



Bootstrap v4.5 Manual

— Compiled by **Rit Chakraborty**

Accessibility

Search...

Getting started

[Introduction](#)

[Download](#)

[Contents](#)

[Browsers & devices](#)

[JavaScript](#)

[Theming](#)

[Build tools](#)

[Webpack](#)

Accessibility

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Bootstrap provides an easy-to-use framework of ready-made styles, layout tools, and interactive components, allowing developers to create websites and applications that are visually appealing, functionally rich, and accessible out of the box.

Overview and Limitations

The overall accessibility of any project built with Bootstrap depends in large part on the author's markup, additional styling, and scripting they've included. However, provided that these have been implemented correctly, it should be perfectly possible to create websites and applications with Bootstrap that fulfill [WCAG \(Web Content Accessibility Guidelines\) 2.0](#) (A/AA/AAA), [Section 508](#) and similar accessibility standards and requirements.

Structural markup

Bootstrap's styling and layout can be applied to a wide range of markup structures. This documentation aims to provide developers with best practice examples to demonstrate the use of Bootstrap itself and illustrate appropriate semantic markup, including ways in which potential accessibility concerns can be addressed.

Interactive components

Bootstrap's interactive components—such as modal dialogs, dropdown menus and custom tooltips—are designed to work for touch, mouse and keyboard users. Through the use of relevant [WAI \(Web Accessibility Initiative\)-ARIA \(Accessible Rich Internet Applications\)](#) roles and attributes, these components should also be understandable and operable using assistive technologies (such as screen readers).

Because Bootstrap's components are purposely designed to be fairly generic, authors may need to include further [ARIA \(Accessible Rich Internet Applications\)](#) roles and attributes, as well as JavaScript behavior, to more accurately convey the precise nature and functionality of their component. This is usually noted in the documentation.

Color contrast

Most colors that currently make up Bootstrap's default palette—used throughout the framework for things such as button variations, alert variations, form validation indicators—lead to *insufficient* color contrast (below the recommended [WCAG 2.0 color contrast ratio of 4.5:1](#)) when used against a light background. Authors will need to manually modify/extend these default colors to ensure adequate color contrast ratios.

Visually hidden content

Content which should be visually hidden, but remain accessible to assistive technologies such as screen readers, can be styled using the `.sr-only` class. This can be useful in situations where additional visual information or cues (such as meaning denoted through the use of color) need to also be conveyed to non-visual users.

```
<p class="text-danger">
  <span class="sr-only">Danger: </span>
  This action is not reversible
</p>
```

For visually hidden interactive controls, such as traditional “skip” links, `.sr-only` can be combined with the `.sr-only-focusable` class. This will ensure that the control becomes visible once focused (for sighted keyboard users).

```
<a class="sr-only sr-only-focusable" href="#content">Skip to main content</a>
```

Copy

Reduced motion

Bootstrap includes support for the [prefers-reduced-motion media feature](#). In browsers/environments that allow the user to specify their preference for reduced motion, most CSS transition effects in Bootstrap (for instance, when a modal dialog is opened or closed, or the sliding animation in carousels) will be disabled.

Additional resources

- [Web Content Accessibility Guidelines \(WCAG\) 2.0](#)
- [The A11Y Project](#)
- [MDN accessibility documentation](#)
- [Tenon.io Accessibility Checker](#)
- [Colour Contrast Analyser \(CCA\)](#)
- [“HTML Codesniffer” bookmarklet for identifying accessibility issues](#)

Browsers and devices

Search...

Getting started

[Introduction](#)

[Download](#)

[Contents](#)

Browsers & devices

[JavaScript](#)

[Theming](#)

[Build tools](#)

[Webpack](#)

[Accessibility](#)

Layout

Content

Components

Utilities

Extend

Migration

About



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Supported browsers

Bootstrap supports the **latest, stable releases** of all major browsers and platforms. On Windows, we support Internet Explorer 10-11 / Microsoft Edge.

