stack them on some viewports but not others? Use the responsive versions (e.g., `.flex-sm-column`).

- Active
- Link
- Link
- Disabled

html

```
<ul class="nav flex-column">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

As always, vertical navigation is possible without `<ul>`s, too.

Active Link Link Disabled

html

```
<nav class="nav flex-column">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled">Disabled</a>
</nav>
```

Tabs

Takes the basic nav from above and adds the `.nav-tabs` class to generate a tabbed interface. Use them to create tabbable regions with our tab JavaScript plugin.

- Active
- Link
- Link
- Disabled

html

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

Pills

Take that same HTML, but use `.nav-pills` instead:

- Active
- Link
- Link

- Disabled

html

```
<ul class="nav nav-pills">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

Fill and justify

Force your `.nav`'s contents to extend the full available width one of two modifier classes. To proportionately fill all available space with your `.nav-item`s, use `.nav-fill`. Notice that all horizontal space is occupied, but not every nav item has the same width.

- Active
- Much longer nav link
- Link
- Disabled

html

```
<ul class="nav nav-pills nav-fill">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Much longer nav link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

When using a `<nav>`-based navigation, you can safely omit `.nav-item` as only `.nav-link` is required for styling `<a>` elements.

Active Much longer nav link Link Disabled

html

```
<nav class="nav nav-pills nav-fill">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Much longer nav link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled">Disabled</a>
</nav>
```

For equal-width elements, use `.nav-justified`. All horizontal space will be occupied by nav links, but unlike the `.nav-fill` above, every nav item will be the same width.

- Active
- Much longer nav link
- Link
- Disabled

html

```
<ul class="nav nav-pills nav-justified">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Much longer nav link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

Similar to the `.nav-fill` example using a `<nav>`-based navigation.

Active Much longer nav link Link Disabled

html

```
<nav class="nav nav-pills nav-justified">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Much longer nav link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled">Disabled</a>
</nav>
```

Working with flex utilities

If you need responsive nav variations, consider using a series of flexbox utilities. While more verbose, these utilities offer greater customization across responsive breakpoints. In the example below, our nav will be stacked on the lowest breakpoint, then adapt to a horizontal layout that fills the available width starting from the small breakpoint.

Active Longer nav link Link Disabled

html

```
<nav class="nav nav-pills flex-column flex-sm-row">
  <a class="flex-sm-fill text-sm-center nav-link active" aria-current="page"
href="#">Active</a>
  <a class="flex-sm-fill text-sm-center nav-link" href="#">Longer nav link</a>
  <a class="flex-sm-fill text-sm-center nav-link" href="#">Link</a>
  <a class="flex-sm-fill text-sm-center nav-link disabled">Disabled</a>
</nav>
```

Regarding accessibility

If you’re using navs to provide a navigation bar, be sure to add a `role="navigation"` to the most logical parent container of the `<ul>`, or wrap a `<nav>` element around the whole navigation. Do not add the role to the `<ul>` itself, as this would prevent it from being announced as an actual list by assistive technologies.

Note that navigation bars, even if visually styled as tabs with the `.nav-tabs` class, should **not** be given `role="tablist"`, `role="tab"` or `role="tabpanel"` attributes. These are only appropriate for dynamic tabbed interfaces, as described in the Authoring Practices. See JavaScript behavior for dynamic tabbed interfaces in this section for an example. The `aria-current` attribute is not necessary on dynamic tabbed interfaces since our JavaScript handles the selected state by adding `aria-selected="true"` on the active tab.

Using dropdowns

Add dropdown menus with a little extra HTML and the dropdowns JavaScript plugin.

Tabs with dropdowns

- Active
- Dropdown
- Link
- Disabled

html

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown" href="#"
role="button" aria-expanded="false">Dropdown</a>
    <ul class="dropdown-menu">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
      <li><hr class="dropdown-divider"></li>
      <li><a class="dropdown-item" href="#">Separated link</a></li>
    </ul>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

Pills with dropdowns

- Active
- Dropdown
- Link

- Disabled

html

```
<ul class="nav nav-pills">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown" href="#"
role="button" aria-expanded="false">Dropdown</a>
    <ul class="dropdown-menu">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
      <li><hr class="dropdown-divider"></li>
      <li><a class="dropdown-item" href="#">Separated link</a></li>
    </ul>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

CSS

Variables

Added in v5.2.0

As part of Bootstrap’s evolving CSS variables approach, navs now use local CSS variables on `.nav`, `.nav-tabs`, and `.nav-pills` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

On the `.nav` base class:

```
  --#{$prefix}nav-link-padding-x: #{$nav-link-padding-x};
  --#{$prefix}nav-link-padding-y: #{$nav-link-padding-y};
  @include rfs($nav-link-font-size, --#{$prefix}nav-link-font-size);
  --#{$prefix}nav-link-font-weight: #{$nav-link-font-weight}; // stylelint-
disable-line custom-property-empty-line-before
  --#{$prefix}nav-link-color: #{$nav-link-color};
  --#{$prefix}nav-link-hover-color: #{$nav-link-hover-color};
  --#{$prefix}nav-link-disabled-color: #{$nav-link-disabled-color};
```

On the `.nav-tabs` modifier class:

```
  --#{$prefix}nav-tabs-border-width: #{$nav-tabs-border-width};
  --#{$prefix}nav-tabs-border-color: #{$nav-tabs-border-color};
  --#{$prefix}nav-tabs-border-radius: #{$nav-tabs-border-radius};
  --#{$prefix}nav-tabs-link-hover-border-color: #{$nav-tabs-link-hover-border-
color};
  --#{$prefix}nav-tabs-link-active-color: #{$nav-tabs-link-active-color};
  --#{$prefix}nav-tabs-link-active-bg: #{$nav-tabs-link-active-bg};
  --#{$prefix}nav-tabs-link-active-border-color: #{$nav-tabs-link-active-border-
color};
```

On the `.nav-pills` modifier class:

```
  --#{$prefix}nav-pills-border-radius: #{$nav-pills-border-radius};
  --#{$prefix}nav-pills-link-active-color: #{$nav-pills-link-active-color};
  --#{$prefix}nav-pills-link-active-bg: #{$nav-pills-link-active-bg};
```

Sass variables

```
$nav-link-padding-y:                .5rem;
$nav-link-padding-x:                1rem;
$nav-link-font-size:                null;
$nav-link-font-weight:              null;
$nav-link-color:                    var(--#{$prefix}link-color);
$nav-link-hover-color:              var(--#{$prefix}link-hover-color);
$nav-link-transition:               color .15s ease-in-out, background-
color .15s ease-in-out, border-color .15s ease-in-out;
$nav-link-disabled-color:           $gray-600;

$nav-tabs-border-color:             $gray-300;
$nav-tabs-border-width:             $border-width;
$nav-tabs-border-radius:            $border-radius;
$nav-tabs-link-hover-border-color:  $gray-200 $gray-200 $nav-tabs-border-color;
$nav-tabs-link-active-color:        $gray-700;
$nav-tabs-link-active-bg:           $body-bg;
$nav-tabs-link-active-border-color: $gray-300 $gray-300 $nav-tabs-link-active-
bg;

$nav-pills-border-radius:           $border-radius;
$nav-pills-link-active-color:       $component-active-color;
$nav-pills-link-active-bg:          $component-active-bg;
```

JavaScript behavior

Use the tab JavaScript plugin—include it individually or through the compiled `bootstrap.js` file—to extend our navigational tabs and pills to create tabbable panes of local content.

-

This is some placeholder content the **Home tab's** associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other `.nav`-powered navigation.

```
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="home-tab" data-bs-toggle="tab" data-bs-
target="#home-tab-pane" type="button" role="tab" aria-controls="home-tab-pane"
aria-selected="true">Home</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="profile-tab" data-bs-toggle="tab" data-bs-
target="#profile-tab-pane" type="button" role="tab" aria-controls="profile-tab-
pane" aria-selected="false">Profile</button>
  </li>
  <li class="nav-item" role="presentation">
```

```
    <button class="nav-link" id="contact-tab" data-bs-toggle="tab" data-bs-
target="#contact-tab-pane" type="button" role="tab" aria-controls="contact-tab-
pane" aria-selected="false">Contact</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="disabled-tab" data-bs-toggle="tab" data-bs-
target="#disabled-tab-pane" type="button" role="tab" aria-controls="disabled-
tab-pane" aria-selected="false" disabled>Disabled</button>
  </li>
</ul>
<div class="tab-content" id="myTabContent">
  <div class="tab-pane fade show active" id="home-tab-pane" role="tabpanel"
aria-labelledby="home-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="profile-tab-pane" role="tabpanel" aria-
labelledby="profile-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="contact-tab-pane" role="tabpanel" aria-
labelledby="contact-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="disabled-tab-pane" role="tabpanel" aria-
labelledby="disabled-tab" tabindex="0">...</div>
</div>
```

To help fit your needs, this works with `<ul>`-based markup, as shown above, or with any arbitrary “roll your own” markup. Note that if you’re using `<nav>`, you shouldn’t add `role="tablist"` directly to it, as this would override the element’s native role as a navigation landmark. Instead, switch to an alternative element (in the example below, a simple `<div>`) and wrap the `<nav>` around it.

This is some placeholder content the **Home tab's** associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other `.nav`-powered navigation.

```
<nav>
  <div class="nav nav-tabs" id="nav-tab" role="tablist">
    <button class="nav-link active" id="nav-home-tab" data-bs-toggle="tab" data-
bs-target="#nav-home" type="button" role="tab" aria-controls="nav-home" aria-
selected="true">Home</button>
    <button class="nav-link" id="nav-profile-tab" data-bs-toggle="tab" data-bs-
target="#nav-profile" type="button" role="tab" aria-controls="nav-profile" aria-
selected="false">Profile</button>
    <button class="nav-link" id="nav-contact-tab" data-bs-toggle="tab" data-bs-
target="#nav-contact" type="button" role="tab" aria-controls="nav-contact" aria-
selected="false">Contact</button>
    <button class="nav-link" id="nav-disabled-tab" data-bs-toggle="tab" data-bs-
target="#nav-disabled" type="button" role="tab" aria-controls="nav-disabled"
aria-selected="false" disabled>Disabled</button>
  </div>
</nav>
<div class="tab-content" id="nav-tabContent">
  <div class="tab-pane fade show active" id="nav-home" role="tabpanel" aria-
labelledby="nav-home-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="nav-profile" role="tabpanel" aria-
labelledby="nav-profile-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="nav-contact" role="tabpanel" aria-
labelledby="nav-contact-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="nav-disabled" role="tabpanel" aria-
labelledby="nav-disabled-tab" tabindex="0">...</div>
</div>
```

The tabs plugin also works with pills.

-

This is some placeholder content the **Home tab's** associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other `.nav`-powered navigation.

```
<ul class="nav nav-pills mb-3" id="pills-tab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="pills-home-tab" data-bs-toggle="pill"
data-bs-target="#pills-home" type="button" role="tab" aria-controls="pills-home"
aria-selected="true">Home</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="pills-profile-tab" data-bs-toggle="pill" data-
bs-target="#pills-profile" type="button" role="tab" aria-controls="pills-
profile" aria-selected="false">Profile</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="pills-contact-tab" data-bs-toggle="pill" data-
bs-target="#pills-contact" type="button" role="tab" aria-controls="pills-
contact" aria-selected="false">Contact</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="pills-disabled-tab" data-bs-toggle="pill" data-
bs-target="#pills-disabled" type="button" role="tab" aria-controls="pills-
disabled" aria-selected="false" disabled>Disabled</button>
  </li>
</ul>
<div class="tab-content" id="pills-tabContent">
  <div class="tab-pane fade show active" id="pills-home" role="tabpanel" aria-
labelledby="pills-home-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="pills-profile" role="tabpanel" aria-
labelledby="pills-profile-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="pills-contact" role="tabpanel" aria-
labelledby="pills-contact-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="pills-disabled" role="tabpanel" aria-
labelledby="pills-disabled-tab" tabindex="0">...</div>
</div>
```

And with vertical pills. Ideally, for vertical tabs, you should also add `aria-orientation="vertical"` to the tab list container.

This is some placeholder content the **Home tab's** associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other `.nav`-powered navigation.

```
<div class="d-flex align-items-start">
  <div class="nav flex-column nav-pills me-3" id="v-pills-tab" role="tablist"
aria-orientation="vertical">
    <button class="nav-link active" id="v-pills-home-tab" data-bs-toggle="pill"
data-bs-target="#v-pills-home" type="button" role="tab" aria-controls="v-pills-
home" aria-selected="true">Home</button>
```

```
    <button class="nav-link" id="v-pills-profile-tab" data-bs-toggle="pill"
data-bs-target="#v-pills-profile" type="button" role="tab" aria-controls="v-
pills-profile" aria-selected="false">Profile</button>
    <button class="nav-link" id="v-pills-disabled-tab" data-bs-toggle="pill"
data-bs-target="#v-pills-disabled" type="button" role="tab" aria-controls="v-
pills-disabled" aria-selected="false" disabled>Disabled</button>
    <button class="nav-link" id="v-pills-messages-tab" data-bs-toggle="pill"
data-bs-target="#v-pills-messages" type="button" role="tab" aria-controls="v-
pills-messages" aria-selected="false">Messages</button>
    <button class="nav-link" id="v-pills-settings-tab" data-bs-toggle="pill"
data-bs-target="#v-pills-settings" type="button" role="tab" aria-controls="v-
pills-settings" aria-selected="false">Settings</button>
  </div>
  <div class="tab-content" id="v-pills-tabContent">
    <div class="tab-pane fade show active" id="v-pills-home" role="tabpanel"
aria-labelledby="v-pills-home-tab" tabindex="0">...</div>
    <div class="tab-pane fade" id="v-pills-profile" role="tabpanel" aria-
labelledby="v-pills-profile-tab" tabindex="0">...</div>
    <div class="tab-pane fade" id="v-pills-disabled" role="tabpanel" aria-
labelledby="v-pills-disabled-tab" tabindex="0">...</div>
    <div class="tab-pane fade" id="v-pills-messages" role="tabpanel" aria-
labelledby="v-pills-messages-tab" tabindex="0">...</div>
    <div class="tab-pane fade" id="v-pills-settings" role="tabpanel" aria-
labelledby="v-pills-settings-tab" tabindex="0">...</div>
  </div>
</div>
```

Accessibility

Dynamic tabbed interfaces, as described in the Authoring Practices 1.2, require `role="tablist"`, `role="tab"`, `role="tabpanel"`, and additional `aria-` attributes in order to convey their structure, functionality, and current state to users of assistive technologies (such as screen readers). As a best practice, we recommend using `<button>` elements for the tabs, as these are controls that trigger a dynamic change, rather than links that navigate to a new page or location.

In line with the ARIA Authoring Practices pattern, only the currently active tab receives keyboard focus. When the JavaScript plugin is initialized, it will set `tabindex="-1"` on all inactive tab controls. Once the currently active tab has focus, the cursor keys activate the previous/next tab, with the plugin changing the roving `tabindex` accordingly. However, note that the JavaScript plugin does not distinguish between horizontal and vertical tab lists when it comes to cursor key interactions: regardless of the tab list’s orientation, both the up *and* left cursor go to the previous tab, and down *and* right cursor go to the next tab.

In general, to facilitate keyboard navigation, it’s recommended to make the tab panels themselves focusable as well, unless the first element containing meaningful content inside the tab panel is already focusable. The JavaScript plugin does not try to handle this aspect—where appropriate, you’ll need to explicitly make your tab panels focusable by adding `tabindex="0"` in your markup.
The tab JavaScript plugin **does not** support tabbed interfaces that contain dropdown menus, as these cause both usability and accessibility issues. From a usability perspective, the fact that the currently displayed tab’s trigger element is not immediately visible (as it’s inside the closed dropdown menu) can cause confusion. From an accessibility point of view, there is currently no sensible way to map

this sort of construct to a standard WAI ARIA pattern, meaning that it cannot be easily made understandable to users of assistive technologies.

Using data attributes

You can activate a tab or pill navigation without writing any JavaScript by simply specifying `data-bs-toggle="tab"` or `data-bs-toggle="pill"` on an element. Use these data attributes on `.nav-tabs` or `.nav-pills`.

```
<!-- Nav tabs -->
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="home-tab" data-bs-toggle="tab" data-bs-
target="#home" type="button" role="tab" aria-controls="home" aria-
selected="true">Home</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="profile-tab" data-bs-toggle="tab" data-bs-
target="#profile" type="button" role="tab" aria-controls="profile" aria-
selected="false">Profile</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="messages-tab" data-bs-toggle="tab" data-bs-
target="#messages" type="button" role="tab" aria-controls="messages" aria-
selected="false">Messages</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="settings-tab" data-bs-toggle="tab" data-bs-
target="#settings" type="button" role="tab" aria-controls="settings" aria-
selected="false">Settings</button>
  </li>
</ul>

<!-- Tab panes -->
<div class="tab-content">
  <div class="tab-pane active" id="home" role="tabpanel" aria-labelledby="home-
tab" tabindex="0">...</div>
  <div class="tab-pane" id="profile" role="tabpanel" aria-labelledby="profile-
tab" tabindex="0">...</div>
  <div class="tab-pane" id="messages" role="tabpanel" aria-labelledby="messages-
tab" tabindex="0">...</div>
  <div class="tab-pane" id="settings" role="tabpanel" aria-labelledby="settings-
tab" tabindex="0">...</div>
</div>
```

Via JavaScript

Enable tabbable tabs via JavaScript (each tab needs to be activated individually):

```
const triggerTabList = document.querySelectorAll('#myTab button')
triggerTabList.forEach(triggerEl => {
  const tabTrigger = new bootstrap.Tab(triggerEl)

  triggerEl.addEventListener('click', event => {
    event.preventDefault()
    tabTrigger.show()
  })
})
```

You can activate individual tabs in several ways:

```
const triggerEl = document.querySelector('#myTab button[data-bs-
target="#profile"]')
bootstrap.Tab.getInstance(triggerEl).show() // Select tab by name

const triggerFirstTabEl = document.querySelector('#myTab li:first-child button')
bootstrap.Tab.getInstance(triggerFirstTabEl).show() // Select first tab
```

Fade effect

To make tabs fade in, add `.fade` to each `.tab-pane`. The first tab pane must also have `.show` to make the initial content visible.

```
<div class="tab-content">
  <div class="tab-pane fade show active" id="home" role="tabpanel" aria-
labelledby="home-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="profile" role="tabpanel" aria-
labelledby="profile-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="messages" role="tabpanel" aria-
labelledby="messages-tab" tabindex="0">...</div>
  <div class="tab-pane fade" id="settings" role="tabpanel" aria-
labelledby="settings-tab" tabindex="0">...</div>
</div>
```

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.

constructor

Activates a tab element and content container. Tab should have either a `data-bs-target` or, if using a link, an `href` attribute, targeting a container node in the DOM.

```
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="home-tab" data-bs-toggle="tab" data-bs-
target="#home" type="button" role="tab" aria-controls="home" aria-
selected="true">Home</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="profile-tab" data-bs-toggle="tab" data-bs-
target="#profile" type="button" role="tab" aria-controls="profile" aria-
selected="false">Profile</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="messages-tab" data-bs-toggle="tab" data-bs-
target="#messages" type="button" role="tab" aria-controls="messages" aria-
selected="false">Messages</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="settings-tab" data-bs-toggle="tab" data-bs-
target="#settings" type="button" role="tab" aria-controls="settings" aria-
selected="false">Settings</button>
  </li>
</ul>
```

```
<div class="tab-content">
  <div class="tab-pane active" id="home" role="tabpanel" aria-labelledby="home-tab" tabindex="0">...</div>
  <div class="tab-pane" id="profile" role="tabpanel" aria-labelledby="profile-tab" tabindex="0">...</div>
  <div class="tab-pane" id="messages" role="tabpanel" aria-labelledby="messages-tab" tabindex="0">...</div>
  <div class="tab-pane" id="settings" role="tabpanel" aria-labelledby="settings-tab" tabindex="0">...</div>
</div>

<script>
  const firstTabEl = document.querySelector('#myTab li:last-child button')
  const firstTab = new bootstrap.Tab(firstTabEl)

  firstTab.show()
</script>
```

show

Selects the given tab and shows its associated pane. Any other tab that was previously selected becomes unselected and its associated pane is hidden. **Returns to the caller before the tab pane has actually been shown** (i.e. before the `shown.bs.tab` event occurs).

```
const someTabTriggerEl = document.querySelector('#someTabTrigger')
const tab = new bootstrap.Tab(someTabTriggerEl)

tab.show()
```

dispose

Destroys an element’s tab.

getInstance

Static method which allows you to get the tab instance associated with a DOM element

```
const tab = bootstrap.Tab.getInstance('#trigger') // Returns a Bootstrap tab instance
```

getOrCreateInstance

Static method which allows you to get the tab instance associated with a DOM element, or create a new one in case it wasn’t initialized

```
const tab = bootstrap.Tab.getOrCreateInstance('#trigger') // Returns a Bootstrap tab instance
```

Events

When showing a new tab, the events fire in the following order:

1. `hide.bs.tab` (on the current active tab)
2. `show.bs.tab` (on the to-be-shown tab)
3. `hidden.bs.tab` (on the previous active tab, the same one as for the `hide.bs.tab` event)

4. `shown.bs.tab` (on the newly-active just-shown tab, the same one as for the `show.bs.tab` event)

If no tab was already active, then the `hide.bs.tab` and `hidden.bs.tab` events will not be fired.

Event type	Description
`show.bs.tab`	This event fires on tab show, but before the new tab has been shown. Use `event.target` and `event.relatedTarget` to target the active tab and the previous active tab (if available) respectively.
`shown.bs.tab`	This event fires on tab show after a tab has been shown. Use `event.target` and `event.relatedTarget` to target the active tab and the previous active tab (if available) respectively.
`hide.bs.tab`	This event fires when a new tab is to be shown (and thus the previous active tab is to be hidden). Use `event.target` and `event.relatedTarget` to target the current active tab and the new soon-to-be-active tab, respectively.
`hidden.bs.tab`	This event fires after a new tab is shown (and thus the previous active tab is hidden). Use `event.target` and `event.relatedTarget` to target the previous active tab and the new active tab, respectively.

```
const tabEl = document.querySelector('button[data-bs-toggle="tab"]')
tabEl.addEventListener('shown.bs.tab', event => {
  event.target // newly activated tab
  event.relatedTarget // previous active tab
})
```

Pagination

Documentation and examples for showing pagination to indicate a series of related content exists across multiple pages.

On this page

Overview

We use a large block of connected links for our pagination, making links hard to miss and easily scalable—all while providing large hit areas. Pagination is built with list HTML elements so screen readers can announce the number of available links. Use a wrapping `<nav>` element to identify it as a navigation section to screen readers and other assistive technologies.

In addition, as pages likely have more than one such navigation section, it’s advisable to provide a descriptive `aria-label` for the `<nav>` to reflect its purpose. For example, if the pagination component is used to navigate between a set of search results, an appropriate label could be `aria-label="Search results pages"`.

- Previous
- 1
- 2
- 3
- Next

html

```
<nav aria-label="Page navigation example">
  <ul class="pagination">
    <li class="page-item"><a class="page-link" href="#">Previous</a></li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item"><a class="page-link" href="#">Next</a></li>
  </ul>
</nav>
```

Working with icons

Looking to use an icon or symbol in place of text for some pagination links? Be sure to provide proper screen reader support with `aria` attributes.

- «
- 1
- 2
- 3
- »

html

```
<nav aria-label="Page navigation example">
  <ul class="pagination">
    <li class="page-item">
      <a class="page-link" href="#" aria-label="Previous">
        <span aria-hidden="true">&laquo;</span>
      </a>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#" aria-label="Next">
        <span aria-hidden="true">&raquo;</span>
      </a>
    </li>
  </ul>
</nav>
```

Disabled and active states

Pagination links are customizable for different circumstances. Use `.disabled` for links that appear un-clickable and `.active` to indicate the current page.

While the `.disabled` class uses `pointer-events: none` to *try* to disable the link functionality of `<a>`s, that CSS property is not yet standardized and doesn't account for keyboard navigation. As such, you should always add `tabindex="-1"` on disabled links and use custom JavaScript to fully disable their functionality.

- Previous
- 1
- 2
- 3
- Next

html

```
<nav aria-label="...">
  <ul class="pagination">
    <li class="page-item disabled">
      <a class="page-link">Previous</a>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item active" aria-current="page">
```

```
      <a class="page-link" href="#">2</a>
    </li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#">Next</a>
    </li>
  </ul>
</nav>
```

You can optionally swap out active or disabled anchors for `<span>`, or omit the anchor in the case of the prev/next arrows, to remove click functionality and prevent keyboard focus while retaining intended styles.

- Previous
- 1
- 2
- 3
- Next

html

```
<nav aria-label="...">
  <ul class="pagination">
    <li class="page-item disabled">
      <span class="page-link">Previous</span>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item active" aria-current="page">
      <span class="page-link">2</span>
    </li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#">Next</a>
    </li>
  </ul>
</nav>
```

Sizing

Fancy larger or smaller pagination? Add `.pagination-lg` or `.pagination-sm` for additional sizes.

- 1
- 2
- 3

html

```
<nav aria-label="...">
  <ul class="pagination pagination-lg">
    <li class="page-item active" aria-current="page">
      <span class="page-link">1</span>
    </li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
  </ul>
</nav>
```

- 1
- 2
- 3

html

```
<nav aria-label="...">
  <ul class="pagination pagination-sm">
    <li class="page-item active" aria-current="page">
      <span class="page-link">1</span>
    </li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
  </ul>
</nav>
```

Alignment

Change the alignment of pagination components with flexbox utilities. For example, with `.justify-content-center`:

- Previous
- 1
- 2
- 3
- Next

html

```
<nav aria-label="Page navigation example">
  <ul class="pagination justify-content-center">
    <li class="page-item disabled">
      <a class="page-link">Previous</a>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#">Next</a>
    </li>
  </ul>
</nav>
```

Or with `.justify-content-end`:

- Previous
- 1
- 2
- 3
- Next

html

```
<nav aria-label="Page navigation example">
  <ul class="pagination justify-content-end">
    <li class="page-item disabled">
      <a class="page-link">Previous</a>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#">Next</a>
    </li>
  </ul>
</nav>
```

CSS

Variables

Added in v5.2.0

As part of Bootstrap's evolving CSS variables approach, pagination now uses local CSS variables on `.pagination` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}pagination-padding-x: #{$pagination-padding-x};
  --#{$prefix}pagination-padding-y: #{$pagination-padding-y};
  @include rfs($pagination-font-size, --#{$prefix}pagination-font-size);
  --#{$prefix}pagination-color: #{$pagination-color};
  --#{$prefix}pagination-bg: #{$pagination-bg};
  --#{$prefix}pagination-border-width: #{$pagination-border-width};
  --#{$prefix}pagination-border-color: #{$pagination-border-color};
  --#{$prefix}pagination-border-radius: #{$pagination-border-radius};
  --#{$prefix}pagination-hover-color: #{$pagination-hover-color};
  --#{$prefix}pagination-hover-bg: #{$pagination-hover-bg};
  --#{$prefix}pagination-hover-border-color: #{$pagination-hover-border-color};
  --#{$prefix}pagination-focus-color: #{$pagination-focus-color};
  --#{$prefix}pagination-focus-bg: #{$pagination-focus-bg};
  --#{$prefix}pagination-focus-box-shadow: #{$pagination-focus-box-shadow};
  --#{$prefix}pagination-active-color: #{$pagination-active-color};
  --#{$prefix}pagination-active-bg: #{$pagination-active-bg};
  --#{$prefix}pagination-active-border-color: #{$pagination-active-border-
color};
  --#{$prefix}pagination-disabled-color: #{$pagination-disabled-color};
  --#{$prefix}pagination-disabled-bg: #{$pagination-disabled-bg};
  --#{$prefix}pagination-disabled-border-color: #{$pagination-disabled-border-
color};
```

Sass variables

```
$pagination-padding-y:              .375rem;
$pagination-padding-x:              .75rem;
$pagination-padding-y-sm:           .25rem;
$pagination-padding-x-sm:           .5rem;
$pagination-padding-y-lg:           .75rem;
$pagination-padding-x-lg:           1.5rem;

$pagination-font-size:              $font-size-base;

$pagination-color:                  var(--#{$prefix}link-color);
$pagination-bg:                     $white;
$pagination-border-radius:          $border-radius;
$pagination-border-width:           $border-width;
```

```
$pagination-margin-start:           calc($pagination-border-width * -1); //
stylelint-disable-line function-disallowed-list
$pagination-border-color:           $gray-300;

$pagination-focus-color:            var(--#{$prefix}link-hover-color);
$pagination-focus-bg:               $gray-200;
$pagination-focus-box-shadow:       $input-btn-focus-box-shadow;
$pagination-focus-outline:          0;

$pagination-hover-color:            var(--#{$prefix}link-hover-color);
$pagination-hover-bg:               $gray-200;
$pagination-hover-border-color:     $gray-300;

$pagination-active-color:           $component-active-color;
$pagination-active-bg:              $component-active-bg;
$pagination-active-border-color:    $pagination-active-bg;

$pagination-disabled-color:         $gray-600;
$pagination-disabled-bg:            $white;
$pagination-disabled-border-color:  $gray-300;

$pagination-transition:              color .15s ease-in-out, background-
color .15s ease-in-out, border-color .15s ease-in-out, box-shadow .15s ease-in-
out;

$pagination-border-radius-sm:       $border-radius-sm;
$pagination-border-radius-lg:       $border-radius-lg;
```

Sass mixins

```
@mixin pagination-size($padding-y, $padding-x, $font-size, $border-radius) {
  --#{$prefix}pagination-padding-x: #{$padding-x};
  --#{$prefix}pagination-padding-y: #{$padding-y};
  @include rfs($font-size, --#{$prefix}pagination-font-size);
  --#{$prefix}pagination-border-radius: #{$border-radius}; // stylelint-disable-
line custom-property-empty-line-before
}
```

Placeholders

Use loading placeholders for your components or pages to indicate something may still be loading.

On this page

About

Placeholders can be used to enhance the experience of your application. They’re built only with HTML and CSS, meaning you don’t need any JavaScript to create them. You will, however, need some custom JavaScript to toggle their visibility. Their appearance, color, and sizing can be easily customized with our utility classes.

Example

In the example below, we take a typical card component and recreate it with placeholders applied to create a “loading card”. Size and proportions are the same between the two.

Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

```
<div class="card">
  <img src="..." class="card-img-top" alt="...">

  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>

<div class="card" aria-hidden="true">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title placeholder-glow">
      <span class="placeholder col-6"></span>
    </h5>
    <p class="card-text placeholder-glow">
      <span class="placeholder col-7"></span>
      <span class="placeholder col-4"></span>
      <span class="placeholder col-4"></span>
      <span class="placeholder col-6"></span>
      <span class="placeholder col-8"></span>
    </p>
    <a href="#" tabindex="-1" class="btn btn-primary disabled placeholder col-
6"></a>
  </div>
</div>
```

How it works

Create placeholders with the `.placeholder` class and a grid column class (e.g., `.col-6`) to set the `width`. They can replace the text inside an element or be added as a modifier class to an existing component.

We apply additional styling to `.btn`s via `::before` to ensure the `height` is respected. You may extend this pattern for other situations as needed, or add a `&nbsp;` within the element to reflect the height when actual text is rendered in its place.

html

```
<p aria-hidden="true">
  <span class="placeholder col-6"></span>
</p>

<a href="#" tabindex="-1" class="btn btn-primary disabled placeholder col-4"
aria-hidden="true"></a>
```

The use of `aria-hidden="true"` only indicates that the element should be hidden to screen readers. The *loading* behavior of the placeholder depends on how authors will actually use the placeholder styles, how they plan to update things, etc. Some JavaScript code may be needed to *swap* the state of the placeholder and inform AT users of the update.

Width

You can change the `width` through grid column classes, width utilities, or inline styles.

html

```
<span class="placeholder col-6"></span>
<span class="placeholder w-75"></span>
<span class="placeholder" style="width: 25%;"></span>
```

Color

By default, the `placeholder` uses `currentColor`. This can be overridden with a custom color or utility class.

html

```
<span class="placeholder col-12"></span>

<span class="placeholder col-12 bg-primary"></span>
<span class="placeholder col-12 bg-secondary"></span>
<span class="placeholder col-12 bg-success"></span>
<span class="placeholder col-12 bg-danger"></span>
<span class="placeholder col-12 bg-warning"></span>
<span class="placeholder col-12 bg-info"></span>
```

```
<span class="placeholder col-12 bg-light"></span>
<span class="placeholder col-12 bg-dark"></span>
```

Sizing

The size of `.placeholder`s are based on the typographic style of the parent element. Customize them with sizing modifiers: `.placeholder-lg`, `.placeholder-sm`, or `.placeholder-xs`.

html

```
<span class="placeholder col-12 placeholder-lg"></span>
<span class="placeholder col-12"></span>
<span class="placeholder col-12 placeholder-sm"></span>
<span class="placeholder col-12 placeholder-xs"></span>
```

Animation

Animate placeholders with `.placeholder-glow` or `.placeholder-wave` to better convey the perception of something being *actively* loaded.

html

```
<p class="placeholder-glow">
  <span class="placeholder col-12"></span>
</p>

<p class="placeholder-wave">
  <span class="placeholder col-12"></span>
</p>
```

Sass

Variables

```
$placeholder-opacity-max:           .5;
$placeholder-opacity-min:           .2;
```

Popovers

Documentation and examples for adding Bootstrap popovers, like those found in iOS, to any element on your site.

On this page

Overview

Things to know when using the popover plugin:

- Popovers rely on the third party library Popper for positioning. You must include popper.min.js before `bootstrap.js`, or use one `bootstrap.bundle.min.js` which contains Popper.
- Popovers require the popover plugin as a dependency.
- Popovers are opt-in for performance reasons, so **you must initialize them yourself**.
- Zero-length `title` and `content` values will never show a popover.
- Specify `container: 'body'` to avoid rendering problems in more complex components (like our input groups, button groups, etc).
- Triggering popovers on hidden elements will not work.
- Popovers for `.disabled` or `disabled` elements must be triggered on a wrapper element.
- When triggered from anchors that wrap across multiple lines, popovers will be centered between the anchors' overall width. Use `.text-nowrap` on your `<a>`s to avoid this behavior.
- Popovers must be hidden before their corresponding elements have been removed from the DOM.
- Popovers can be triggered thanks to an element inside a shadow DOM.

By default, this component uses the built-in content sanitizer, which strips out any HTML elements that are not explicitly allowed. See the sanitizer section in our JavaScript documentation for more details.
The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the reduced motion section of our accessibility documentation.

Keep reading to see how popovers work with some examples.

Examples

Enable popovers

As mentioned above, you must initialize popovers before they can be used. One way to initialize all popovers on a page would be to select them by their `data-bs-toggle` attribute, like so:

```
const popoverTriggerList = document.querySelectorAll('[data-bs-
toggle="popover"]')
const popoverList = [...popoverTriggerList].map(popoverTriggerEl => new
bootstrap.Popover(popoverTriggerEl))
```

Live demo

We use JavaScript similar to the snippet above to render the following live popover. Titles are set via `title` attribute and body content is set via `data-bs-content`.

html

```
<button type="button" class="btn btn-lg btn-danger" data-bs-toggle="popover"
title="Popover title" data-bs-content="And here's some amazing content. It's
very engaging. Right?">Click to toggle popover</button>
```

Four directions

Four options are available: top, right, bottom, and left. Directions are mirrored when using Bootstrap in RTL. Set `data-bs-placement` to change the direction.

html

```
<button type="button" class="btn btn-secondary" data-bs-container="body" data-
bs-toggle="popover" data-bs-placement="top" data-bs-content="Top popover">
  Popover on top
</button>
<button type="button" class="btn btn-secondary" data-bs-container="body" data-
bs-toggle="popover" data-bs-placement="right" data-bs-content="Right popover">
  Popover on right
</button>
<button type="button" class="btn btn-secondary" data-bs-container="body" data-
bs-toggle="popover" data-bs-placement="bottom" data-bs-content="Bottom popover">
  Popover on bottom
</button>
<button type="button" class="btn btn-secondary" data-bs-container="body" data-
bs-toggle="popover" data-bs-placement="left" data-bs-content="Left popover">
  Popover on left
</button>
```

Custom `container`

When you have some styles on a parent element that interfere with a popover, you'll want to specify a custom `container` so that the popover's HTML appears within that element instead. This is common in responsive tables, input groups, and the like.

```
const popover = new bootstrap.Popover('.example-popover', {
  container: 'body'
})
```

Custom popovers

Added in v5.2.0

You can customize the appearance of popovers using CSS variables. We set a custom class with `data-bs-custom-class="custom-popover"` to scope our custom appearance and use it to override some of the local CSS variables.

```
.custom-popover {
  --bs-popover-max-width: 200px;
  --bs-popover-border-color: var(--bs-primary);
  --bs-popover-header-bg: var(--bs-primary);
  --bs-popover-header-color: var(--bs-white);
  --bs-popover-body-padding-x: 1rem;
  --bs-popover-body-padding-y: .5rem;
}
```

html

```
<button type="button" class="btn btn-secondary"
        data-bs-toggle="popover" data-bs-placement="right"
        data-bs-custom-class="custom-popover"
        title="Custom popover"
        data-bs-content="This popover is themed via CSS variables.">
  Custom popover
</button>
```

Dismiss on next click

Use the `focus` trigger to dismiss popovers on the user's next click of a different element than the toggle element.

Specific markup required for dismiss-on-next-click

For proper cross-browser and cross-platform behavior, you must use the `<a>` tag, *not* the `<button>` tag, and you also must include a `tabindex` attribute.

html

```
<a tabindex="0" class="btn btn-lg btn-danger" role="button" data-bs-
toggle="popover" data-bs-trigger="focus" title="Dismissible popover" data-bs-
content="And here's some amazing content. It's very engaging.
Right?">Dismissible popover</a>
```

```
const popover = new bootstrap.Popover('.popover-dismiss', {
  trigger: 'focus'
})
```

Disabled elements

Elements with the `disabled` attribute aren't interactive, meaning users cannot hover or click them to trigger a popover (or tooltip). As a workaround, you'll want to trigger the popover from a wrapper `<div>` or `<span>`, ideally made keyboard-focusable using `tabindex="0"`.