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
```

Copy

```
>= 1%
last 1 major version
not dead
Chrome >= 45
Firefox >= 38
Edge >= 12
Explorer >= 10
iOS >= 9
Safari >= 9
Android >= 4.4
Opera >= 30
```

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	Microsoft Edge
Android	Supported	Supported	N/A	Android v5.0+ supported	Supported
iOS	Supported	Supported	Supported	N/A	Supported
Windows 10 Mobile	N/A	N/A	N/A	N/A	Supported

Desktop browsers

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

	Chrome	Firefox	Internet Explorer	Microsoft Edge	Opera	Safari
Mac	Supported	Supported	N/A	Supported	Supported	Supported
Windows	Supported	Supported	Supported, IE10+	Supported	Supported	Not 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, Firefox for Linux, and Internet Explorer 9, though they are not officially supported.

For a list of some of the browser bugs that Bootstrap has to grapple with, see our [Wall of browser bugs](#).

Internet Explorer

Internet Explorer 10+ is supported; IE9 and down is not. Please be aware that some CSS3 properties and HTML5 elements are not fully supported in IE10, or require prefixed properties for full functionality. Visit [Can I use...](#) for details on browser support of CSS3 and HTML5 features. **If you require IE8-9 support, use Bootstrap 3.**

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.

Sticky :hover/:focus on iOS

While `:hover` isn't possible on most touch devices, iOS emulates this behavior, resulting in "sticky" hover styles that persist after tapping one element. These hover styles are only removed when users tap another element. This behavior is considered largely undesirable and appears to not be an issue on Android or Windows devices.

Throughout our v4 alpha and beta releases, we included incomplete and commented out code for opting into a media query shim that would disable hover styles in touch device browsers that emulate hovering. This work was never fully completed or enabled, but to avoid complete breakage, we've opted to deprecate [this shim](#) and keep the mixins as shortcuts for the pseudo-classes.

Printing

Even in some modern browsers, printing can be quirky.

As of Safari v8.0, use of the fixed-width `.container` class can cause Safari to use an unusually small font size when printing. See [issue #14868](#) and [WebKit bug #138192](#) for more details. One potential workaround is the following CSS:

```
@media print {  
  .container {  
    width: auto;  
  }  
}
```

Copy

Android stock browser

Out of the box, Android 4.1 (and even some newer releases apparently) ship with the Browser app as the default web browser of choice (as opposed to Chrome). Unfortunately, the Browser app has lots of bugs and inconsistencies with CSS in general.

Select menu

On `<select>` elements, the Android stock browser will not display the side controls if there is a `border-radius` and/or `border` applied. (See [this StackOverflow question](#) for details.) Use the snippet of code below to remove the offending CSS and render the `<select>` as an unstyled element on the Android stock browser. The user agent sniffing avoids interference with Chrome, Safari, and Mozilla browsers.