For disabled popover triggers, you may also prefer `data-bs-trigger="hover focus"` so that the popover appears as immediate visual feedback to your users as they may not expect to *click* on a disabled element.

html

```
<span class="d-inline-block" tabindex="0" data-bs-toggle="popover" data-bs-
trigger="hover focus" data-bs-content="Disabled popover">
  <button class="btn btn-primary" type="button" disabled>Disabled
button</button>
</span>
```

CSS

Variables

Added in v5.2.0

As part of Bootstrap's evolving CSS variables approach, popovers now use local CSS variables on `.popover` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}popover-zindex: #{$zindex-popover};
  --#{$prefix}popover-max-width: #{$popover-max-width};
  @include rfs($popover-font-size, --#{$prefix}popover-font-size);
  --#{$prefix}popover-bg: #{$popover-bg};
  --#{$prefix}popover-border-width: #{$popover-border-width};
  --#{$prefix}popover-border-color: #{$popover-border-color};
  --#{$prefix}popover-border-radius: #{$popover-border-radius};
  --#{$prefix}popover-inner-border-radius: #{$popover-inner-border-radius};
  --#{$prefix}popover-box-shadow: #{$popover-box-shadow};
  --#{$prefix}popover-header-padding-x: #{$popover-header-padding-x};
  --#{$prefix}popover-header-padding-y: #{$popover-header-padding-y};
  @include rfs($popover-header-font-size, --#{$prefix}popover-header-font-size);
  --#{$prefix}popover-header-color: #{$popover-header-color};
  --#{$prefix}popover-header-bg: #{$popover-header-bg};
  --#{$prefix}popover-body-padding-x: #{$popover-body-padding-x};
  --#{$prefix}popover-body-padding-y: #{$popover-body-padding-y};
  --#{$prefix}popover-body-color: #{$popover-body-color};
  --#{$prefix}popover-arrow-width: #{$popover-arrow-width};
  --#{$prefix}popover-arrow-height: #{$popover-arrow-height};
  --#{$prefix}popover-arrow-border: var(--#{$prefix}popover-border-color);
```

Sass variables

```
$popover-font-size:                 $font-size-sm;
$popover-bg:                        $white;
$popover-max-width:                 276px;
$popover-border-width:              $border-width;
$popover-border-color:              var(--#{$prefix}border-color-translucent);
$popover-border-radius:             $border-radius-lg;
```

```
$popover-inner-border-radius:       subtract($popover-border-radius, $popover-
border-width);
$popover-box-shadow:                $box-shadow;

$popover-header-font-size:          $font-size-base;
$popover-header-bg:                 shade-color($popover-bg, 6%);
$popover-header-color:              var(--#{$prefix}heading-color);
$popover-header-padding-y:          .5rem;
$popover-header-padding-x:          $spacer;

$popover-body-color:                $body-color;
$popover-body-padding-y:            $spacer;
$popover-body-padding-x:            $spacer;

$popover-arrow-width:               1rem;
$popover-arrow-height:              .5rem;
```

Usage

Enable popovers via JavaScript:

```
const exampleEl = document.getElementById('example')
const popover = new bootstrap.Popover(exampleEl, options)
```

Making popovers work for keyboard and assistive technology users

To allow keyboard users to activate your popovers, you should only add them to HTML elements that are traditionally keyboard-focusable and interactive (such as links or form controls). Although arbitrary HTML elements (such as `<span>`s) can be made focusable by adding the `tabindex="0"` attribute, this will add potentially annoying and confusing tab stops on non-interactive elements for keyboard users, and most assistive technologies currently do not announce the popover’s content in this situation. Additionally, do not rely solely on `hover` as the trigger for your popovers, as this will make them impossible to trigger for keyboard users.

While you can insert rich, structured HTML in popovers with the `html` option, we strongly recommend that you avoid adding an excessive amount of content. The way popovers currently work is that, once displayed, their content is tied to the trigger element with the `aria-describedby` attribute. As a result, the entirety of the popover’s content will be announced to assistive technology users as one long, uninterrupted stream.

Additionally, while it is possible to also include interactive controls (such as form elements or links) in your popover (by adding these elements to the `allowList` of allowed attributes and tags), be aware that currently the popover does not manage keyboard focus order. When a keyboard user opens a popover, focus remains on the triggering element, and as the popover usually does not immediately follow the trigger in the document’s structure, there is no guarantee that moving forward/pressing `TAB` will move a keyboard user into the popover itself. In short, simply adding interactive controls to a popover is likely to make these controls unreachable/unusable for keyboard users and users of assistive technologies, or at the very least make for an illogical overall focus order. In these cases, consider using a modal dialog instead.

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-bs-`, as in `data-bs-animation=""`. Make sure to change the case type of the option name from camelCase to kebab-case when passing the options via data attributes. For example, use `data-bs-custom-class="beautifier"` instead of `data-bs-customClass="beautifier"`.

As of Bootstrap 5.2.0, all components support an **experimental** reserved data attribute `data-bs-config` that can house simple component configuration as a JSON string. When an element has `data-bs-config='{"delay":0, "title":123}'` and `data-bs-title="456"` attributes, the final `title` value will be `456` and the separate data attributes will override values given on `data-bs-config`. In addition, existing data attributes are able to house JSON values like `data-bs-delay='{"show":0,"hide":150}'`.

Note that for security reasons the `sanitize`, `sanitizeFn`, and `allowList` options cannot be supplied using data attributes.

Name	Type	Default	Description
`animation`	boolean	`true`	Apply a CSS fade transition to the popover
`container`	string, element, false	`false`	Appends the popover to a specific element. Example: `container: 'body'`. This option is particularly useful in that it allows you to position the popover in the flow of the document near the triggering element - which will prevent the popover from floating away from the triggering element during a window resize.
`content`	string, element, function	`''`	Default content value if `data-bs-content` attribute isn’t present. If a function is given, it will be called with its `this` reference set to the element that the popover is attached to.
`delay`	number, object	`0`	Delay showing and hiding the popover (ms)—doesn’t apply to manual trigger type. If a number is supplied, delay is applied to both hide/show. Object structure is: `delay: { "show": 500, "hide": 100 }`.
`html`	boolean	`false`	Allow HTML in the popover. If true, HTML tags in the popover’s `title` will be rendered in the popover. If false, `innerText` property will be used to insert content into the DOM. Use text if

Name	Type	Default	Description
			you’re worried about XSS attacks.
`placement`	string, function	`'top'`	How to position the popover: auto, top, bottom, left, right. When `auto` is specified, it will dynamically reorient the popover. When a function is used to determine the placement, it is called with the popover DOM node as its first argument and the triggering element DOM node as its second. The `this` context is set to the popover instance.
`selector`	string, false	`false`	If a selector is provided, popover objects will be delegated to the specified targets. In practice, this is used to also apply popovers to dynamically added DOM elements (`jQuery.on` support). See this issue and an informative example.
`template`	string	`'<div class="popover" role="popover"><div class="popover-arrow"></div><div class="popover-inner"></div></div>'`	Base HTML to use when creating the popover. The popover’s `title` will be injected into the `.popover-inner`. `.popover-arrow` will become the popover’s arrow. The outermost wrapper element should have the `.popover` class and `role="popover"`.
`title`	string, element, function	`''`	Default title value if `title` attribute isn’t present. If a function is given, it will be called with its `this` reference set to the element that the popover is attached to.
`customClass`	string, function	`''`	Add classes to the popover when it is shown. Note that these classes will be added in addition to any classes specified in the template. To add multiple classes, separate them with spaces: `'class-1 class-2'`. You can also pass a function that should return a single string containing additional class names.
`trigger`	string	`'hover focus'`	How popover is triggered: click, hover, focus, manual. You may pass multiple triggers; separate them with a space. `'manual'`

Name	Type	Default	Description
			indicates that the popover will be triggered programmatically via the `.popover('show')`, `.popover('hide')` and `.popover('toggle')` methods; this value cannot be combined with any other trigger. `'hover'` on its own will result in popovers that cannot be triggered via the keyboard, and should only be used if alternative methods for conveying the same information for keyboard users is present.
`offset`	number, string, function	`[0, 0]`	Offset of the popover relative to its target. You can pass a string in data attributes with comma separated values like: `data-bs-offset="10,20"`. When a function is used to determine the offset, it is called with an object containing the popper placement, the reference, and popper rects as its first argument. The triggering element DOM node is passed as the second argument. The function must return an array with two numbers: skidding, distance. For more information refer to Popper's offset docs.
`fallbackPlacements`	string, array	`['top', 'right', 'bottom', 'left']`	Define fallback placements by providing a list of placements in array (in order of preference). For more information refer to Popper's behavior docs.
`boundary`	string, element	`'clippingParents'`	Overflow constraint boundary of the popover (applies only to Popper's preventOverflow modifier). By default, it's `'clippingParents'` and can accept an HTMLElement reference (via JavaScript only). For more information refer to Popper's detectOverflow docs.
`sanitize`	boolean	`true`	Enable or disable the sanitization. If activated `'template'`, `'content'` and `'title'` options will be sanitized.
`allowList`	object	Default value	Object which contains allowed

Name	Type	Default	Description
			attributes and tags.
`sanitizeFn`	null, function	`null`	Here you can supply your own sanitize function. This can be useful if you prefer to use a dedicated library to perform sanitization.
`popperConfig`	null, object, function	`null`	To change Bootstrap's default Popper config, see [Popper's configuration](). When a function is used to create the Popper configuration, it's called with an object that contains the Bootstrap's default Popper configuration. It helps you use and merge the default with your own configuration. The function must return a configuration object for Popper.

Data attributes for individual popovers

Options for individual popovers can alternatively be specified through the use of data attributes, as explained above.

Using function with `popperConfig`

```
const popover = new bootstrap.Popover(element, {
  popperConfig(defaultBsPopperConfig) {
    // const newPopperConfig = {...}
    // use defaultBsPopperConfig if needed...
    // return newPopperConfig
  }
})
```

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

[See our JavaScript documentation for more information]().

Method	Description
`show`	Reveals an element's popover. **Returns to the caller before the popover has actually been shown** (i.e. before the `shown.bs.popover` event occurs). This is considered a "manual" triggering of the popover. Popovers whose title and content are both zero-length are never displayed.
`hide`	Hides an element's popover. **Returns to the caller before the popover has actually been hidden** (i.e. before the `hidden.bs.popover` event occurs). This is considered a

Method	Description
	“manual” triggering of the popover.
`toggle`	Toggles an element’s popover. **Returns to the caller before the popover has actually been shown or hidden** (i.e. before the `shown.bs.popover` or `hidden.bs.popover` event occurs). This is considered a “manual” triggering of the popover.
`dispose`	Hides and destroys an element’s popover (Removes stored data on the DOM element). Popovers that use delegation (which are created using the `selector` option) cannot be individually destroyed on descendant trigger elements.
`enable`	Gives an element’s popover the ability to be shown. **Popovers are enabled by default.**
`disable`	Removes the ability for an element’s popover to be shown. The popover will only be able to be shown if it is re-enabled.
`setContent`	Gives a way to change the popover’s content after its initialization.
`toggleEnabled`	Toggles the ability for an element’s popover to be shown or hidden.
`update`	Updates the position of an element’s popover.
`getInstance`	*Static* method which allows you to get the popover instance associated with a DOM element.
`getOrCreateInstance`	*Static* method which allows you to get the popover instance associated with a DOM element, or create a new one in case it wasn’t initialized

```
// getOrCreateInstance example
const popover = bootstrap.Popover.getOrCreateInstance('#example') // Returns a
Bootstrap popover instance

// setContent example
myPopover.setContent({
  '.popover-header': 'another title',
  '.popover-body': 'another content'
})
```

The `setContent` method accepts an `object` argument, where each property-key is a valid `string` selector within the popover template, and each related property-value can be `string` | `element` | `function` | `null`

Events

Event	Description
`show.bs.popover`	This event fires immediately when the `show` instance method is called.
`shown.bs.popover`	This event is fired when the popover has been made visible to the user (will wait for CSS transitions to complete).
`hide.bs.popover`	This event is fired immediately when the `hide` instance method has been called.
`hidden.bs.popover`	This event is fired when the popover has finished being hidden from the user (will wait for CSS transitions to complete).
`inserted.bs.popover`	This event is fired after the `show.bs.popover` event when the popover template has been added to the DOM.

```
const myPopoverTrigger = document.getElementById('myPopover')
```

```
myPopoverTrigger.addEventListener('hidden.bs.popover', () => {
  // do something...
})
```

Progress

Documentation and examples for using Bootstrap custom progress bars featuring support for stacked bars, animated backgrounds, and text labels.

On this page

How it works

Progress components are built with two HTML elements, some CSS to set the width, and a few attributes. We don't use the HTML5 `<progress>` element, ensuring you can stack progress bars, animate them, and place text labels over them.

- We use the `.progress` as a wrapper to indicate the max value of the progress bar.
- We use the inner `.progress-bar` to indicate the progress so far.
- The `.progress-bar` requires an inline style, utility class, or custom CSS to set their width.
- The `.progress-bar` also requires some `role` and `aria` attributes to make it accessible.

Put that all together, and you have the following examples.

html

```
<div class="progress">
  <div class="progress-bar" role="progressbar" aria-valuenow="0" aria-
valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 25%" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 50%" aria-
valuenow="50" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 75%" aria-
valuenow="75" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
```

```
  <div class="progress-bar" role="progressbar" style="width: 100%" aria-
valuenow="100" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Bootstrap provides a handful of utilities for setting width. Depending on your needs, these may help with quickly configuring progress.

html

```
<div class="progress">
  <div class="progress-bar w-75" role="progressbar" aria-valuenow="75" aria-
valuemin="0" aria-valuemax="100"></div>
</div>
```

Labels

Add labels to your progress bars by placing text within the `.progress-bar`.

25%

html

```
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 25%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="100">25%</div>
</div>
```

Height

We only set a `height` value on the `.progress`, so if you change that value the inner `.progress-bar` will automatically resize accordingly.

html

```
<div class="progress" style="height: 1px;">
  <div class="progress-bar" role="progressbar" style="width: 25%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress" style="height: 20px;">
  <div class="progress-bar" role="progressbar" style="width: 25%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Backgrounds

Use background utility classes to change the appearance of individual progress bars.

html

```
<div class="progress">
  <div class="progress-bar bg-success" role="progressbar" style="width: 25%"
aria-valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar bg-info" role="progressbar" style="width: 50%" aria-
valuenow="50" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
```

```
  <div class="progress-bar bg-warning" role="progressbar" style="width: 75%"
aria-valuenow="75" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar bg-danger" role="progressbar" style="width: 100%"
aria-valuenow="100" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Multiple bars

Include multiple progress bars in a progress component if you need.

html

```
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 15%" aria-
valuenow="15" aria-valuemin="0" aria-valuemax="100"></div>
  <div class="progress-bar bg-success" role="progressbar" style="width: 30%"
aria-valuenow="30" aria-valuemin="0" aria-valuemax="100"></div>
  <div class="progress-bar bg-info" role="progressbar" style="width: 20%" aria-
valuenow="20" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Striped

Add `.progress-bar-striped` to any `.progress-bar` to apply a stripe via CSS gradient over the progress bar’s background color.

html

```
<div class="progress">
  <div class="progress-bar progress-bar-striped" role="progressbar"
style="width: 10%" aria-valuenow="10" aria-valuemin="0"
aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-striped bg-success" role="progressbar"
style="width: 25%" aria-valuenow="25" aria-valuemin="0"
aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-striped bg-info" role="progressbar"
style="width: 50%" aria-valuenow="50" aria-valuemin="0"
aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-striped bg-warning" role="progressbar"
style="width: 75%" aria-valuenow="75" aria-valuemin="0"
aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-striped bg-danger" role="progressbar"
style="width: 100%" aria-valuenow="100" aria-valuemin="0" aria-
valuemax="100"></div>
</div>
```

Animated stripes

The striped gradient can also be animated. Add `.progress-bar-animated` to `.progress-bar` to animate the stripes right to left via CSS3 animations.

html

```
<div class="progress">
  <div class="progress-bar progress-bar-striped progress-bar-animated"
role="progressbar" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100"
style="width: 75%"></div>
</div>
```

CSS

Variables

Added in v5.2.0

As part of Bootstrap's evolving CSS variables approach, progress bars now use local CSS variables on `.progress` for enhanced real-time customization. Values for the CSS variables are set via Sass, so Sass customization is still supported, too.

```
  --#{$prefix}progress-height: #{$progress-height};
  @include rfs($progress-font-size, --#{$prefix}progress-font-size);
  --#{$prefix}progress-bg: #{$progress-bg}; // stylelint-disable-line custom-
property-empty-line-before
  --#{$prefix}progress-border-radius: #{$progress-border-radius};
  --#{$prefix}progress-box-shadow: #{$progress-box-shadow};
  --#{$prefix}progress-bar-color: #{$progress-bar-color};
  --#{$prefix}progress-bar-bg: #{$progress-bar-bg};
  --#{$prefix}progress-bar-transition: #{$progress-bar-transition};
```

Sass variables

```
$progress-height:                   1rem;
$progress-font-size:                $font-size-base * .75;
$progress-bg:                       $gray-200;
$progress-border-radius:            $border-radius;
$progress-box-shadow:               $box-shadow-inset;
$progress-bar-color:                $white;
$progress-bar-bg:                   $primary;
$progress-bar-animation-timing:     1s linear infinite;
$progress-bar-transition:           width .6s ease;
```

Keyframes

Used for creating the CSS animations for `.progress-bar-animated`. Included in `scss/_progress-bar.scss`.

```
@if $enable-transitions {
  @keyframes progress-bar-stripes {
    0% { background-position-x: $progress-height; }
  }
}
```

Color & background

Set a background color with contrasting foreground color.

On this page

- Overview
- With components

Overview

Added in v5.2.0

Color and background helpers combine the power of our `.text-*` utilities and `.bg-*` utilities in one class. Using our Sass `color-contrast()` function, we automatically determine a contrasting `color` for a particular `background-color`.

Heads up! There's currently no support for a CSS-native `color-contrast` function, so we use our own via Sass. This means that customizing our theme colors via CSS variables may cause color contrast issues with these utilities.
Primary with contrasting color
Secondary with contrasting color
Success with contrasting color
Danger with contrasting color
Warning with contrasting color
Info with contrasting color
Light with contrasting color
Dark with contrasting color
html

```html
<div class="text-bg-primary p-3">Primary with contrasting color</div>
<div class="text-bg-secondary p-3">Secondary with contrasting color</div>
<div class="text-bg-success p-3">Success with contrasting color</div>
<div class="text-bg-danger p-3">Danger with contrasting color</div>
<div class="text-bg-warning p-3">Warning with contrasting color</div>
<div class="text-bg-info p-3">Info with contrasting color</div>
<div class="text-bg-light p-3">Light with contrasting color</div>
<div class="text-bg-dark p-3">Dark with contrasting color</div>
```

With components

Use them in place of combined `.text-*` and `.bg-*` classes, like on badges:

Primary Info

html

```html
<span class="badge text-bg-primary">Primary</span>
<span class="badge text-bg-info">Info</span>
```

Or on cards:

Header

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Some quick example text to build on the card title and make up the bulk of the card's content.

html

```
<div class="card text-bg-primary mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-bg-info mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
  </div>
</div>
```

Colored links

Colored links with hover states

You can use the `.link-*` classes to colorize links. Unlike the `.text-*` classes, these classes have a `:hover` and `:focus` state.

Primary link Secondary link Success link Danger link Warning link Info link Light link Dark link

html

```
<a href="#" class="link-primary">Primary link</a>
<a href="#" class="link-secondary">Secondary link</a>
<a href="#" class="link-success">Success link</a>
<a href="#" class="link-danger">Danger link</a>
<a href="#" class="link-warning">Warning link</a>
<a href="#" class="link-info">Info link</a>
<a href="#" class="link-light">Light link</a>
<a href="#" class="link-dark">Dark link</a>
```

Some of the link styles use a relatively light foreground color, and should only be used on a dark background in order to have sufficient contrast.

Position

Use these helpers for quickly configuring the position of an element.

On this page

Fixed top

Position an element at the top of the viewport, from edge to edge. Be sure you understand the ramifications of fixed position in your project; you may need to add additional CSS.

```
<div class="fixed-top">...</div>
```

Fixed bottom

Position an element at the bottom of the viewport, from edge to edge. Be sure you understand the ramifications of fixed position in your project; you may need to add additional CSS.

```
<div class="fixed-bottom">...</div>
```

Sticky top

Position an element at the top of the viewport, from edge to edge, but only after you scroll past it.

```
<div class="sticky-top">...</div>
```

Responsive sticky top

Responsive variations also exist for `.sticky-top` utility.

```
<div class="sticky-sm-top">Stick to the top on viewports sized SM (small) or
wider</div>
<div class="sticky-md-top">Stick to the top on viewports sized MD (medium) or
wider</div>
<div class="sticky-lg-top">Stick to the top on viewports sized LG (large) or
wider</div>
<div class="sticky-xl-top">Stick to the top on viewports sized XL (extra-large)
or wider</div>
<div class="sticky-xxl-top">Stick to the top on viewports sized XXL (extra-
extra-large) or wider</div>
```

Sticky bottom

Position an element at the bottom of the viewport, from edge to edge, but only after you scroll past it.

```
<div class="sticky-bottom">...</div>
```

Responsive sticky bottom

Responsive variations also exist for `.sticky-bottom` utility.

```
<div class="sticky-sm-bottom">Stick to the bottom on viewports sized SM (small)
or wider</div>
<div class="sticky-md-bottom">Stick to the bottom on viewports sized MD (medium)
or wider</div>
<div class="sticky-lg-bottom">Stick to the bottom on viewports sized LG (large)
or wider</div>
<div class="sticky-xl-bottom">Stick to the bottom on viewports sized XL (extra-
large) or wider</div>
<div class="sticky-xxl-bottom">Stick to the bottom on viewports sized XXL
(extra-extra-large) or wider</div>
```

Visually hidden

Use these helpers to visually hide elements but keep them accessible to assistive technologies.

Visually hide an element while still allowing it to be exposed to assistive technologies (such as screen readers) with `.visually-hidden`. Use `.visually-hidden-focusable` to visually hide an element by default, but to display it when it's focused (e.g. by a keyboard-only user). `.visually-hidden-focusable` can also be applied to a container–thanks to `:focus-within`, the container will be displayed when any child element of the container receives focus.

Title for screen readers

Skip to main content
A container with a focusable element.

html

```html
<h2 class="visually-hidden">Title for screen readers</h2>
<a class="visually-hidden-focusable" href="#content">Skip to main content</a>
<div class="visually-hidden-focusable">A container with a <a href="#">focusable
element</a>.</div>
```

Both `visually-hidden` and `visually-hidden-focusable` can also be used as mixins.

```scss
// Usage as a mixin

.visually-hidden-title {
  @include visually-hidden;
}

.skip-navigation {
  @include visually-hidden-focusable;
}
```

Text truncation

Truncate long strings of text with an ellipsis.

For longer content, you can add a `.text-truncate` class to truncate the text with an ellipsis. **Requires `display: inline-block` or `display: block`.**

This text is quite long, and will be truncated once displayed.
This text is quite long, and will be truncated once displayed.

html

```
<!-- Block level -->
<div class="row">
  <div class="col-2 text-truncate">
    This text is quite long, and will be truncated once displayed.
  </div>
</div>

<!-- Inline level -->
<span class="d-inline-block text-truncate" style="max-width: 150px;">
  This text is quite long, and will be truncated once displayed.
</span>
```

Vertical rule

Use the custom vertical rule helper to create vertical dividers like the `<hr>` element.

On this page

How it works

Vertical rules are inspired by the `<hr>` element, allowing you to create vertical dividers in common layouts. They're styled just like `<hr>` elements:

- They're `1px` wide
- They have `min-height` of `1em`
- Their color is set via `currentColor` and `opacity`

Customize them with additional styles as needed.

Example

html

```
<div class="vr"></div>
```

Vertical rules scale their height in flex layouts:

html

```
<div class="d-flex" style="height: 200px;">
  <div class="vr"></div>
</div>
```

With stacks

They can also be used in stacks:

First item
Second item
Third item

html

```
<div class="hstack gap-3">
  <div class="bg-light border">First item</div>
  <div class="bg-light border ms-auto">Second item</div>
  <div class="vr"></div>
  <div class="bg-light border">Third item</div>
</div>
```

Background

Convey meaning through `background-color` and add decoration with gradients.

On this page

Background color

Similar to the contextual text color classes, set the background of an element to any contextual class. Background utilities **do not set `color`**, so in some cases you'll want to use `.text-*` color utilities.

.bg-primary
.bg-secondary
.bg-success
.bg-danger
.bg-warning
.bg-info
.bg-light
.bg-dark
.bg-body
.bg-white
.bg-transparent

html

```html
<div class="p-3 mb-2 bg-primary text-white">.bg-primary</div>
<div class="p-3 mb-2 bg-secondary text-white">.bg-secondary</div>
<div class="p-3 mb-2 bg-success text-white">.bg-success</div>
<div class="p-3 mb-2 bg-danger text-white">.bg-danger</div>
<div class="p-3 mb-2 bg-warning text-dark">.bg-warning</div>
<div class="p-3 mb-2 bg-info text-dark">.bg-info</div>
<div class="p-3 mb-2 bg-light text-dark">.bg-light</div>
<div class="p-3 mb-2 bg-dark text-white">.bg-dark</div>
<div class="p-3 mb-2 bg-body text-dark">.bg-body</div>
<div class="p-3 mb-2 bg-white text-dark">.bg-white</div>
<div class="p-3 mb-2 bg-transparent text-dark">.bg-transparent</div>
```

Background gradient

By adding a `.bg-gradient` class, a linear gradient is added as background image to the backgrounds. This gradient starts with a semi-transparent white which fades out to the bottom.

Do you need a gradient in your custom CSS? Just add `background-image: var(--bs-gradient);`.

.bg-primary.bg-gradient
.bg-secondary.bg-gradient
.bg-success.bg-gradient
.bg-danger.bg-gradient
.bg-warning.bg-gradient
.bg-info.bg-gradient
.bg-light.bg-gradient
.bg-dark.bg-gradient

Opacity

Added in v5.1.0

As of v5.1.0, `background-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 `.bg-success` utility.

```
.bg-success {
  --bs-bg-opacity: 1;
  background-color: rgba(var(--bs-success-rgb), var(--bs-bg-opacity)) !
important;
}
```

We use an RGB version of our `--bs-success` (with the value of `25, 135, 84`) CSS variable and attached a second CSS variable, `--bs-bg-opacity`, for the alpha transparency (with a default value `1` thanks to a local CSS variable). That means anytime you use `.bg-success` now, your computed `color` value is `rgba(25, 135, 84, 1)`. The local CSS variable inside each `.bg-*` 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-bg-opacity` via custom styles or inline styles.

This is default success background
This is 50% opacity success background
html

```
<div class="bg-success p-2 text-white">This is default success background</div>
<div class="bg-success p-2" style="--bs-bg-opacity: .5;">This is 50% opacity
success background</div>
```

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

This is default success background
This is 75% opacity success background
This is 50% opacity success background
This is 25% opacity success background
This is 10% opacity success background
html

```
<div class="bg-success p-2 text-white">This is default success background</div>
<div class="bg-success p-2 text-white bg-opacity-75">This is 75% opacity success
background</div>
<div class="bg-success p-2 text-dark bg-opacity-50">This is 50% opacity success
background</div>
<div class="bg-success p-2 text-dark bg-opacity-25">This is 25% opacity success
background</div>
<div class="bg-success p-2 text-dark bg-opacity-10">This is 10% opacity success
background</div>
```

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 `background-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;

$gradient: linear-gradient(180deg, rgba($white, .15), rgba($white, 0));
```

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

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

Mixins

No mixins are used to generate our background utilities, but we do have some additional mixins for other situations where you’d like to create your own gradients.

```
@mixin gradient-bg($color: null) {
  background-color: $color;

  @if $enable-gradients {
    background-image: var(--#{$prefix}gradient);
  }
}

// Horizontal gradient, from left to right
//
// Creates two color stops, start and end, by specifying a color and position
for each color stop.
@mixin gradient-x($start-color: $gray-700, $end-color: $gray-800, $start-
percent: 0%, $end-percent: 100%) {
  background-image: linear-gradient(to right, $start-color $start-percent, $end-
color $end-percent);
}

// Vertical gradient, from top to bottom
//
// Creates two color stops, start and end, by specifying a color and position
for each color stop.
@mixin gradient-y($start-color: $gray-700, $end-color: $gray-800, $start-
percent: null, $end-percent: null) {
  background-image: linear-gradient(to bottom, $start-color $start-percent,
$end-color $end-percent);
}

@mixin gradient-directional($start-color: $gray-700, $end-color: $gray-800,
$deg: 45deg) {
  background-image: linear-gradient($deg, $start-color, $end-color);
}

@mixin gradient-x-three-colors($start-color: $blue, $mid-color: $purple, $color-
stop: 50%, $end-color: $red) {
  background-image: linear-gradient(to right, $start-color, $mid-color $color-
stop, $end-color);
}

@mixin gradient-y-three-colors($start-color: $blue, $mid-color: $purple, $color-
stop: 50%, $end-color: $red) {
  background-image: linear-gradient($start-color, $mid-color $color-stop, $end-
color);
}

@mixin gradient-radial($inner-color: $gray-700, $outer-color: $gray-800) {
  background-image: radial-gradient(circle, $inner-color, $outer-color);
}

@mixin gradient-striped($color: rgba($white, .15), $angle: 45deg) {
  background-image: linear-gradient($angle, $color 25%, transparent 25%,
transparent 50%, $color 50%, $color 75%, transparent 75%, transparent);
}
```

Utilities API

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

```
    "background-color": (
      property: background-color,
      class: bg,
      local-vars: (
        "bg-opacity": 1
      ),
      values: map-merge(
        $utilities-bg-colors,
        (
          "transparent": transparent
        )
      )
    ),
    "bg-opacity": (
      css-var: true,
      class: bg-opacity,
      values: (
        10: .1,
        25: .25,
        50: .5,
        75: .75,
        100: 1
      )
    ),
```

Borders

Use border utilities to quickly style the border and border-radius of an element. Great for images, buttons, or any other element.

On this page

Border

Use border utilities to add or remove an element’s borders. Choose from all borders or one at a time.

Additive

Add borders to custom elements:

html

```html
<span class="border"></span>
<span class="border-top"></span>
<span class="border-end"></span>
<span class="border-bottom"></span>
<span class="border-start"></span>
```

Subtractive

Or remove borders:

html

```html
<span class="border-0"></span>
<span class="border-top-0"></span>
<span class="border-end-0"></span>
<span class="border-bottom-0"></span>
<span class="border-start-0"></span>
```

Color

Change the border color using utilities built on our theme colors.

html

```
<span class="border border-primary"></span>
<span class="border border-secondary"></span>
<span class="border border-success"></span>
<span class="border border-danger"></span>
<span class="border border-warning"></span>
<span class="border border-info"></span>
<span class="border border-light"></span>
<span class="border border-dark"></span>
<span class="border border-white"></span>
```

Or modify the default `border-color` of a component:

Dangerous heading

Changing border color and width

html

```
<div class="mb-4">
  <label for="exampleFormControlInput1" class="form-label">Email address</label>
  <input type="email" class="form-control border-success"
id="exampleFormControlInput1" placeholder="name@example.com">
</div>

<div class="h4 pb-2 mb-4 text-danger border-bottom border-danger">
  Dangerous heading
</div>

<div class="p-3 bg-info bg-opacity-10 border border-info border-start-0 rounded-
end">
  Changing border color and width
</div>
```

Opacity

Added in v5.2.0

Bootstrap `border-{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 `.border-success` utility.

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

We use an RGB version of our `--bs-success` (with the value of `25, 135, 84`) CSS variable and attached a second CSS variable, `--bs-border-opacity`, for the alpha transparency (with a default value `1` thanks to a local CSS variable). That means anytime you use `.border-success` now, your computed `color` value is `rgba(25, 135, 84, 1)`. The local CSS variable inside each `.border-*` 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-border-opacity` via custom styles or inline styles.

This is default success border
This is 50% opacity success border

html

```
<div class="border border-success p-2 mb-2">This is default success border</div>
<div class="border border-success p-2" style="--bs-border-opacity: .5;">This is
50% opacity success border</div>
```

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

This is default success border
This is 75% opacity success border
This is 50% opacity success border
This is 25% opacity success border
This is 10% opacity success border

html

```
<div class="border border-success p-2 mb-2">This is default success border</div>
<div class="border border-success p-2 mb-2 border-opacity-75">This is 75%
opacity success border</div>
<div class="border border-success p-2 mb-2 border-opacity-50">This is 50%
opacity success border</div>
<div class="border border-success p-2 mb-2 border-opacity-25">This is 25%
opacity success border</div>
<div class="border border-success p-2 border-opacity-10">This is 10% opacity
success border</div>
```

Width

html

```
<span class="border border-1"></span>
<span class="border border-2"></span>
<span class="border border-3"></span>
<span class="border border-4"></span>
<span class="border border-5"></span>
```

Radius

Add classes to an element to easily round its corners.

html

```
<img src="..." class="rounded" alt="...">
<img src="..." class="rounded-top" alt="...">
<img src="..." class="rounded-end" alt="...">
<img src="..." class="rounded-bottom" alt="...">
<img src="..." class="rounded-start" alt="...">
```

```
<img src="..." class="rounded-circle" alt="...">
<img src="..." class="rounded-pill" alt="...">
```

Sizes

Use the scaling classes for larger or smaller rounded corners. Sizes range from `0` to `3`, and can be configured by modifying the utilities API.

html

```
<img src="..." class="rounded-0" alt="...">
<img src="..." class="rounded-1" alt="...">
<img src="..." class="rounded-2" alt="...">
<img src="..." class="rounded-3" alt="...">
<img src="..." class="rounded-4" alt="...">
<img src="..." class="rounded-5" alt="...">
```

CSS

Variables

```
  --#{$prefix}border-width: #{$border-width};
  --#{$prefix}border-style: #{$border-style};
  --#{$prefix}border-color: #{$border-color};
  --#{$prefix}border-color-translucent: #{$border-color-translucent};

  --#{$prefix}border-radius: #{$border-radius};
  --#{$prefix}border-radius-sm: #{$border-radius-sm};
  --#{$prefix}border-radius-lg: #{$border-radius-lg};
  --#{$prefix}border-radius-xl: #{$border-radius-xl};
  --#{$prefix}border-radius-2xl: #{$border-radius-2xl};
  --#{$prefix}border-radius-pill: #{$border-radius-pill};
```

Sass variables

```
$border-width:                1px;
$border-widths: (
  1: 1px,
  2: 2px,
  3: 3px,
  4: 4px,
  5: 5px
);

$border-style:                solid;
$border-color:                $gray-300;
$border-color-translucent:    rgba($black, .175);

$border-radius:               .375rem;
$border-radius-sm:            .25rem;
$border-radius-lg:            .5rem;
$border-radius-xl:            1rem;
$border-radius-2xl:           2rem;
$border-radius-pill:          50rem;
```

Sass mixins

```
@mixin border-radius($radius: $border-radius, $fallback-border-radius: false) {
  @if $enable-rounded {
    border-radius: valid-radius($radius);
  }
  @else if $fallback-border-radius != false {
    border-radius: $fallback-border-radius;
  }
}

@mixin border-top-radius($radius: $border-radius) {
  @if $enable-rounded {
    border-top-left-radius: valid-radius($radius);
    border-top-right-radius: valid-radius($radius);
  }
}

@mixin border-end-radius($radius: $border-radius) {
  @if $enable-rounded {
    border-top-right-radius: valid-radius($radius);
    border-bottom-right-radius: valid-radius($radius);
  }
}

@mixin border-bottom-radius($radius: $border-radius) {
  @if $enable-rounded {
    border-bottom-right-radius: valid-radius($radius);
    border-bottom-left-radius: valid-radius($radius);
  }
}

@mixin border-start-radius($radius: $border-radius) {
  @if $enable-rounded {
    border-top-left-radius: valid-radius($radius);
    border-bottom-left-radius: valid-radius($radius);
  }
}

@mixin border-top-start-radius($radius: $border-radius) {
  @if $enable-rounded {
    border-top-left-radius: valid-radius($radius);
  }
}

@mixin border-top-end-radius($radius: $border-radius) {
  @if $enable-rounded {
    border-top-right-radius: valid-radius($radius);
  }
}

@mixin border-bottom-end-radius($radius: $border-radius) {
  @if $enable-rounded {
    border-bottom-right-radius: valid-radius($radius);
  }
}

@mixin border-bottom-start-radius($radius: $border-radius) {
  @if $enable-rounded {
    border-bottom-left-radius: valid-radius($radius);
  }
}
```

Utilities API

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

```
    "border": (
      property: border,
      values: (
        null: var(--#{$prefix}border-width) var(--#{$prefix}border-style) var(--
#{$prefix}border-color),
        0: 0,
      )
    ),
    "border-top": (
      property: border-top,
      values: (
        null: var(--#{$prefix}border-width) var(--#{$prefix}border-style) var(--
#{$prefix}border-color),
        0: 0,
      )
    ),
    "border-end": (
      property: border-right,
      class: border-end,
      values: (
        null: var(--#{$prefix}border-width) var(--#{$prefix}border-style) var(--
#{$prefix}border-color),
        0: 0,
      )
    ),
    "border-bottom": (
      property: border-bottom,
      values: (
        null: var(--#{$prefix}border-width) var(--#{$prefix}border-style) var(--
#{$prefix}border-color),
        0: 0,
      )
    ),
    "border-start": (
      property: border-left,
      class: border-start,
      values: (
        null: var(--#{$prefix}border-width) var(--#{$prefix}border-style) var(--
#{$prefix}border-color),
        0: 0,
      )
    ),
    "border-color": (
      property: border-color,
      class: border,
      local-vars: (
        "border-opacity": 1
      ),
      values: $utilities-border-colors
    ),
    "border-width": (
      css-var: true,
      css-variable-name: border-width,
      class: border,
      values: $border-widths
    ),
    "border-opacity": (
      css-var: true,
      class: border-opacity,
      values: (
        10: .1,
        25: .25,
        50: .5,
        75: .75,
        100: 1
      )
    ),


    "rounded": (
      property: border-radius,
      class: rounded,
      values: (
        null: var(--#{$prefix}border-radius),
        0: 0,
        1: var(--#{$prefix}border-radius-sm),
        2: var(--#{$prefix}border-radius),
        3: var(--#{$prefix}border-radius-lg),
        4: var(--#{$prefix}border-radius-xl),
        5: var(--#{$prefix}border-radius-2xl),
        circle: 50%,
        pill: var(--#{$prefix}border-radius-pill)
      )
    ),
    "rounded-top": (
      property: border-top-left-radius border-top-right-radius,
      class: rounded-top,
      values: (null: var(--#{$prefix}border-radius))
    ),
    "rounded-end": (
      property: border-top-right-radius border-bottom-right-radius,
      class: rounded-end,
      values: (null: var(--#{$prefix}border-radius))
    ),
    "rounded-bottom": (
      property: border-bottom-right-radius border-bottom-left-radius,
      class: rounded-bottom,
      values: (null: var(--#{$prefix}border-radius))
    ),
    "rounded-start": (
      property: border-bottom-left-radius border-top-left-radius,
      class: rounded-start,
      values: (null: var(--#{$prefix}border-radius))
    ),
```

Colors

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

On this page

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

html

```html
<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

html

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

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
html

```
<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
      )
    ),
```

Display property

Quickly and responsively toggle the display value of components and more with our display utilities. Includes support for some of the more common values, as well as some extras for controlling display when printing.

On this page

How it works

Change the value of the `display` property with our responsive display utility classes. We purposely support only a subset of all possible values for `display`. Classes can be combined for various effects as you need.

Notation

Display utility classes that apply to all breakpoints, from `xs` to `xxl`, have no breakpoint abbreviation in them. This is because those classes are applied from `min-width: 0;` and up, and thus are not bound by a media query. The remaining breakpoints, however, do include a breakpoint abbreviation.

As such, the classes are named using the format:

- `.d-{value}` for `xs`
- `.d-{breakpoint}-{value}` for `sm`, `md`, `lg`, `xl`, and `xxl`.

Where *value* is one of:

- `none`
- `inline`
- `inline-block`
- `block`
- `grid`
- `table`
- `table-cell`
- `table-row`
- `flex`
- `inline-flex`

The display values can be altered by changing the `$displays` variable and recompiling the SCSS.

The media queries affect screen widths with the given breakpoint *or larger*. For example, `.d-lg-none` sets `display: none;` on `lg`, `xl`, and `xxl` screens.

Examples

d-inline
d-inline
html

```
<div class="d-inline p-2 bg-primary text-white">d-inline</div>
<div class="d-inline p-2 bg-dark text-white">d-inline</div>
```

d-block d-block
html