```
<script>  
$(function () {  
  var nua = navigator.userAgent  
  var isAndroid = (nua.indexOf('Mozilla/5.0') > -1 && nua.indexOf('Android ') >  
-1 && nua.indexOf('AppleWebKit') > -1 && nua.indexOf('Chrome') === -1)  
  if (isAndroid) {  
    $('select.form-control').removeClass('form-control').css('width', '100%')  
  }  
})  
</script>
```

Copy

Want to see an example? [Check out this JS Bin demo.](#)

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](#).

Build tools

Search...

Getting started

[Introduction](#)

[Download](#)

[Contents](#)

[Browsers & devices](#)

[JavaScript](#)

[Theming](#)

Build tools

[Webpack](#)

[Accessibility](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Tooling setup

Bootstrap uses [npm scripts](#) for its build system. Our [package.json](#) includes convenient methods for working with the framework, including compiling code, running tests, and more.

To use our build system and run our documentation locally, you'll need a copy of Bootstrap's source files and Node. Follow these steps and you should be ready to rock:

1. [Download and install Node.js](#), which we use to manage our dependencies.
2. Navigate to the root `/bootstrap` directory and run `npm install` to install our local dependencies listed in [package.json](#).
3. [Install Ruby](#), install [Bundler](#) with `gem install bundler`, and finally run `bundle install`. This will install all Ruby dependencies, such as Jekyll and plugins.
 - **Windows users:** Read [this guide](#) to get Jekyll up and running without problems.

When completed, you'll be able to run the various commands provided from the command line.

Using npm scripts

Our [package.json](#) includes the following commands and tasks:

Task	Description
<code>npm run dist</code>	<code>npm run dist</code> creates the <code>/dist/</code> directory with compiled files. Uses Sass , Autoprefixer , and terser .
<code>npm test</code>	Same as <code>npm run dist</code> plus it runs tests locally
<code>npm run docs</code>	Builds and lints CSS and JavaScript for docs. You can then run the documentation locally via <code>npm run docs-serve</code> .

Run `npm run` to see all the npm scripts.

Autoprefixer

Bootstrap uses [Autoprefixer](#) (included in our build process) to automatically add vendor prefixes to some CSS properties at build time. Doing so saves us time and code by allowing us to write key parts of our CSS a single time while eliminating the need for vendor mixins like those found in v3.

We maintain the list of browsers supported through Autoprefixer in a separate file within our GitHub repository. See [browserslistrc](#) for details.

Local documentation

Running our documentation locally requires the use of Jekyll, a decently flexible static site generator that provides us: basic includes, Markdown-based files, templates, and more. Here's how to get it started:

1. Run through the [tooling setup](#) above to install Jekyll (the site builder) and other Ruby dependencies with `bundle install`.
2. From the root `/bootstrap` directory, run `npm run docs-serve` in the command line.
3. Open `http://localhost:9001` in your browser, and voilà.

Learn more about using Jekyll by reading its [documentation](#).

Troubleshooting

Should you encounter problems with installing dependencies, uninstall all previous dependency versions (global and local). Then, rerun `npm install`.

Contents

Search...

Getting started

[Introduction](#)

[Download](#)

Contents

[Browsers & devices](#)

[JavaScript](#)

[Theming](#)

[Build tools](#)

[Webpack](#)

[Accessibility](#)

[Layout](#)

Content

[Components](#)

[Utilities](#)

Extend

[Migration](#)

[About](#)



Adobe Creative Cloud for
Teams starting at \$33.99
per month.

ads via Carbon

Precompiled Bootstrap

Once downloaded, unzip the compressed folder and you'll see something like this:

```
bootstrap/
  └── css/
    ├── bootstrap-grid.css
    ├── bootstrap-grid.css.map
    ├── bootstrap-grid.min.css
    ├── bootstrap-grid.min.css.map
    ├── bootstrap-reboot.css
    ├── bootstrap-reboot.css.map
    ├── bootstrap-reboot.min.css
    ├── bootstrap-reboot.min.css.map
    ├── bootstrap.css
    ├── bootstrap.css.map
    ├── bootstrap.min.css
    └── bootstrap.min.css.map
  └── js/
    ├── bootstrap.bundle.js
    ├── bootstrap.bundle.js.map
    ├── bootstrap.bundle.min.js
    ├── bootstrap.bundle.min.js.map
    ├── bootstrap.js
    ├── bootstrap.js.map
    ├── bootstrap.min.js
    └── bootstrap.min.js.map
```

Copy

This is the most basic form of Bootstrap: precompiled files for quick drop-in usage in nearly any web project. We provide compiled CSS and JS (`bootstrap.*`), as well as compiled and minified CSS and JS (`bootstrap.min.*`). [source maps](#) (`bootstrap.*.map`) are available for use with certain browsers' developer tools. Bundled JS files (`bootstrap.bundle.js` and minified `bootstrap.bundle.min.js`) include [Popper](#), but not [jQuery](#).

CSS files

Bootstrap includes a handful of options for including some or all of our compiled CSS.

CSS files	Layout	Content	Components	Utilities
<code>bootstrap.css</code> <code>bootstrap.min.css</code>	Included	Included	Included	Included
<code>bootstrap-grid.css</code> <code>bootstrap-grid.min.css</code>	Only grid system	Not included	Not included	Only flex utilities

CSS files	Layout	Content	Components	Utilities
bootstrap-reboot.css bootstrap-reboot.min.css	Not included	Only Reboot	Not included	Not included

JS files

Similarly, we have options for including some or all of our compiled JavaScript.

JS files	Popper	jQuery
bootstrap.bundle.js bootstrap.bundle.min.js	Included	Not included
bootstrap.js bootstrap.min.js	Not included	Not included

Bootstrap source code

The Bootstrap source code download includes the precompiled CSS and JavaScript assets, along with source Sass, JavaScript, and documentation. More specifically, it includes the following and more:

```
bootstrap/
├── dist/
│   ├── css/
│   └── js/
├── site/
│   └── docs/
│       └── 4.5/
│           └── examples/
└── js/
    └── scss/
```

Copy

The `scss/` and `js/` are the source code for our CSS and JavaScript. The `dist/` folder includes everything listed in the precompiled download section above. The `site/docs/` folder includes the source code for our documentation, and `examples/` of Bootstrap usage. Beyond that, any other included file provides support for packages, license information, and development.

Download

Search...

Getting started

[Introduction](#)

[Download](#)

[Contents](#)

[Browsers & devices](#)

[JavaScript](#)

[Theming](#)

[Build tools](#)

[Webpack](#)

[Accessibility](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10
free Adobe Stock images.

ads via Carbon

Compiled CSS and JS

Download ready-to-use compiled code for **Bootstrap v4.5.0** to easily drop into your project, which includes:

- Compiled and minified CSS bundles (see [CSS files comparison](#))
- Compiled and minified JavaScript plugins

This doesn't include documentation, source files, or any optional JavaScript dependencies (jQuery and Popper.js).

[Download](#)

Source files

Compile Bootstrap with your own asset pipeline by downloading our source Sass, JavaScript, and documentation files. This option requires some additional tooling:

- Sass compiler (Libsass or Ruby Sass is supported) for compiling your CSS.
- [Autoprefixer](#) for CSS vendor prefixing

Should you require [build tools](#), they are included for developing Bootstrap and its docs, but they're likely unsuitable for your own purposes.

[Download source](#)

Examples

If you want to download and examine our [examples](#), you can grab the already built examples:

[Download Examples](#)

BootstrapCDN

Skip the download with [BootstrapCDN](#) to deliver cached version of Bootstrap's compiled CSS and JS to your project.

Copy

```
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
      integrity="sha384-9AIt2nRpC12Uk9gS9baD1411NQApFmC26EwAOH8WgZ15MYxFfc+NcPb1dKGj7SK"
      crossorigin="anonymous">
<script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
      integrity="sha384-0gVRvuATP1z7JjHLku0U7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
      crossorigin="anonymous"></script>
```

If you're using our compiled JavaScript, don't forget to include CDN versions of jQuery and Popper.js before it.

```
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
      integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+0GpamoFVy38MVBN+E+IbbVYUew+OrCXaRkj"
      crossorigin="anonymous"></script>
<script
      src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
      integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
      crossorigin="anonymous"></script>
```

Copy

Package managers

Pull in Bootstrap's [source files](#) into nearly any project with some of the most popular package managers. No matter the package manager, Bootstrap will [require a Sass compiler](#) and [Autoprefixer](#) for a setup that matches our official compiled versions.

npm

Install Bootstrap in your Node.js powered apps with [the npm package](#):

```
$ npm install bootstrap
```

Copy

`require('bootstrap')` will load all of Bootstrap's jQuery plugins onto the jQuery object. The `bootstrap` module itself does not export anything. You can manually load Bootstrap's jQuery plugins individually by loading the `/js/*.js` files under the package's top-level directory.

Bootstrap's `package.json` contains some additional metadata under the following keys:

- `sass` - path to Bootstrap's main [Sass](#) source file
- `style` - path to Bootstrap's non-minified CSS that's been precompiled using the default settings (no customization)

yarn

Install Bootstrap in your Node.js powered apps with [the yarn package](#):

```
$ yarn add bootstrap
```

Copy

RubyGems

Install Bootstrap in your Ruby apps using [Bundler \(recommended\)](#) and [RubyGems](#) by adding the following line to your [Gemfile](#):

```
gem 'bootstrap', '~> 4.5.0'
```

Copy

Alternatively, if you're not using Bundler, you can install the gem by running this command:

```
$ gem install bootstrap -v 4.5.0
```

Copy

[See the gem's README](#) for further details.

Composer

You can also install and manage Bootstrap's Sass and JavaScript using [Composer](#):

```
$ composer require twbs/bootstrap:4.5.0
```

Copy

NuGet

If you develop in .NET, you can also install and manage Bootstrap's [CSS](#) or [Sass](#) and JavaScript using [NuGet](#):