```
<span class="d-block p-2 bg-primary text-white">d-block</span>
<span class="d-block p-2 bg-dark text-white">d-block</span>
```

Hiding elements

For faster mobile-friendly development, use responsive display classes for showing and hiding elements by device. Avoid creating entirely different versions of the same site, instead hide elements responsively for each screen size.

To hide elements simply use the `.d-none` class or one of the `.d-{sm,md,lg,xl,xxl}-none` classes for any responsive screen variation.

To show an element only on a given interval of screen sizes you can combine one `.d-*-none` class with a `.d-*-*` class, for example `.d-none .d-md-block .d-xl-none .d-xxl-none` will hide the element for all screen sizes except on medium and large devices.

Screen size	Class
Hidden on all	`.d-none`
Hidden only on xs	`.d-none .d-sm-block`
Hidden only on sm	`.d-sm-none .d-md-block`
Hidden only on md	`.d-md-none .d-lg-block`
Hidden only on lg	`.d-lg-none .d-xl-block`
Hidden only on xl	`.d-xl-none`
Hidden only on xxl	`.d-xxl-none .d-xxl-block`
Visible on all	`.d-block`
Visible only on xs	`.d-block .d-sm-none`
Visible only on sm	`.d-none .d-sm-block .d-md-none`
Visible only on md	`.d-none .d-md-block .d-lg-none`
Visible only on lg	`.d-none .d-lg-block .d-xl-none`
Visible only on xl	`.d-none .d-xl-block .d-xxl-none`
Visible only on xxl	`.d-none .d-xxl-block`

hide on screens smaller than lg

html

```
<div class="d-lg-none">hide on lg and wider screens</div>
<div class="d-none d-lg-block">hide on screens smaller than lg</div>
```

Display in print

Change the `display` value of elements when printing with our print display utility classes. Includes support for the same `display` values as our responsive `.d-*` utilities.

- `.d-print-none`
- `.d-print-inline`
- `.d-print-inline-block`
- `.d-print-block`
- `.d-print-grid`
- `.d-print-table`
- `.d-print-table-row`
- `.d-print-table-cell`
- `.d-print-flex`
- `.d-print-inline-flex`

The print and display classes can be combined.

Screen Only (Hide on print only)
Hide up to large on screen, but always show on print
html

```
<div class="d-print-none">Screen Only (Hide on print only)</div>
<div class="d-none d-print-block">Print Only (Hide on screen only)</div>
<div class="d-none d-lg-block d-print-block">Hide up to large on screen, but
always show on print</div>
```

Sass

Utilities API

Display utilities are declared in our utilities API in `scss/_utilities.scss`. [Learn how to use the utilities API.]()

```
    "display": (
      responsive: true,
      print: true,
      property: display,
      class: d,
      values: inline inline-block block grid table table-row table-cell flex
inline-flex none
    ),
```

Flex

Quickly manage the layout, alignment, and sizing of grid columns, navigation, components, and more with a full suite of responsive flexbox utilities. For more complex implementations, custom CSS may be necessary.

On this page

Enable flex behaviors

Apply `display` utilities to create a flexbox container and transform **direct children elements** into flex items. Flex containers and items are able to be modified further with additional flex properties.

I'm a flexbox container!

html

```
<div class="d-flex p-2">I'm a flexbox container!</div>
```

I'm an inline flexbox container!

html

```
<div class="d-inline-flex p-2">I'm an inline flexbox container!</div>
```

Responsive variations also exist for `.d-flex` and `.d-inline-flex`.

- `.d-flex`
- `.d-inline-flex`
- `.d-sm-flex`
- `.d-sm-inline-flex`
- `.d-md-flex`
- `.d-md-inline-flex`
- `.d-lg-flex`

- `.d-lg-inline-flex`
- `.d-xl-flex`
- `.d-xl-inline-flex`
- `.d-xxl-flex`
- `.d-xxl-inline-flex`

Direction

Set the direction of flex items in a flex container with direction utilities. In most cases you can omit the horizontal class here as the browser default is `row`. However, you may encounter situations where you needed to explicitly set this value (like responsive layouts).

Use `.flex-row` to set a horizontal direction (the browser default), or `.flex-row-reverse` to start the horizontal direction from the opposite side.

Flex item 1
Flex item 2
Flex item 3
Flex item 1
Flex item 2
Flex item 3

html

```
<div class="d-flex flex-row mb-3">
  <div class="p-2">Flex item 1</div>
  <div class="p-2">Flex item 2</div>
  <div class="p-2">Flex item 3</div>
</div>
<div class="d-flex flex-row-reverse">
  <div class="p-2">Flex item 1</div>
  <div class="p-2">Flex item 2</div>
  <div class="p-2">Flex item 3</div>
</div>
```

Use `.flex-column` to set a vertical direction, or `.flex-column-reverse` to start the vertical direction from the opposite side.

Flex item 1
Flex item 2
Flex item 3
Flex item 1
Flex item 2
Flex item 3

html

```
<div class="d-flex flex-column mb-3">
  <div class="p-2">Flex item 1</div>
  <div class="p-2">Flex item 2</div>
  <div class="p-2">Flex item 3</div>
</div>
<div class="d-flex flex-column-reverse">
  <div class="p-2">Flex item 1</div>
  <div class="p-2">Flex item 2</div>
  <div class="p-2">Flex item 3</div>
</div>
```

Responsive variations also exist for `flex-direction`.

- `.flex-row`
- `.flex-row-reverse`
- `.flex-column`
- `.flex-column-reverse`
- `.flex-sm-row`
- `.flex-sm-row-reverse`
- `.flex-sm-column`
- `.flex-sm-column-reverse`
- `.flex-md-row`
- `.flex-md-row-reverse`
- `.flex-md-column`
- `.flex-md-column-reverse`
- `.flex-lg-row`
- `.flex-lg-row-reverse`
- `.flex-lg-column`
- `.flex-lg-column-reverse`
- `.flex-xl-row`
- `.flex-xl-row-reverse`
- `.flex-xl-column`
- `.flex-xl-column-reverse`
- `.flex-xxl-row`
- `.flex-xxl-row-reverse`
- `.flex-xxl-column`
- `.flex-xxl-column-reverse`

Justify content

Use `justify-content` utilities on flexbox containers to change the alignment of flex items on the main axis (the x-axis to start, y-axis if `flex-direction: column`). Choose from `start` (browser default), `end`, `center`, `between`, `around`, or `evenly`.

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

```
<div class="d-flex justify-content-start">...</div>
<div class="d-flex justify-content-end">...</div>
<div class="d-flex justify-content-center">...</div>
<div class="d-flex justify-content-between">...</div>
<div class="d-flex justify-content-around">...</div>
<div class="d-flex justify-content-evenly">...</div>
```

Responsive variations also exist for `justify-content`.

- `.justify-content-start`
- `.justify-content-end`
- `.justify-content-center`
- `.justify-content-between`
- `.justify-content-around`
- `.justify-content-evenly`
- `.justify-content-sm-start`
- `.justify-content-sm-end`
- `.justify-content-sm-center`
- `.justify-content-sm-between`
- `.justify-content-sm-around`
- `.justify-content-sm-evenly`
- `.justify-content-md-start`
- `.justify-content-md-end`
- `.justify-content-md-center`
- `.justify-content-md-between`
- `.justify-content-md-around`
- `.justify-content-md-evenly`
- `.justify-content-lg-start`
- `.justify-content-lg-end`
- `.justify-content-lg-center`
- `.justify-content-lg-between`
- `.justify-content-lg-around`
- `.justify-content-lg-evenly`
- `.justify-content-xl-start`
- `.justify-content-xl-end`
- `.justify-content-xl-center`
- `.justify-content-xl-between`

- `.justify-content-xl-around`
- `.justify-content-xl-evenly`
- `.justify-content-xxl-start`
- `.justify-content-xxl-end`
- `.justify-content-xxl-center`
- `.justify-content-xxl-between`
- `.justify-content-xxl-around`
- `.justify-content-xxl-evenly`

Align items

Use `align-items` utilities on flexbox containers to change the alignment of flex items on the cross axis (the y-axis to start, x-axis if `flex-direction: column`). Choose from `start`, `end`, `center`, `baseline`, or `stretch` (browser default).

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

```
<div class="d-flex align-items-start">...</div>
<div class="d-flex align-items-end">...</div>
<div class="d-flex align-items-center">...</div>
<div class="d-flex align-items-baseline">...</div>
<div class="d-flex align-items-stretch">...</div>
```

Responsive variations also exist for `align-items`.

- `.align-items-start`
- `.align-items-end`
- `.align-items-center`
- `.align-items-baseline`
- `.align-items-stretch`
- `.align-items-sm-start`
- `.align-items-sm-end`
- `.align-items-sm-center`
- `.align-items-sm-baseline`

- `.align-items-sm-stretch`
- `.align-items-md-start`
- `.align-items-md-end`
- `.align-items-md-center`
- `.align-items-md-baseline`
- `.align-items-md-stretch`
- `.align-items-lg-start`
- `.align-items-lg-end`
- `.align-items-lg-center`
- `.align-items-lg-baseline`
- `.align-items-lg-stretch`
- `.align-items-xl-start`
- `.align-items-xl-end`
- `.align-items-xl-center`
- `.align-items-xl-baseline`
- `.align-items-xl-stretch`
- `.align-items-xxl-start`
- `.align-items-xxl-end`
- `.align-items-xxl-center`
- `.align-items-xxl-baseline`
- `.align-items-xxl-stretch`

Align self

Use `align-self` utilities on flexbox items to individually change their alignment on the cross axis (the y-axis to start, x-axis if `flex-direction: column`). Choose from the same options as `align-items`: `start`, `end`, `center`, `baseline`, or `stretch` (browser default).

Flex item
Aligned flex item
Flex item
Flex item
Aligned flex item
Flex item
Flex item
Aligned flex item
Flex item
Flex item
Aligned flex item
Flex item
Flex item
Aligned flex item
Flex item

```
<div class="align-self-start">Aligned flex item</div>
<div class="align-self-end">Aligned flex item</div>
```

```
<div class="align-self-center">Aligned flex item</div>
<div class="align-self-baseline">Aligned flex item</div>
<div class="align-self-stretch">Aligned flex item</div>
```

Responsive variations also exist for `align-self`.

- `.align-self-start`
- `.align-self-end`
- `.align-self-center`
- `.align-self-baseline`
- `.align-self-stretch`
- `.align-self-sm-start`
- `.align-self-sm-end`
- `.align-self-sm-center`
- `.align-self-sm-baseline`
- `.align-self-sm-stretch`
- `.align-self-md-start`
- `.align-self-md-end`
- `.align-self-md-center`
- `.align-self-md-baseline`
- `.align-self-md-stretch`
- `.align-self-lg-start`
- `.align-self-lg-end`
- `.align-self-lg-center`
- `.align-self-lg-baseline`
- `.align-self-lg-stretch`
- `.align-self-xl-start`
- `.align-self-xl-end`
- `.align-self-xl-center`
- `.align-self-xl-baseline`
- `.align-self-xl-stretch`
- `.align-self-xxl-start`
- `.align-self-xxl-end`
- `.align-self-xxl-center`
- `.align-self-xxl-baseline`
- `.align-self-xxl-stretch`

Fill

Use the `.flex-fill` class on a series of sibling elements to force them into widths equal to their content (or equal widths if their content does not surpass their border-boxes) while taking up all available horizontal space.

Flex item with a lot of content
Flex item
Flex item

html

```
<div class="d-flex">
  <div class="p-2 flex-fill">Flex item with a lot of content</div>
  <div class="p-2 flex-fill">Flex item</div>
  <div class="p-2 flex-fill">Flex item</div>
</div>
```

Responsive variations also exist for `flex-fill`.

- `.flex-fill`
- `.flex-sm-fill`
- `.flex-md-fill`
- `.flex-lg-fill`
- `.flex-xl-fill`
- `.flex-xxl-fill`

Grow and shrink

Use `.flex-grow-*` utilities to toggle a flex item's ability to grow to fill available space. In the example below, the `.flex-grow-1` elements uses all available space it can, while allowing the remaining two flex items their necessary space.

Flex item
Flex item
Third flex item
html

```
<div class="d-flex">
  <div class="p-2 flex-grow-1">Flex item</div>
  <div class="p-2">Flex item</div>
  <div class="p-2">Third flex item</div>
</div>
```

Use `.flex-shrink-*` utilities to toggle a flex item's ability to shrink if necessary. In the example below, the second flex item with `.flex-shrink-1` is forced to wrap its contents to a new line, "shrinking" to allow more space for the previous flex item with `.w-100`.

Flex item
Flex item
html

```
<div class="d-flex">
  <div class="p-2 w-100">Flex item</div>
  <div class="p-2 flex-shrink-1">Flex item</div>
</div>
```

Responsive variations also exist for `flex-grow` and `flex-shrink`.

- `.flex-{grow|shrink}-0`
- `.flex-{grow|shrink}-1`
- `.flex-sm-{grow|shrink}-0`
- `.flex-sm-{grow|shrink}-1`

- `.flex-md-{grow|shrink}-0`
- `.flex-md-{grow|shrink}-1`
- `.flex-lg-{grow|shrink}-0`
- `.flex-lg-{grow|shrink}-1`
- `.flex-xl-{grow|shrink}-0`
- `.flex-xl-{grow|shrink}-1`
- `.flex-xxl-{grow|shrink}-0`
- `.flex-xxl-{grow|shrink}-1`

Auto margins

Flexbox can do some pretty awesome things when you mix flex alignments with auto margins. Shown below are three examples of controlling flex items via auto margins: default (no auto margin), pushing two items to the right (`.me-auto`), and pushing two items to the left (`.ms-auto`).

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

html

```html
<div class="d-flex mb-3">
  <div class="p-2">Flex item</div>
  <div class="p-2">Flex item</div>
  <div class="p-2">Flex item</div>
</div>

<div class="d-flex mb-3">
  <div class="me-auto p-2">Flex item</div>
  <div class="p-2">Flex item</div>
  <div class="p-2">Flex item</div>
</div>

<div class="d-flex mb-3">
  <div class="p-2">Flex item</div>
  <div class="p-2">Flex item</div>
  <div class="ms-auto p-2">Flex item</div>
</div>
```

With align-items

Vertically move one flex item to the top or bottom of a container by mixing `align-items`, `flex-direction: column`, and `margin-top: auto` or `margin-bottom: auto`.

Flex item
Flex item

Flex item
Flex item
Flex item
Flex item

html

```
<div class="d-flex align-items-start flex-column mb-3" style="height: 200px;">
  <div class="mb-auto p-2">Flex item</div>
  <div class="p-2">Flex item</div>
  <div class="p-2">Flex item</div>
</div>

<div class="d-flex align-items-end flex-column mb-3" style="height: 200px;">
  <div class="p-2">Flex item</div>
  <div class="p-2">Flex item</div>
  <div class="mt-auto p-2">Flex item</div>
</div>
```

Wrap

Change how flex items wrap in a flex container. Choose from no wrapping at all (the browser default) with `.flex-nowrap`, wrapping with `.flex-wrap`, or reverse wrapping with `.flex-wrap-reverse`.

Flex item
Flex item
Flex item
Flex item
Flex item

```
<div class="d-flex flex-nowrap">
  ...
</div>
```

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

```
<div class="d-flex flex-wrap">
  ...
```

```
</div>
```

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

```
<div class="d-flex flex-wrap-reverse">
  ...
</div>
```

Responsive variations also exist for `flex-wrap`.

- `.flex-nowrap`
- `.flex-wrap`
- `.flex-wrap-reverse`
- `.flex-sm-nowrap`
- `.flex-sm-wrap`
- `.flex-sm-wrap-reverse`
- `.flex-md-nowrap`
- `.flex-md-wrap`
- `.flex-md-wrap-reverse`
- `.flex-lg-nowrap`
- `.flex-lg-wrap`
- `.flex-lg-wrap-reverse`
- `.flex-xl-nowrap`
- `.flex-xl-wrap`
- `.flex-xl-wrap-reverse`
- `.flex-xxl-nowrap`
- `.flex-xxl-wrap`
- `.flex-xxl-wrap-reverse`

Order

Change the *visual* order of specific flex items with a handful of `order` utilities. We only provide options for making an item first or last, as well as a reset to use the DOM order. As `order` takes any integer value from 0 to 5, add custom CSS for any additional values needed.

First flex item
Second flex item
Third flex item

html

```
<div class="d-flex flex-nowrap">
  <div class="order-3 p-2">First flex item</div>
  <div class="order-2 p-2">Second flex item</div>
  <div class="order-1 p-2">Third flex item</div>
</div>
```

Responsive variations also exist for `order`.

- `.order-0`
- `.order-1`
- `.order-2`
- `.order-3`
- `.order-4`
- `.order-5`
- `.order-sm-0`
- `.order-sm-1`
- `.order-sm-2`
- `.order-sm-3`
- `.order-sm-4`
- `.order-sm-5`
- `.order-md-0`
- `.order-md-1`
- `.order-md-2`
- `.order-md-3`
- `.order-md-4`
- `.order-md-5`
- `.order-lg-0`
- `.order-lg-1`
- `.order-lg-2`
- `.order-lg-3`
- `.order-lg-4`
- `.order-lg-5`
- `.order-xl-0`
- `.order-xl-1`
- `.order-xl-2`
- `.order-xl-3`

- `.order-xl-4`
- `.order-xl-5`
- `.order-xxl-0`
- `.order-xxl-1`
- `.order-xxl-2`
- `.order-xxl-3`
- `.order-xxl-4`
- `.order-xxl-5`

Additionally there are also responsive `.order-first` and `.order-last` classes that change the `order` of an element by applying `order: -1` and `order: 6`, respectively.

- `.order-first`
- `.order-last`
- `.order-sm-first`
- `.order-sm-last`
- `.order-md-first`
- `.order-md-last`
- `.order-lg-first`
- `.order-lg-last`
- `.order-xl-first`
- `.order-xl-last`
- `.order-xxl-first`
- `.order-xxl-last`

Align content

Use `align-content` utilities on flexbox containers to align flex items *together* on the cross axis. Choose from `start` (browser default), `end`, `center`, `between`, `around`, or `stretch`. To demonstrate these utilities, we've enforced `flex-wrap: wrap` and increased the number of flex items.

Heads up! This property has no effect on single rows of flex items.

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

Flex item
Flex item
Flex item

```
<div class="d-flex align-content-start flex-wrap">
  ...
</div>
```

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

```
<div class="d-flex align-content-end flex-wrap">...</div>
```

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

```
<div class="d-flex align-content-center flex-wrap">...</div>
```

Flex item
Flex item
Flex item
Flex item
Flex item

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

```
<div class="d-flex align-content-between flex-wrap">...</div>
```

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

```
<div class="d-flex align-content-around flex-wrap">...</div>
```

Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item
Flex item

```
<div class="d-flex align-content-stretch flex-wrap">...</div>
```

Responsive variations also exist for `align-content`.

- `.align-content-start`
- `.align-content-end`
- `.align-content-center`
- `.align-content-between`
- `.align-content-around`
- `.align-content-stretch`
- `.align-content-sm-start`
- `.align-content-sm-end`
- `.align-content-sm-center`
- `.align-content-sm-between`
- `.align-content-sm-around`
- `.align-content-sm-stretch`
- `.align-content-md-start`
- `.align-content-md-end`
- `.align-content-md-center`
- `.align-content-md-between`
- `.align-content-md-around`
- `.align-content-md-stretch`
- `.align-content-lg-start`
- `.align-content-lg-end`
- `.align-content-lg-center`
- `.align-content-lg-between`
- `.align-content-lg-around`
- `.align-content-lg-stretch`
- `.align-content-xl-start`
- `.align-content-xl-end`
- `.align-content-xl-center`
- `.align-content-xl-between`
- `.align-content-xl-around`
- `.align-content-xl-stretch`
- `.align-content-xxl-start`
- `.align-content-xxl-end`
- `.align-content-xxl-center`
- `.align-content-xxl-between`
- `.align-content-xxl-around`
- `.align-content-xxl-stretch`

Media object

Looking to replicate the media object component from Bootstrap 4? Recreate it in no time with a few flex utilities that allow even more flexibility and customization than before.

This is some content from a media component. You can replace this with any content and adjust it as needed.

html

```
<div class="d-flex">
  <div class="flex-shrink-0">
    <img src="..." alt="...">
  </div>
  <div class="flex-grow-1 ms-3">
    This is some content from a media component. You can replace this with any
content and adjust it as needed.
  </div>
</div>
```

And say you want to vertically center the content next to the image:

This is some content from a media component. You can replace this with any content and adjust it as needed.

html

```
<div class="d-flex align-items-center">
  <div class="flex-shrink-0">
    <img src="..." alt="...">
  </div>
  <div class="flex-grow-1 ms-3">
    This is some content from a media component. You can replace this with any
content and adjust it as needed.
  </div>
</div>
```

Sass

Utilities API

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

```
    "flex": (
      responsive: true,
      property: flex,
      values: (fill: 1 1 auto)
    ),
    "flex-direction": (
      responsive: true,
      property: flex-direction,
      class: flex,
      values: row column row-reverse column-reverse
    ),
    "flex-grow": (
      responsive: true,
      property: flex-grow,
      class: flex,
      values: (
        grow-0: 0,
        grow-1: 1,
      )
    ),
    "flex-shrink": (
      responsive: true,
      property: flex-shrink,
      class: flex,
      values: (
        shrink-0: 0,
        shrink-1: 1,
      )
    ),
    "flex-wrap": (
      responsive: true,
      property: flex-wrap,
      class: flex,
      values: wrap nowrap wrap-reverse
    ),
    "justify-content": (
      responsive: true,
      property: justify-content,
      values: (
        start: flex-start,
        end: flex-end,
        center: center,
        between: space-between,
        around: space-around,
        evenly: space-evenly,
      )
    ),
    "align-items": (
      responsive: true,
      property: align-items,
      values: (
        start: flex-start,
        end: flex-end,
        center: center,
        baseline: baseline,
        stretch: stretch,
      )
    ),
    "align-content": (
      responsive: true,
      property: align-content,
      values: (
        start: flex-start,
        end: flex-end,
        center: center,
        between: space-between,
        around: space-around,
        stretch: stretch,
      )
    ),
    "align-self": (
      responsive: true,
      property: align-self,
      values: (
        auto: auto,
        start: flex-start,
        end: flex-end,
        center: center,
        baseline: baseline,
        stretch: stretch,
      )
    ),
    "order": (
      responsive: true,
      property: order,
      values: (
    first: -1,
    0: 0,
    1: 1,
    2: 2,
    3: 3,
    4: 4,
    5: 5,
    last: 6,
  ),
),
```

Float

Toggle floats on any element, across any breakpoint, using our responsive float utilities.

On this page

Overview

These utility classes float an element to the left or right, or disable floating, based on the current viewport size using the CSS `float` property. `!important` is included to avoid specificity issues. These use the same viewport breakpoints as our grid system. Please be aware float utilities have no effect on flex items.

Float start on all viewport sizes

Float end on all viewport sizes

Don't float on all viewport sizes

html

```
<div class="float-start">Float start on all viewport sizes</div><br>
<div class="float-end">Float end on all viewport sizes</div><br>
<div class="float-none">Don't float on all viewport sizes</div>
```

Responsive

Responsive variations also exist for each `float` value.

Float start on viewports sized SM (small) or wider

Float start on viewports sized MD (medium) or wider

Float start on viewports sized LG (large) or wider

Float start on viewports sized XL (extra-large) or wider

html

```
<div class="float-sm-start">Float start on viewports sized SM (small) or
wider</div><br>
<div class="float-md-start">Float start on viewports sized MD (medium) or
wider</div><br>
<div class="float-lg-start">Float start on viewports sized LG (large) or
wider</div><br>
```

```
<div class="float-xl-start">Float start on viewports sized XL (extra-large) or
wider</div><br>
```

Here are all the support classes:

- `.float-start`
- `.float-end`
- `.float-none`
- `.float-sm-start`
- `.float-sm-end`
- `.float-sm-none`
- `.float-md-start`
- `.float-md-end`
- `.float-md-none`
- `.float-lg-start`
- `.float-lg-end`
- `.float-lg-none`
- `.float-xl-start`
- `.float-xl-end`
- `.float-xl-none`
- `.float-xxl-start`
- `.float-xxl-end`
- `.float-xxl-none`

Sass

Utilities API

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

```
    "float": (
      responsive: true,
      property: float,
      values: (
        start: left,
        end: right,
        none: none,
      )
    ),
```

Opacity

Control the opacity of elements.

The `opacity` property sets the opacity level for an element. The opacity level describes the transparency level, where `1` is not transparent at all, `.5` is 50% visible, and `0` is completely transparent.

Set the `opacity` of an element using `.opacity-{value}` utilities.

100%
75%
50%
25%

```
<div class="opacity-100">...</div>
<div class="opacity-75">...</div>
<div class="opacity-50">...</div>
<div class="opacity-25">...</div>
```

Utilities API

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

```
    "opacity": (
      property: opacity,
      values: (
        0: 0,
        25: .25,
        50: .5,
        75: .75,
        100: 1,
      )
    ),
```

Overflow

Use these shorthand utilities for quickly configuring how content overflows an element.

Adjust the `overflow` property on the fly with four default values and classes. These classes are not responsive by default.

This is an example of using `.overflow-auto` on an element with set width and height dimensions. By design, this content will vertically scroll.
This is an example of using `.overflow-hidden` on an element with set width and height dimensions.
This is an example of using `.overflow-visible` on an element with set width and height dimensions.
This is an example of using `.overflow-scroll` on an element with set width and height dimensions.

```
<div class="overflow-auto">...</div>
<div class="overflow-hidden">...</div>
<div class="overflow-visible">...</div>
<div class="overflow-scroll">...</div>
```

Position

Use these shorthand utilities for quickly configuring the position of an element.

On this page

Position values

Quick positioning classes are available, though they are not responsive.

```
<div class="position-static">...</div>
<div class="position-relative">...</div>
<div class="position-absolute">...</div>
<div class="position-fixed">...</div>
<div class="position-sticky">...</div>
```

Arrange elements

Arrange elements easily with the edge positioning utilities. The format is `{property}-{position}`.

Where *property* is one of:

- `top` - for the vertical `top` position
- `start` - for the horizontal `left` position (in LTR)
- `bottom` - for the vertical `bottom` position
- `end` - for the horizontal `right` position (in LTR)

Where *position* is one of:

- `0` - for `0` edge position
- `50` - for `50%` edge position
- `100` - for `100%` edge position

(You can add more position values by adding entries to the `$position-values` Sass map variable.)

html

```
<div class="position-relative">
  <div class="position-absolute top-0 start-0"></div>
  <div class="position-absolute top-0 end-0"></div>
  <div class="position-absolute top-50 start-50"></div>
  <div class="position-absolute bottom-50 end-50"></div>
  <div class="position-absolute bottom-0 start-0"></div>
  <div class="position-absolute bottom-0 end-0"></div>
</div>
```

Center elements

In addition, you can also center the elements with the transform utility class `.translate-middle`.

This class applies the transformations `translateX(-50%)` and `translateY(-50%)` to the element which, in combination with the edge positioning utilities, allows you to absolute center an element.

html

```
<div class="position-relative">
  <div class="position-absolute top-0 start-0 translate-middle"></div>
  <div class="position-absolute top-0 start-50 translate-middle"></div>
  <div class="position-absolute top-0 start-100 translate-middle"></div>
  <div class="position-absolute top-50 start-0 translate-middle"></div>
  <div class="position-absolute top-50 start-50 translate-middle"></div>
  <div class="position-absolute top-50 start-100 translate-middle"></div>
  <div class="position-absolute top-100 start-0 translate-middle"></div>
  <div class="position-absolute top-100 start-50 translate-middle"></div>
  <div class="position-absolute top-100 start-100 translate-middle"></div>
</div>
```

By adding `.translate-middle-x` or `.translate-middle-y` classes, elements can be positioned only in horizontal or vertical direction.

html

```
<div class="position-relative">
  <div class="position-absolute top-0 start-0"></div>
  <div class="position-absolute top-0 start-50 translate-middle-x"></div>
  <div class="position-absolute top-0 end-0"></div>
  <div class="position-absolute top-50 start-0 translate-middle-y"></div>
  <div class="position-absolute top-50 start-50 translate-middle"></div>
  <div class="position-absolute top-50 end-0 translate-middle-y"></div>
  <div class="position-absolute bottom-0 start-0"></div>
  <div class="position-absolute bottom-0 start-50 translate-middle-x"></div>
  <div class="position-absolute bottom-0 end-0"></div>
</div>
```

Examples

Here are some real life examples of these classes:

html

```
<button type="button" class="btn btn-primary position-relative">
  Mails <span class="position-absolute top-0 start-100 translate-middle badge
rounded-pill bg-secondary">+99 <span class="visually-hidden">unread
messages</span></span>
</button>

<button type="button" class="btn btn-dark position-relative">
```

```
  Marker <svg width="1em" height="1em" viewBox="0 0 16 16" class="position-
absolute top-100 start-50 translate-middle mt-1" fill="#212529"
xmlns="http://www.w3.org/2000/svg"><path d="M7.247 11.14L2.451 5.658C1.885 5.013
2.345 4 3.204 4h9.592a1 1 0 0 1 .753 1.659l-4.796 5.48a1 1 0 0 1-1.506
0z"/></svg>
</button>

<button type="button" class="btn btn-primary position-relative">
  Alerts <span class="position-absolute top-0 start-100 translate-middle badge
border border-light rounded-circle bg-danger p-2"><span class="visually-
hidden">unread messages</span></span>
</button>
```

You can use these classes with existing components to create new ones. Remember that you can extend its functionality by adding entries to the `$position-values` variable.

html

```
<div class="position-relative m-4">
  <div class="progress" style="height: 1px;">
    <div class="progress-bar" role="progressbar" style="width: 50%;" aria-
valuenow="50" aria-valuemin="0" aria-valuemax="100"></div>
  </div>
  <button type="button" class="position-absolute top-0 start-0 translate-middle
btn btn-sm btn-primary rounded-pill" style="width: 2rem;
height:2rem;">1</button>
  <button type="button" class="position-absolute top-0 start-50 translate-middle
btn btn-sm btn-primary rounded-pill" style="width: 2rem;
height:2rem;">2</button>
  <button type="button" class="position-absolute top-0 start-100 translate-
middle btn btn-sm btn-secondary rounded-pill" style="width: 2rem;
height:2rem;">3</button>
</div>
```

Sass

Maps

Default position utility values are declared in a Sass map, then used to generate our utilities.

```
$position-values: (
  0: 0,
  50: 50%,
  100: 100%
);
```

Utilities API

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

```
    "position": (
      property: position,
      values: static relative absolute fixed sticky
    ),
    "top": (
      property: top,
      values: $position-values
    ),
    "bottom": (
      property: bottom,
      values: $position-values
    ),
    "start": (
      property: left,
      class: start,
      values: $position-values
    ),
    "end": (
      property: right,
      class: end,
      values: $position-values
    ),
    "translate-middle": (
      property: transform,
      class: translate-middle,
      values: (
        null: translate(-50%, -50%),
        x: translateX(-50%),
        y: translateY(-50%),
      )
    ),
```

Shadows

Add or remove shadows to elements with box-shadow utilities.

On this page

Examples

While shadows on components are disabled by default in Bootstrap and can be enabled via `$enable-shadows`, you can also quickly add or remove a shadow with our `box-shadow` utility classes. Includes support for `.shadow-none` and three default sizes (which have associated variables to match).

No shadow
Small shadow
Regular shadow
Larger shadow

html

```
<div class="shadow-none p-3 mb-5 bg-light rounded">No shadow</div>
<div class="shadow-sm p-3 mb-5 bg-body rounded">Small shadow</div>
<div class="shadow p-3 mb-5 bg-body rounded">Regular shadow</div>
<div class="shadow-lg p-3 mb-5 bg-body rounded">Larger shadow</div>
```

Sass

Variables

```
$box-shadow:                  0 .5rem 1rem rgba($black, .15);
$box-shadow-sm:               0 .125rem .25rem rgba($black, .075);
$box-shadow-lg:               0 1rem 3rem rgba($black, .175);
$box-shadow-inset:            inset 0 1px 2px rgba($black, .075);
```

Utilities API

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

```
    "shadow": (
      property: box-shadow,
      class: shadow,
      values: (
        null: $box-shadow,
        sm: $box-shadow-sm,
        lg: $box-shadow-lg,
        none: none,
      )
    ),
```

Sizing

Easily make an element as wide or as tall with our width and height utilities.

On this page

Relative to the parent

Width and height utilities are generated from the utility API in `_utilities.scss`. Includes support for `25%`, `50%`, `75%`, `100%`, and `auto` by default. Modify those values as you need to generate different utilities here.

Width 25%
Width 50%
Width 75%
Width 100%
Width auto

html

```html
<div class="w-25 p-3" style="background-color: #eee;">Width 25%</div>
<div class="w-50 p-3" style="background-color: #eee;">Width 50%</div>
<div class="w-75 p-3" style="background-color: #eee;">Width 75%</div>
<div class="w-100 p-3" style="background-color: #eee;">Width 100%</div>
<div class="w-auto p-3" style="background-color: #eee;">Width auto</div>
```

Height 25%
Height 50%
Height 75%
Height 100%
Height auto

html

```html
<div style="height: 100px; background-color: rgba(255,0,0,0.1);">
  <div class="h-25 d-inline-block" style="width: 120px; background-color:
rgba(0,0,255,.1)">Height 25%</div>
  <div class="h-50 d-inline-block" style="width: 120px; background-color:
rgba(0,0,255,.1)">Height 50%</div>
  <div class="h-75 d-inline-block" style="width: 120px; background-color:
rgba(0,0,255,.1)">Height 75%</div>
  <div class="h-100 d-inline-block" style="width: 120px; background-color:
rgba(0,0,255,.1)">Height 100%</div>
  <div class="h-auto d-inline-block" style="width: 120px; background-color:
rgba(0,0,255,.1)">Height auto</div>
```

```
</div>
```

You can also use `max-width: 100%;` and `max-height: 100%;` utilities as needed.

html

```
<img src="..." class="mw-100" alt="...">
```

Max-height 100%

html

```
<div style="height: 100px; background-color: rgba(255,0,0,.1);">
  <div class="mh-100" style="width: 100px; height: 200px; background-color:
rgba(0,0,255,.1);">Max-height 100%</div>
</div>
```

Relative to the viewport

You can also use utilities to set the width and height relative to the viewport.

```
<div class="min-vw-100">Min-width 100vw</div>
<div class="min-vh-100">Min-height 100vh</div>
<div class="vw-100">Width 100vw</div>
<div class="vh-100">Height 100vh</div>
```

Sass

Utilities API

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

```
    "width": (
      property: width,
      class: w,
      values: (
        25: 25%,
        50: 50%,
        75: 75%,
        100: 100%,
        auto: auto
      )
    ),
    "max-width": (
      property: max-width,
      class: mw,
      values: (100: 100%)
    ),
    "viewport-width": (
      property: width,
      class: vw,
      values: (100: 100vw)
    ),
    "min-viewport-width": (
      property: min-width,
      class: min-vw,
      values: (100: 100vw)
    ),
    "height": (
      property: height,
      class: h,
      values: (
        25: 25%,
        50: 50%,
        75: 75%,
        100: 100%,
        auto: auto
      )
    ),
    "max-height": (
      property: max-height,
      class: mh,
      values: (100: 100%)
    ),
    "viewport-height": (
      property: height,
      class: vh,
      values: (100: 100vh)
    ),
    "min-viewport-height": (
      property: min-height,
      class: min-vh,
      values: (100: 100vh)
    ),
```

Spacing

Bootstrap includes a wide range of shorthand responsive margin, padding, and gap utility classes to modify an element's appearance.

On this page

Margin and padding

Assign responsive-friendly `margin` or `padding` values to an element or a subset of its sides with shorthand classes. Includes support for individual properties, all properties, and vertical and horizontal properties. Classes are built from a default Sass map ranging from `.25rem` to `3rem`.

Using the CSS Grid layout module? Consider using the gap utility instead.

Notation

Spacing utilities that apply to all breakpoints, from `xs` to `xxl`, have no breakpoint abbreviation in them. This is because those classes are applied from `min-width: 0` and up, and thus are not bound by a media query. The remaining breakpoints, however, do include a breakpoint abbreviation.

The classes are named using the format `{property}{sides}-{size}` for `xs` and `{property}{sides}-{breakpoint}-{size}` for `sm`, `md`, `lg`, `xl`, and `xxl`.

Where *property* is one of:

- `m` - for classes that set `margin`
- `p` - for classes that set `padding`

Where *sides* is one of:

- `t` - for classes that set `margin-top` or `padding-top`
- `b` - for classes that set `margin-bottom` or `padding-bottom`
- `s` - (start) for classes that set `margin-left` or `padding-left` in LTR, `margin-right` or `padding-right` in RTL
- `e` - (end) for classes that set `margin-right` or `padding-right` in LTR, `margin-left` or `padding-left` in RTL

- `x` - for classes that set both `*-left` and `*-right`
- `y` - for classes that set both `*-top` and `*-bottom`
- blank - for classes that set a `margin` or `padding` on all 4 sides of the element

Where *size* is one of:

- `0` - for classes that eliminate the `margin` or `padding` by setting it to `0`
- `1` - (by default) for classes that set the `margin` or `padding` to `$spacer * .25`
- `2` - (by default) for classes that set the `margin` or `padding` to `$spacer * .5`
- `3` - (by default) for classes that set the `margin` or `padding` to `$spacer`
- `4` - (by default) for classes that set the `margin` or `padding` to `$spacer * 1.5`
- `5` - (by default) for classes that set the `margin` or `padding` to `$spacer * 3`
- `auto` - for classes that set the `margin` to auto

(You can add more sizes by adding entries to the `$spacers` Sass map variable.)

Examples

Here are some representative examples of these classes:

```
.mt-0 {
  margin-top: 0 !important;
}

.ms-1 {
  margin-left: ($spacer * .25) !important;
}

.px-2 {
  padding-left: ($spacer * .5) !important;
  padding-right: ($spacer * .5) !important;
}

.p-3 {
  padding: $spacer !important;
}
```

Horizontal centering

Additionally, Bootstrap also includes an `.mx-auto` class for horizontally centering fixed-width block level content—that is, content that has `display: block` and a `width` set—by setting the horizontal margins to `auto`.

Centered element

```
<div class="mx-auto" style="width: 200px;">
  Centered element
</div>
```

Negative margin

In CSS, `margin` properties can utilize negative values (`padding` cannot). These negative margins are **disabled by default**, but can be enabled in Sass by setting `$enable-negative-margins: true`.

The syntax is nearly the same as the default, positive margin utilities, but with the addition of `n` before the requested size. Here's an example class that's the opposite of `.mt-1`:

```
.mt-n1 {
  margin-top: -0.25rem !important;
}
```

Gap

When using `display: grid`, you can make use of `gap` utilities on the parent grid container. This can save on having to add margin utilities to individual grid items (children of a `display: grid` container). Gap utilities are responsive by default, and are generated via our utilities API, based on the `$spacers` Sass map.

Grid item 1
Grid item 2
Grid item 3

html

```
<div class="d-grid gap-3">
  <div class="p-2 bg-light border">Grid item 1</div>
  <div class="p-2 bg-light border">Grid item 2</div>
  <div class="p-2 bg-light border">Grid item 3</div>
</div>
```

Support includes responsive options for all of Bootstrap's grid breakpoints, as well as six sizes from the `$spacers` map (`0`–`5`). There is no `.gap-auto` utility class as it's effectively the same as `.gap-0`.

Sass

Maps

Spacing utilities are declared via Sass map and then generated with our utilities API.