```
PM> Install-Package bootstrap
```

Copy

```
PM> Install-Package bootstrap.sass
```

Copy

Introduction

Search...

Getting started

Introduction

[Download](#)

[Contents](#)

[Browsers & devices](#)

[JavaScript](#)

[Theming](#)

[Build tools](#)

[Webpack](#)

[Accessibility](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10
free Adobe Stock images.

ads via Carbon

Quick start

Looking to quickly add Bootstrap to your project? Use BootstrapCDN, provided for free by the folks at StackPath. Using a package manager or need to download the source files? [Head to the downloads page](#).

CSS

Copy-paste the stylesheet `<link>` into your `<head>` before all other stylesheets to load our CSS.

```
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
      integrity="sha384-9AI2nRpC12Uk9gS9baD1411NQApFmC26EwAOH8WgZ15MYxFfc+NcPb1dKGj7Sk"
      crossorigin="anonymous">
```

Copy

JS

Many of our components require the use of JavaScript to function. Specifically, they require [jQuery](#), [Popper.js](#), and our own JavaScript plugins. Place the following `<script>`s near the end of your pages, right before the closing `</body>` tag, to enable them. jQuery must come first, then Popper.js, and then our JavaScript plugins.

We use [jQuery's slim build](#), but the full version is also supported.

```
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
      integrity="sha384-9AI2nRpC12Uk9gS9baD1411NQApFmC26EwAOH8WgZ15MYxFfc+NcPb1dKGj7Sk"
      crossorigin="anonymous"></script>
<script
      src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
      integrity="sha384-9AI2nRpC12Uk9gS9baD1411NQApFmC26EwAOH8WgZ15MYxFfc+NcPb1dKGj7Sk"
      crossorigin="anonymous"></script>
<script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
      integrity="sha384-9AI2nRpC12Uk9gS9baD1411NQApFmC26EwAOH8WgZ15MYxFfc+NcPb1dKGj7Sk"
      crossorigin="anonymous"></script>
```

Copy

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

Our `bootstrap.bundle.js` and `bootstrap.bundle.min.js` include [Popper](#), but not [jQuery](#). For more information about what's included in Bootstrap, please see our [contents](#) section.

Starter template

Be sure to have your pages set up with the latest design and development standards. That means using an HTML5 doctype and including a viewport meta tag for proper responsive behaviors. Put it all together and your pages should look like this:

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
      integrity="sha384-9aIt2nRpC12Uk9gS9baD1411NQApFmC26EwAOH8WgZ15MYxFfc+NcPb1dKGj7SK"
      crossorigin="anonymous">

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
      integrity="sha384-DfXz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
      crossorigin="anonymous"></script>
    <script
      src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
      integrity="sha384-"
      Q6E9RHvbIyZFJof+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
      crossorigin="anonymous"></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
      integrity="sha384-"
      OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ip6Tp75j7Bh/kR0JKI"
      crossorigin="anonymous"></script>
  </body>
</html>
```

Copy

That's all you need for overall page requirements. Visit the [Layout docs](#) or [our official examples](#) to start laying out your site's content and components.

Important globals

Bootstrap employs a handful of important global styles and settings that you'll need to be aware of when using it, 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 incomplete styling, but including it shouldn't cause any considerable hiccups.

Copy

```
<!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, shrink-to-fit=no">
```

Copy

You can see an example of this in action in the [starter template](#).

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;
}
```

Copy

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.

- Follow [@getbootstrap on Twitter](#).
- Read and subscribe to [The Official Bootstrap Blog](#).
- Join [the official Slack room](#).
- Chat with fellow Bootstrappers in IRC. On the `irc.freenode.net` server, in the `##bootstrap` channel.
- Implementation help may be found at Stack Overflow (tagged `bootstrap-4`).
- Developers should use the keyword `bootstrap` on packages which 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.

JavaScript

Search...

Getting started

[Introduction](#)

[Download](#)

[Contents](#)

[Browsers & devices](#)

JavaScript

[Theming](#)

[Build tools](#)

[Webpack](#)

[Accessibility](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

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.

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. Also note that **all plugins depend on jQuery** (this means jQuery must be included **before** the plugin files). [Consult our package.json](#) to see which versions of jQuery are supported.

Our dropdowns, popovers and tooltips also depend on [Popper.js](#).

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.)

However, in some situations it may be desirable to disable this functionality. To disable the data attribute API, unbind all events on the document namespaced with `data-api` like so:

```
$(document).off('.data-api')
```

Copy

Alternatively, to target a specific plugin, just include the plugin's name as a namespace along with the `data-api` namespace like this:

```
$(document).off('.alert.data-api')
```

Copy

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()`.