```
$spacer: 1rem;
$spacers: (
  0: 0,
  1: $spacer * .25,
  2: $spacer * .5,
  3: $spacer,
  4: $spacer * 1.5,
  5: $spacer * 3,
);
```

Utilities API

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

```
    "margin": (
      responsive: true,
      property: margin,
      class: m,
      values: map-merge($spacers, (auto: auto))
    ),
    "margin-x": (
      responsive: true,
      property: margin-right margin-left,
      class: mx,
      values: map-merge($spacers, (auto: auto))
    ),
    "margin-y": (
      responsive: true,
      property: margin-top margin-bottom,
      class: my,
      values: map-merge($spacers, (auto: auto))
    ),
    "margin-top": (
      responsive: true,
      property: margin-top,
      class: mt,
      values: map-merge($spacers, (auto: auto))
    ),
    "margin-end": (
      responsive: true,
      property: margin-right,
      class: me,
      values: map-merge($spacers, (auto: auto))
    ),
    "margin-bottom": (
      responsive: true,
      property: margin-bottom,
      class: mb,
      values: map-merge($spacers, (auto: auto))
    ),
    "margin-start": (
      responsive: true,
      property: margin-left,
      class: ms,
      values: map-merge($spacers, (auto: auto))
    ),
    // Negative margin utilities
    "negative-margin": (
      responsive: true,
      property: margin,
      class: m,
      values: $negative-spacers
    ),
    "negative-margin-x": (
      responsive: true,
      property: margin-right margin-left,
      class: mx,
      values: $negative-spacers
    ),
    "negative-margin-y": (
      responsive: true,
      property: margin-top margin-bottom,
      class: my,
      values: $negative-spacers
    ),
    "negative-margin-top": (
      responsive: true,
      property: margin-top,
      class: mt,
      values: $negative-spacers
    ),
    "negative-margin-end": (
      responsive: true,
      property: margin-right,
      class: me,
      values: $negative-spacers
    ),
    "negative-margin-bottom": (
      responsive: true,
      property: margin-bottom,
      class: mb,
      values: $negative-spacers
    ),
    "negative-margin-start": (
      responsive: true,
      property: margin-left,
      class: ms,
      values: $negative-spacers
    ),
    // Padding utilities
    "padding": (
      responsive: true,
      property: padding,
      class: p,
      values: $spacers
    ),
    "padding-x": (
      responsive: true,
      property: padding-right padding-left,
      class: px,
      values: $spacers
    ),
    "padding-y": (
      responsive: true,
      property: padding-top padding-bottom,
      class: py,
      values: $spacers
    ),
    "padding-top": (
      responsive: true,
      property: padding-top,
      class: pt,
      values: $spacers
    ),
    "padding-end": (
      responsive: true,
      property: padding-right,
      class: pe,
      values: $spacers
    ),
    "padding-bottom": (
      responsive: true,
      property: padding-bottom,
      class: pb,
      values: $spacers
    ),
    "padding-start": (
      responsive: true,
      property: padding-left,
      class: ps,
      values: $spacers
    ),
    // Gap utility
    "gap": (
      responsive: true,
      property: gap,
      class: gap,
      values: $spacers
    ),
```

Text

Documentation and examples for common text utilities to control alignment, wrapping, weight, and more.

On this page

Text alignment

Easily realign text to components with text alignment classes. For start, end, and center alignment, responsive classes are available that use the same viewport width breakpoints as the grid system.

Start aligned text on all viewport sizes.

Center aligned text on all viewport sizes.

End aligned text on all viewport sizes.

Start aligned text on viewports sized SM (small) or wider.

Start aligned text on viewports sized MD (medium) or wider.

Start aligned text on viewports sized LG (large) or wider.

Start aligned text on viewports sized XL (extra-large) or wider.

html

```html
<p class="text-start">Start aligned text on all viewport sizes.</p>
<p class="text-center">Center aligned text on all viewport sizes.</p>
```

```
<p class="text-end">End aligned text on all viewport sizes.</p>

<p class="text-sm-start">Start aligned text on viewports sized SM (small) or
wider.</p>
<p class="text-md-start">Start aligned text on viewports sized MD (medium) or
wider.</p>
<p class="text-lg-start">Start aligned text on viewports sized LG (large) or
wider.</p>
<p class="text-xl-start">Start aligned text on viewports sized XL (extra-large)
or wider.</p>
```

Note that we don't provide utility classes for justified text. While, aesthetically, justified text might look more appealing, it does make word-spacing more random and therefore harder to read.

Text wrapping and overflow

Wrap text with a `.text-wrap` class.

This text should wrap.

html

```
<div class="badge bg-primary text-wrap" style="width: 6rem;">
  This text should wrap.
</div>
```

Prevent text from wrapping with a `.text-nowrap` class.

This text should overflow the parent.

html

```
<div class="text-nowrap bg-light border" style="width: 8rem;">
  This text should overflow the parent.
</div>
```

Word break

Prevent long strings of text from breaking your components' layout by using `.text-break` to set `word-wrap: break-word` and `word-break: break-word`. We use `word-wrap` instead of the more common `overflow-wrap` for wider browser support, and add the deprecated `word-break: break-word` to avoid issues with flex containers.

mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm

html

```
<p class="text-
break">mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmm</p>
```

Note that breaking words isn't possible in Arabic, which is the most used RTL language. Therefore `.text-break` is removed from our RTL compiled CSS.

Text transform

Transform text in components with text capitalization classes.

Lowercased text.

Uppercased text.

CapiTaliZed text.

html

```
<p class="text-lowercase">Lowercased text.</p>
<p class="text-uppercase">Uppercased text.</p>
<p class="text-capitalize">CapiTaliZed text.</p>
```

Note how `.text-capitalize` only changes the first letter of each word, leaving the case of any other letters unaffected.

Font size

Quickly change the `font-size` of text. While our heading classes (e.g., `.h1`–`.h6`) apply `font-size`, `font-weight`, and `line-height`, these utilities *only* apply `font-size`. Sizing for these utilities matches HTML's heading elements, so as the number increases, their size decreases.

.fs-1 text

.fs-2 text

.fs-3 text

.fs-4 text

.fs-5 text

.fs-6 text

html

```
<p class="fs-1">.fs-1 text</p>
<p class="fs-2">.fs-2 text</p>
<p class="fs-3">.fs-3 text</p>
<p class="fs-4">.fs-4 text</p>
<p class="fs-5">.fs-5 text</p>
<p class="fs-6">.fs-6 text</p>
```

Customize your available `font-size`s by modifying the `$font-sizes` Sass map.

Font weight and italics

Quickly change the `font-weight` or `font-style` of text with these utilities. `font-style` utilities are abbreviated as `.fst-*` and `font-weight` utilities are abbreviated as `.fw-*`.

Bold text.

Bolder weight text (relative to the parent element).

Semibold weight text.

Normal weight text.

Light weight text.

Lighter weight text (relative to the parent element).

Italic text.

Text with normal font style

html

```html
<p class="fw-bold">Bold text.</p>
<p class="fw-bolder">Bolder weight text (relative to the parent element).</p>
<p class="fw-semibold">Semibold weight text.</p>
<p class="fw-normal">Normal weight text.</p>
<p class="fw-light">Light weight text.</p>
<p class="fw-lighter">Lighter weight text (relative to the parent element).</p>
<p class="fst-italic">Italic text.</p>
<p class="fst-normal">Text with normal font style</p>
```

Line height

Change the line height with `.lh-*` utilities.

This is a long paragraph written to show how the line-height of an element is affected by our utilities. Classes are applied to the element itself or sometimes the parent element. These classes can be customized as needed with our utility API.

This is a long paragraph written to show how the line-height of an element is affected by our utilities. Classes are applied to the element itself or sometimes the parent element. These classes can be customized as needed with our utility API.

This is a long paragraph written to show how the line-height of an element is affected by our utilities. Classes are applied to the element itself or sometimes the parent element. These classes can be customized as needed with our utility API.

This is a long paragraph written to show how the line-height of an element is affected by our utilities. Classes are applied to the element itself or sometimes the parent element. These classes can be customized as needed with our utility API.

html

```html
<p class="lh-1">This is a long paragraph written to show how the line-height of
an element is affected by our utilities. Classes are applied to the element
itself or sometimes the parent element. These classes can be customized as
needed with our utility API.</p>
<p class="lh-sm">This is a long paragraph written to show how the line-height of
an element is affected by our utilities. Classes are applied to the element
itself or sometimes the parent element. These classes can be customized as
needed with our utility API.</p>
<p class="lh-base">This is a long paragraph written to show how the line-height
of an element is affected by our utilities. Classes are applied to the element
itself or sometimes the parent element. These classes can be customized as
needed with our utility API.</p>
<p class="lh-lg">This is a long paragraph written to show how the line-height of
an element is affected by our utilities. Classes are applied to the element
itself or sometimes the parent element. These classes can be customized as
needed with our utility API.</p>
```

Monospace

Change a selection to our monospace font stack with `.font-monospace`.

This is in monospace

html

```
<p class="font-monospace">This is in monospace</p>
```

Reset color

Reset a text or link's color with `.text-reset`, so that it inherits the color from its parent.

Muted text with a reset link.

html

```
<p class="text-muted">
  Muted text with a <a href="#" class="text-reset">reset link</a>.
</p>
```

Text decoration

Decorate text in components with text decoration classes.

This text has a line underneath it.

This text has a line going through it.

This link has its text decoration removed

html

```
<p class="text-decoration-underline">This text has a line underneath it.</p>
<p class="text-decoration-line-through">This text has a line going through
it.</p>
<a href="#" class="text-decoration-none">This link has its text decoration
removed</a>
```

Sass

Variables

```
// stylelint-disable value-keyword-case
$font-family-sans-serif:      system-ui, -apple-system, "Segoe UI", Roboto,
"Helvetica Neue", "Noto Sans", "Liberation Sans", Arial, sans-serif, "Apple
Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";
$font-family-monospace:       SFMono-Regular, Menlo, Monaco, Consolas,
"Liberation Mono", "Courier New", monospace;
// stylelint-enable value-keyword-case
$font-family-base:            var(--#{$prefix}font-sans-serif);
$font-family-code:            var(--#{$prefix}font-monospace);

// $font-size-root affects the value of `rem`, which is used for as well font
sizes, paddings, and margins
// $font-size-base affects the font size of the body text
$font-size-root:              null;
```

```
$font-size-base:              1rem; // Assumes the browser default, typically
`16px`
$font-size-sm:                $font-size-base * .875;
$font-size-lg:                $font-size-base * 1.25;

$font-weight-lighter:         lighter;
$font-weight-light:           300;
$font-weight-normal:          400;
$font-weight-semibold:        600;
$font-weight-bold:            700;
$font-weight-bolder:          bolder;

$font-weight-base:            $font-weight-normal;

$line-height-base:            1.5;
$line-height-sm:              1.25;
$line-height-lg:              2;

$h1-font-size:                $font-size-base * 2.5;
$h2-font-size:                $font-size-base * 2;
$h3-font-size:                $font-size-base * 1.75;
$h4-font-size:                $font-size-base * 1.5;
$h5-font-size:                $font-size-base * 1.25;
$h6-font-size:                $font-size-base;
```

Maps

Font-size utilities are generated from this map, in combination with our utilities API.

```
$font-sizes: (
  1: $h1-font-size,
  2: $h2-font-size,
  3: $h3-font-size,
  4: $h4-font-size,
  5: $h5-font-size,
  6: $h6-font-size
);
```

Utilities API

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

```
    "font-family": (
      property: font-family,
      class: font,
      values: (monospace: var(--#{$prefix}font-monospace))
    ),
    "font-size": (
      rfs: true,
      property: font-size,
      class: fs,
      values: $font-sizes
    ),
    "font-style": (
      property: font-style,
      class: fst,
      values: italic normal
    ),
    "font-weight": (
      property: font-weight,
      class: fw,
      values: (
        light: $font-weight-light,
        lighter: $font-weight-lighter,
        normal: $font-weight-normal,
        bold: $font-weight-bold,
        semibold: $font-weight-semibold,
        bolder: $font-weight-bolder
      )
    ),
    "line-height": (
      property: line-height,
      class: lh,
      values: (
        1: 1,
        sm: $line-height-sm,
        base: $line-height-base,
        lg: $line-height-lg,
      )
    ),
    "text-align": (
      responsive: true,
      property: text-align,
      class: text,
      values: (
        start: left,
        end: right,
        center: center,
      )
    ),
    "text-decoration": (
      property: text-decoration,
      values: none underline line-through
    ),
    "text-transform": (
      property: text-transform,
      class: text,
      values: lowercase uppercase capitalize
    ),
    "white-space": (
      property: white-space,
      class: text,
      values: (
        wrap: normal,
        nowrap: nowrap,
      )
    ),
    "word-wrap": (
      property: word-wrap word-break,
      class: text,
      values: (break: break-word),
      rtl: false
    ),
```

Vertical alignment

Easily change the vertical alignment of inline, inline-block, inline-table, and table cell elements.

Change the alignment of elements with the `vertical-alignment` utilities. Please note that vertical-align only affects inline, inline-block, inline-table, and table cell elements.

Choose from `.align-baseline`, `.align-top`, `.align-middle`, `.align-bottom`, `.align-text-bottom`, and `.align-text-top` as needed.

To vertically center non-inline content (like `<div>`s and more), use our flex box utilities.

With inline elements:

baseline top middle bottom text-top text-bottom

html

```
<span class="align-baseline">baseline</span>
<span class="align-top">top</span>
<span class="align-middle">middle</span>
<span class="align-bottom">bottom</span>
<span class="align-text-top">text-top</span>
<span class="align-text-bottom">text-bottom</span>
```

With table cells:

baseline	top	middle	bottom	text-top	text-bottom

html

```
<table style="height: 100px;">
  <tbody>
    <tr>
      <td class="align-baseline">baseline</td>
      <td class="align-top">top</td>
      <td class="align-middle">middle</td>
      <td class="align-bottom">bottom</td>
      <td class="align-text-top">text-top</td>
      <td class="align-text-bottom">text-bottom</td>
    </tr>
  </tbody>
</table>
```

Sass

Utilities API

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

```
    "align": (
      property: vertical-align,
      class: align,
      values: baseline top middle bottom text-bottom text-top
    ),
```

Visibility

Control the visibility of elements, without modifying their display, with visibility utilities.

Set the `visibility` of elements with our visibility utilities. These utility classes do not modify the `display` value at all and do not affect layout – `.invisible` elements still take up space in the page.

Elements with the `.invisible` class will be hidden *both* visually and for assistive technology/screen reader users.

Apply `.visible` or `.invisible` as needed.

```
<div class="visible">...</div>
<div class="invisible">...</div>

// Class
.visible {
  visibility: visible !important;
}
.invisible {
  visibility: hidden !important;
}
```

Icons

Guidance and suggestions for using external icon libraries with Bootstrap.

While Bootstrap doesn’t include an icon set by default, we do have our own comprehensive icon library called Bootstrap Icons. Feel free to use them or any other icon set in your project. We’ve included details for Bootstrap Icons and other preferred icon sets below.

While most icon sets include multiple file formats, we prefer SVG implementations for their improved accessibility and vector support.

Bootstrap Icons

Bootstrap Icons is a growing library of SVG icons that are designed by @mdo and maintained by the Bootstrap Team. The beginnings of this icon set come from Bootstrap’s very own components—our forms, carousels, and more. Bootstrap has very few icon needs out of the box, so we didn’t need much. However, once we got going, we couldn’t stop making more.

Oh, and did we mention they’re completely open source? Licensed under MIT, just like Bootstrap, our icon set is available to everyone.

Learn more about Bootstrap Icons, including how to install them and recommended usage.

Alternatives

We’ve tested and used these icon sets ourselves as preferred alternatives to Bootstrap Icons.

- Font Awesome
- Feather
- Octicons

More options

While we haven’t tried these out ourselves, they do look promising and provide multiple formats, including SVG.

- Bytesize
- CoreUI Icons
- Google Material icons
- Ionicons
- Dripicons
- Ikons
- Icons8
- icofont
- Tabler Icons

Migrating to v5

Track and review changes to the Bootstrap source files, documentation, and components to help you migrate from v4 to v5.

On this page

- Helpers
- JavaScript

v5.2.0

Refreshed design

Bootstrap v5.2.0 features a subtle design update for a handful of components and properties across the project, **most notably through refined `border-radius` values on buttons and form controls**. Our documentation also has been updated with a new homepage, simpler docs layout that no longer collapses sections of the sidebar, and more prominent examples of Bootstrap Icons.

More CSS variables

We've updated all our components to use CSS variables. While Sass still underpins everything, each component has been updated to include CSS variables on the component base classes (e.g., `.btn`), allowing for more real-time customization of Bootstrap. In subsequent releases, we'll continue to expand our use of CSS variables into our layout, forms, helpers, and utilities. Read more about CSS variables in each component on their respective documentation pages.

Our CSS variable usage will be somewhat incomplete until Bootstrap 6. While we'd love to fully implement these across the board, they do run the risk of causing breaking changes. For example, setting `$alert-border-width: var(--bs-border-width)` in our source code breaks potential Sass in your own code if you were doing `$alert-border-width * 2` for some reason.

As such, wherever possible, we will continue to push towards more CSS variables, but please recognize our implementation may be slightly limited in v5.

New `_maps.scss`

Bootstrap v5.2.0 introduced a new Sass file with `_maps.scss`. It pulls out several Sass maps from `_variables.scss` to fix an issue where updates to an original map were not applied to secondary maps that extend them. For example, updates to `$theme-colors` were not being applied to other theme maps that relied on `$theme-colors`, breaking key customization workflows. In short, Sass has a limitation where once a default variable or map has been *used*, it cannot be updated. *There's a similar shortcoming with CSS variables when they're used to compose other CSS variables.*

This is why variable customizations in Bootstrap have to come after `@import "functions"`, but before `@import "variables"` and the rest of our import stack. The same applies to Sass maps—you must override the defaults before they get used. The following maps have been moved to the new `_maps.scss`:

- `$theme-colors-rgb`
- `$utilities-colors`
- `$utilities-text`

- `$utilities-text-colors`
- `$utilities-bg`
- `$utilities-bg-colors`
- `$negative-spacers`
- `$gutters`

Your custom Bootstrap CSS builds should now look something like this with a separate maps import.

```
  // Functions come first
  @import "functions";

  // Optional variable overrides here
+ $custom-color: #df711b;
+ $custom-theme-colors: (
+   "custom": $custom-color
+ );

  // Variables come next
  @import "variables";

+ // Optional Sass map overrides here
+ $theme-colors: map-merge($theme-colors, $custom-theme-colors);
+
+ // Followed by our default maps
+ @import "maps";
+
  // Rest of our imports
  @import "mixins";
  @import "utilities";
  @import "root";
  @import "reboot";
  // etc
```

New utilities

- Expanded `font-weight` utilities to include `.fw-semibold` for semibold fonts.
- Expanded `border-radius` utilities to include two new sizes, `.rounded-4` and `.rounded-5`, for more options.

Additional changes

- **Introduced new `$enable-container-classes` option. —** Now when opting into the experimental CSS Grid layout, `.container-*` classes will still be compiled, unless this option is set to `false`. Containers also now keep their gutter values.

- **Offcanvas component now has responsive variations.** The original `.offcanvas` class remains unchanged—it hides content across all viewports. To make it responsive, change that `.offcanvas` class to any `.offcanvas-{sm|md|lg|xl|xxl}` class.

- **Thicker table dividers are now opt-in. —** We've removed the thicker and more difficult to override border between table groups and moved it to an optional class you can apply, `.table-group-divider`. See the table docs for an example.

- **Scrollspy has been rewritten to use the Intersection Observer API**, which means you no longer need relative parent wrappers, deprecates `offset` config, and more. Look for your Scrollspy implementations to be more accurate and consistent in their nav highlighting.
- **Popovers and tooltips now use CSS variables.** Some CSS variables have been updated from their Sass counterparts to reduce the number of variables. As a result, three variables have been deprecated in this release: `$popover-arrow-color`, `$popover-arrow-outer-color`, and `$tooltip-arrow-color`.
- **Added new `.text-bg-{color}` helpers.** Instead of setting individual `.text-*` and `.bg-*` utilities, you can now use the `.text-bg-*` helpers to set a `background-color` with contrasting foreground `color`.
- Added `.form-check-reverse` modifier to flip the order of labels and associated checkboxes/radios.
- Added striped columns support to tables via the new `.table-striped-columns` class.

For a complete list of changes, see the v5.2.0 project on GitHub.

v5.1.0

- **Added experimental support for CSS Grid layout.** — This is a work in progress, and is not yet ready for production use, but you can opt into the new feature via Sass. To enable it, disable the default grid, by setting `$enable-grid-classes: false` and enable the CSS Grid by setting `$enable-cssgrid: true`.
- **Updated navbars to support offcanvas.** — Add offcanvas drawers in any navbar with the responsive `.navbar-expand-*` classes and some offcanvas markup.
- **Added new placeholder component.** — Our newest component, a way to provide temporary blocks in lieu of real content to help indicate that something is still loading in your site or app.
- **Collapse plugin now supports horizontal collapsing.** — Add `.collapse-horizontal` to your `.collapse` to collapse the `width` instead of the `height`. Avoid browser repainting by setting a `min-height` or `height`.
- **Added new stack and vertical rule helpers.** — Quickly apply multiple flexbox properties to quickly create custom layouts with stacks. Choose from horizontal (`.hstack`) and vertical (`.vstack`) stacks. Add vertical dividers similar to `<hr>` elements with the new `.vr` helpers.
- **Added new global `:root` CSS variables.** — Added several new CSS variables to the `:root` level for controlling `<body>` styles. More are in the works, including across our utilities and components, but for now read up CSS variables in the Customize section.
- **Overhauled color and background utilities to use CSS variables, and added new text opacity and background opacity utilities.** — `.text-*` and `.bg-*` utilities are now built

with CSS variables and `rgba()` color values, allowing you to easily customize any utility with new opacity utilities.

- **Added new snippet examples based to show how to customize our components. —** Pull ready to use customized components and other common design patterns with our new Snippets examples. Includes footers, dropdowns, list groups, and modals.
- **Removed unused positioning styles from popovers and tooltips** as these are handled solely by Popper. `$tooltip-margin` has been deprecated and set to `null` in the process.

Want more information? Read the v5.1.0 blog post.

Hey there! Changes to our first major release of Bootstrap 5, v5.0.0, are documented below. They don't reflect the additional changes shown above.

Dependencies

- Dropped jQuery.
- Upgraded from Popper v1.x to Popper v2.x.
- Replaced Libsass with Dart Sass as our Sass compiler given Libsass was deprecated.
- Migrated from Jekyll to Hugo for building our documentation

Browser support

- Dropped Internet Explorer 10 and 11
- Dropped Microsoft Edge < 16 (Legacy Edge)
- Dropped Firefox < 60
- Dropped Safari < 12
- Dropped iOS Safari < 12
- Dropped Chrome < 60

Documentation changes

- Redesigned homepage, docs layout, and footer.
- Added new Parcel guide.
- Added new Customize section, replacing v4's Theming page, with new details on Sass, global configuration options, color schemes, CSS variables, and more.
- Reorganized all form documentation into new Forms section, breaking apart the content into more focused pages.
- Similarly, updated the Layout section, to flesh out grid content more clearly.
- Renamed "Navs" component page to "Navs & Tabs".
- Renamed "Checks" page to "Checks & radios".
- Redesigned the navbar and added a new subnav to make it easier to get around our sites and docs versions.
- Added new keyboard shortcut for the search field: `Ctrl + /`.

Sass

- We’ve ditched the default Sass map merges to make it easier to remove redundant values. Keep in mind you now have to define all values in the Sass maps like `$theme-colors`. Check out how to deal with [Sass maps].
- Breaking Renamed `color-yiq()` function and related variables to `color-contrast()` as it’s no longer related to YIQ color space. [See #30168.]
  - `$yiq-contrasted-threshold` is renamed to `$min-contrast-ratio`.
  - `$yiq-text-dark` and `$yiq-text-light` are respectively renamed to `$color-contrast-dark` and `$color-contrast-light`.
- Breaking Media query mixins parameters have changed for a more logical approach.
  - `media-breakpoint-down()` uses the breakpoint itself instead of the next breakpoint (e.g., `media-breakpoint-down(lg)` instead of `media-breakpoint-down(md)` targets viewports smaller than `lg`).
  - Similarly, the second parameter in `media-breakpoint-between()` also uses the breakpoint itself instead of the next breakpoint (e.g., `media-between(sm, lg)` instead of `media-breakpoint-between(sm, md)` targets viewports between `sm` and `lg`).
- Breaking Removed print styles and `$enable-print-styles` variable. Print display classes are still around. [See #28339].
- Breaking Dropped `color()`, `theme-color()`, and `gray()` functions in favor of variables. [See #29083].
- Breaking Renamed `theme-color-level()` function to `color-level()` and now accepts any color you want instead of only `$theme-color` colors. [See #29083] **Watch out:** `color-level()` was later on dropped in `v5.0.0-alpha3`.
- Breaking Renamed `$enable-prefers-reduced-motion-media-query` and `$enable-pointer-cursor-for-buttons` to `$enable-reduced-motion` and `$enable-button-pointers` for brevity.
- Breaking Removed the `bg-gradient-variant()` mixin. Use the `.bg-gradient` class to add gradients to elements instead of the generated `.bg-gradient-*` classes.
- Breaking **Removed previously deprecated mixins:**
  - `hover`, `hover-focus`, `plain-hover-focus`, and `hover-focus-active`
  - `float()`
  - `form-control-mixin()`
  - `nav-divider()`
  - `retina-img()`
  - `text-hide()` (also dropped the associated utility class, `.text-hide`)
  - `visibility()`
  - `form-control-focus()`

- Breaking Renamed `scale-color()` function to `shift-color()` to avoid collision with Sass's own color scaling function.
- `box-shadow` mixins now allow `null` values and drop `none` from multiple arguments. See #30394.
- The `border-radius()` mixin now has a default value.

Color system

- The color system which worked with `color-level()` and `$theme-color-interval` was removed in favor of a new color system. All `lighten()` and `darken()` functions in our codebase are replaced by `tint-color()` and `shade-color()`. These functions will mix the color with either white or black instead of changing its lightness by a fixed amount. The `shift-color()` will either tint or shade a color depending on whether its weight parameter is positive or negative. See #30622 for more details.
- Added new tints and shades for every color, providing nine separate colors for each base color, as new Sass variables.
- Improved color contrast. Bumped color contrast ratio from 3:1 to 4.5:1 and updated blue, green, cyan, and pink colors to ensure WCAG 2.1 AA contrast. Also changed our color contrast color from `$gray-900` to `$black`.
- To support our color system, we've added new custom `tint-color()` and `shade-color()` functions to mix our colors appropriately.

Grid updates

- **New breakpoint!** Added new `xxl` breakpoint for `1400px` and up. No changes to all other breakpoints.
- **Improved gutters.** Gutters are now set in rems, and are narrower than v4 (`1.5rem`, or about `24px`, down from `30px`). This aligns our grid system's gutters with our spacing utilities.
    - Added new gutter class (`.g-*`, `.gx-*`, and `.gy-*`) to control horizontal/vertical gutters, horizontal gutters, and vertical gutters.
    - Breaking Renamed `.no-gutters` to `.g-0` to match new gutter utilities.
- Columns no longer have `position: relative` applied, so you may have to add `.position-relative` to some elements to restore that behavior.
- Breaking Dropped several `.order-*` classes that often went unused. We now only provide `.order-1` to `.order-5` out of the box.
- Breaking Dropped the `.media` component as it can be easily replicated with utilities. See #28265 and the flex utilities page for an example.

- Breaking `bootstrap-grid.css` now only applies `box-sizing: border-box` to the column instead of resetting the global box-sizing. This way, our grid styles can be used in more places without interference.
- `$enable-grid-classes` no longer disables the generation of container classes anymore. See #29146.
- Updated the `make-col` mixin to default to equal columns without a specified size.

Content, Reboot, etc

- **RFS is now enabled by default.** Headings using the `font-size()` mixin will automatically adjust their `font-size` to scale with the viewport. *This feature was previously opt-in with v4.*
- Breaking Overhauled our display typography to replace our `$display-*` variables and with a `$display-font-sizes` Sass map. Also removed the individual `$display-*-weight` variables for a single `$display-font-weight` and adjusted `font-size`s.
- Added two new `.display-*` heading sizes, `.display-5` and `.display-6`.
- **Links are underlined by default** (not just on hover), unless they're part of specific components.
- **Redesigned tables** to refresh their styles and rebuild them with CSS variables for more control over styling.
- Breaking Nested tables do not inherit styles anymore.
- Breaking `.thead-light` and `.thead-dark` are dropped in favor of the `.table-*` variant classes which can be used for all table elements (`thead`, `tbody`, `tfoot`, `tr`, `th` and `td`).
- Breaking The `table-row-variant()` mixin is renamed to `table-variant()` and accepts only 2 parameters: `$color` (color name) and `$value` (color code). The border color and accent colors are automatically calculated based on the table factor variables.
- Split table cell padding variables into `-y` and `-x`.
- Breaking Dropped `.pre-scrollable` class. See #29135
- Breaking `.text-*` utilities do not add hover and focus states to links anymore. `.link-*` helper classes can be used instead. See #29267
- Breaking Dropped `.text-justify` class. See #29793
- Breaking `<hr>` elements now use `height` instead of `border` to better support the `size` attribute. This also enables use of padding utilities to create thicker dividers (e.g., `<hr class="py-1">`).
- Reset default horizontal `padding-left` on `<ul>` and `<ol>` elements from browser default `40px` to `2rem`.

- Added `$enable-smooth-scroll`, which applies `scroll-behavior: smooth` globally—except for users asking for reduced motion through `prefers-reduced-motion` media query. See #31877

RTL

- Horizontal direction specific variables, utilities, and mixins have all been renamed to use logical properties like those found in flexbox layouts—e.g., `start` and `end` in lieu of `left` and `right`.

Forms

- **Added new floating forms!** We've promoted the Floating labels example to fully supported form components. See the new Floating labels page.
- Breaking **Consolidated native and custom form elements.** Checkboxes, radios, selects, and other inputs that had native and custom classes in v4 have been consolidated. Now nearly all our form elements are entirely custom, most without the need for custom HTML.
  - `.custom-control.custom-checkbox` is now `.form-check`.
  - `.custom-control.custom-custom-radio` is now `.form-check`.
  - `.custom-control.custom-switch` is now `.form-check.form-switch`.
  - `.custom-select` is now `.form-select`.
  - `.custom-file` and `.form-file` have been replaced by custom styles on top of `.form-control`.
  - `.custom-range` is now `.form-range`.
  - Dropped native `.form-control-file` and `.form-control-range`.
- Breaking Dropped `.input-group-append` and `.input-group-prepend`. You can now just add buttons and `.input-group-text` as direct children of the input groups.
- The longstanding Missing border radius on input group with validation feedback bug is finally fixed by adding an additional `.has-validation` class to input groups with validation.
- Breaking **Dropped form-specific layout classes for our grid system.** Use our grid and utilities instead of `.form-group`, `.form-row`, or `.form-inline`.
- Breaking Form labels now require `.form-label`.
- Breaking `.form-text` no longer sets `display`, allowing you to create inline or block help text as you wish just by changing the HTML element.
- Form controls no longer used fixed `height` when possible, instead deferring to `min-height` to improve customization and compatibility with other components.
- Validation icons are no longer applied to `<select>`s with `multiple`.
- Rearranged source Sass files under `scss/forms/`, including input group styles.

Components

- Unified `padding` values for alerts, breadcrumbs, cards, dropdowns, list groups, modals, popovers, and tooltips to be based on our `$spacer` variable. See #30564.

Accordion

- Added new accordion component.

Alerts

- Alerts now have examples with icons.
- Removed custom styles for `<hr>`s in each alert since they already use `currentColor`.

Badges

- Breaking Dropped all `.badge-*` color classes for background utilities (e.g., use `.bg-primary` instead of `.badge-primary`).
- Breaking Dropped `.badge-pill`—use the `.rounded-pill` utility instead.
- Breaking Removed hover and focus styles for `<a>` and `<button>` elements.
- Increased default padding for badges from `.25em`/`.5em` to `.35em`/`.65em`.

Breadcrumbs

- Simplified the default appearance of breadcrumbs by removing `padding`, `background-color`, and `border-radius`.
- Added new CSS custom property `--bs-breadcrumb-divider` for easy customization without needing to recompile CSS.

Buttons

- Breaking **Toggle buttons, with checkboxes or radios, no longer require JavaScript and have new markup.** We no longer require a wrapping element, add `.btn-check` to the `<input>`, and pair it with any `.btn` classes on the `<label>`. See #30650. *The docs for this has moved from our Buttons page to the new Forms section.*
- Breaking **Dropped `.btn-block` for utilities.** Instead of using `.btn-block` on the `.btn`, wrap your buttons with `.d-grid` and a `.gap-*` utility to space them as needed. Switch to responsive classes for even more control over them. Read the docs for some examples.
- Updated our `button-variant()` and `button-outline-variant()` mixins to support additional parameters.
- Updated buttons to ensure increased contrast on hover and active states.

- Disabled buttons now have `pointer-events: none;`.

Card

- Breaking Dropped `.card-deck` in favor of our grid. Wrap your cards in column classes and add a parent `.row-cols-*` container to recreate card decks (but with more control over responsive alignment).
- Breaking Dropped `.card-columns` in favor of Masonry. See #28922.
- Breaking Replaced the `.card` based accordion with a new Accordion component.

Carousel

- Added new `.carousel-dark` variant for dark text, controls, and indicators (great for lighter backgrounds).
- Replaced chevron icons for carousel controls with new SVGs from Bootstrap Icons.

Close button

- Breaking Renamed `.close` to `.btn-close` for a less generic name.
- Close buttons now use a `background-image` (embedded SVG) instead of a `&times;` in the HTML, allowing for easier customization without the need to touch your markup.
- Added new `.btn-close-white` variant that uses `filter: invert(1)` to enable higher contrast dismiss icons against darker backgrounds.

Collapse

- Removed scroll anchoring for accordions.

Dropdowns

- Added new `.dropdown-menu-dark` variant and associated variables for on-demand dark dropdowns.
- Added new variable for `$dropdown-padding-x`.
- Darkened the dropdown divider for improved contrast.
- Breaking All the events for the dropdown are now triggered on the dropdown toggle button and then bubbled up to the parent element.
- Dropdown menus now have a `data-bs-popper="static"` attribute set when the positioning of the dropdown is static, or dropdown is in the navbar. This is added by our JavaScript and helps us use custom position styles without interfering with Popper's positioning.
- Breaking Dropped `flip` option for dropdown plugin in favor of native Popper configuration. You can now disable the flipping behavior by passing an empty array for `fallbackPlacements` option in flip modifier.

- Dropdown menus can now be clickable with a new `autoClose` option to handle the auto close behavior. You can use this option to accept the click inside or outside the dropdown menu to make it interactive.
- Dropdowns now support `.dropdown-item`s wrapped in `<li>`s.

Jumbotron

- Breaking Dropped the jumbotron component as it can be replicated with utilities. See our new Jumbotron example for a demo.

List group

- Added new `.list-group-numbered` modifier to list groups.

Navs and tabs

- Added new `null` variables for `font-size`, `font-weight`, `color`, and `:hover color` to the `.nav-link` class.

Navbars

- Breaking Navbars now require a container within (to drastically simplify spacing requirements and CSS required).

Offcanvas

- Added the new offcanvas component.

Pagination

- Pagination links now have customizable `margin-left` that are dynamically rounded on all corners when separated from one another.
- Added `transition`s to pagination links.

Popovers

- Breaking Renamed `.arrow` to `.popover-arrow` in our default popover template.
- Renamed `whiteList` option to `allowList`.

Spinners

- Spinners now honor `prefers-reduced-motion: reduce` by slowing down animations. See #31882.
- Improved spinner vertical alignment.

Toasts

- Toasts can now be positioned in a `.toast-container` with the help of positioning utilities.

- Changed default toast duration to 5 seconds.
- Removed `overflow: hidden` from toasts and replaced with proper `border-radius`s with `calc()` functions.

Tooltips

- Breaking Renamed `.arrow` to `.tooltip-arrow` in our default tooltip template.
- Breaking The default value for the `fallbackPlacements` is changed to `['top', 'right', 'bottom', 'left']` for better placement of popper elements.
- Breaking Renamed `whiteList` option to `allowList`.

Utilities

- Breaking Renamed several utilities to use logical property names instead of directional names with the addition of RTL support:
    - Renamed `.left-*` and `.right-*` to `.start-*` and `.end-*`.
    - Renamed `.float-left` and `.float-right` to `.float-start` and `.float-end`.
    - Renamed `.border-left` and `.border-right` to `.border-start` and `.border-end`.
    - Renamed `.rounded-left` and `.rounded-right` to `.rounded-start` and `.rounded-end`.
    - Renamed `.ml-*` and `.mr-*` to `.ms-*` and `.me-*`.
    - Renamed `.pl-*` and `.pr-*` to `.ps-*` and `.pe-*`.
    - Renamed `.text-left` and `.text-right` to `.text-start` and `.text-end`.
- Breaking Disabled negative margins by default.
- Added new `.bg-body` class for quickly setting the `<body>`'s background to additional elements.
- Added new position utilities for `top`, `right`, `bottom`, and `left`. Values include `0`, `50%`, and `100%` for each property.
- Added new `.translate-middle-x` & `.translate-middle-y` utilities to horizontally or vertically center absolute/fixed positioned elements.
- Added new `border-width` utilities.
- Breaking Renamed `.text-monospace` to `.font-monospace`.
- Breaking Removed `.text-hide` as it's an antiquated method for hiding text that shouldn't be used anymore.
- Added `.fs-*` utilities for `font-size` utilities (with RFS enabled). These use the same scale as HTML's default headings (1-6, large to small), and can be modified via Sass map.
- Breaking Renamed `.font-weight-*` utilities as `.fw-*` for brevity and consistency.

- Breaking Renamed `.font-style-*` utilities as `.fst-*` for brevity and consistency.
- Added `.d-grid` to display utilities and new `gap` utilities (`.gap`) for CSS Grid and flexbox layouts.
- Breaking Removed `.rounded-sm` and `rounded-lg`, and introduced a new scale of classes, `.rounded-0` to `.rounded-3`. See #31687.
- Added new `line-height` utilities: `.lh-1`, `.lh-sm`, `.lh-base` and `.lh-lg`. See here.
- Moved the `.d-none` utility in our CSS to give it more weight over other display utilities.
- Extended the `.visually-hidden-focusable` helper to also work on containers, using `:focus-within`.

Helpers

- Breaking **Responsive embed helpers have been renamed to ratio helpers** with new class names and improved behaviors, as well as a helpful CSS variable.
    - Classes have been renamed to change `by` to `x` in the aspect ratio. For example, `.ratio-16by9` is now `.ratio-16x9`.
    - We've dropped the `.embed-responsive-item` and element group selector in favor of a simpler `.ratio > *` selector. No more class is needed, and the ratio helper now works with any HTML element.
    - The `$embed-responsive-aspect-ratios` Sass map has been renamed to `$aspect-ratios` and its values have been simplified to include the class name and the percentage as the `key: value` pair.
    - CSS variables are now generated and included for each value in the Sass map. Modify the `--bs-aspect-ratio` variable on the `.ratio` to create any custom aspect ratio.
- Breaking **"Screen reader" classes are now "visually hidden" classes.**
    - Changed the Sass file from `scss/helpers/_screenreaders.scss` to `scss/helpers/_visually-hidden.scss`
    - Renamed `.sr-only` and `.sr-only-focusable` to `.visually-hidden` and `.visually-hidden-focusable`
    - Renamed `sr-only()` and `sr-only-focusable()` mixins to `visually-hidden()` and `visually-hidden-focusable()`.
- `bootstrap-utilities.css` now also includes our helpers. Helpers don't need to be imported in custom builds anymore.

JavaScript

- **Dropped jQuery dependency** and rewrote plugins to be in regular JavaScript.

- Breaking Data attributes for all JavaScript plugins are now namespaced to help distinguish Bootstrap functionality from third parties and your own code. For example, we use `data-bs-toggle` instead of `data-toggle`.
- **All plugins can now accept a CSS selector as the first argument.** You can either pass a DOM element or any valid CSS selector to create a new instance of the plugin:

```
•  const modal = new bootstrap.Modal('#myModal')
const dropdown = new bootstrap.Dropdown('[data-bs-toggle="dropdown"]')
```

- `popperConfig` can be passed as a function that accepts the Bootstrap's default Popper config as an argument, so that you can merge this default configuration in your way. **Applies to dropdowns, popovers, and tooltips.**
- The default value for the `fallbackPlacements` is changed to `['top', 'right', 'bottom', 'left']` for better placement of Popper elements. **Applies to dropdowns, popovers, and tooltips.**
- Removed underscore from public static methods like `_getInstance()` → `getInstance()`.

https://getbootstrap.com/docs/5.2/migration/