```
$('#myModal').on('show.bs.modal', function (e) {  
  if (!data) {  
    return e.preventDefault() // stops modal from being shown  
  }  
})
```

Copy

Programmatic API

We also believe you should be able to use all Bootstrap plugins purely through the JavaScript API. All public APIs are single, chainable methods, and return the collection acted upon.

```
$('.btn.danger').button('toggle').addClass('fat')
```

Copy

All methods should accept an optional options object, a string which targets a particular method, or nothing (which initiates a plugin with default behavior):

```
$('#myModal').modal() // initialized with defaults  
$('#myModal').modal({ keyboard: false }) // initialized with no keyboard  
$('#myModal').modal('show') // initializes and invokes show immediately
```

Copy

Each plugin also exposes its raw constructor on a `Constructor` property:

`$.fn.popover.Constructor`. If you'd like to get a particular plugin instance, retrieve it directly from an element: `$('[rel="popover"]').data('popover')`.

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.

```
$('#myCollapse').on('shown.bs.collapse', function (e) {  
  // Action to execute once the collapsible area is expanded  
})
```

Copy

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

```
$('#myCarousel').on('slid.bs.carousel', function (e) {  
  $('#myCarousel').carousel('2') // Will slide to the slide 2 as soon as the  
  // transition to slide 1 is finished  
})  
  
$('#myCarousel').carousel('1') // Will start sliding to the slide 1 and returns  
// to the caller  
$('#myCarousel').carousel('2') // !! Will be ignored, as the transition to the  
// slide 1 is not finished !!
```

Copy

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  
$.fn.modal.Constructor.Default.keyboard = false
```

Copy

No conflict

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.

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

Copy

Version numbers

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

```
$.fn.tooltip.Constructor.VERSION // => "4.5.0"
```

Copy

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.

Util

All Bootstrap's JavaScript files depend on `util.js` and it has to be included alongside the other JavaScript files. If you're using the compiled (or minified) `bootstrap.js`, there is no need to include this—it's already there.

`util.js` includes utility functions and a basic helper for `transitionEnd` events as well as a CSS transition emulator. It's used by the other plugins to check for CSS transition support and to catch hanging transitions.

Sanitizer

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

The default `whiteList` value is the following:

Copy

```

var ARIA_ATTRIBUTE_PATTERN = /^aria-[\\w-]*$/i
var DefaultWhitelist = {
    // 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 `whiteList` you can do the following:

```

var myDefaultWhiteList = $.fn.tooltip.Constructor.Default.whiteList

// To allow table elements
myDefaultWhiteList.table = []

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

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

```

Copy

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

```

$('#yourTooltip').tooltip({
    sanitizeFn: function (content) {
        return DOMPurify.sanitize(content)
    }
})

```

Copy

Theming Bootstrap

Search...

Getting started

[Introduction](#)

[Download](#)

[Contents](#)

[Browsers & devices](#)

[JavaScript](#)

Theming

[Build tools](#)

[Webpack](#)

[Accessibility](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10
free Adobe Stock images.

ads via Carbon

Introduction

In Bootstrap 3, theming was largely driven by variable overrides in LESS, custom CSS, and a separate theme stylesheet that we included in our `dist` files. With some effort, one could completely redesign the look of Bootstrap 3 without touching the core files. Bootstrap 4 provides a familiar, but slightly different approach.

Now, theming is accomplished by Sass variables, Sass maps, and custom CSS. There's no more dedicated theme stylesheet; instead, you can enable the built-in theme to add gradients, shadows, and more.

Sass

Utilize our source Sass files to take advantage of variables, maps, mixins, and more. In our build we've increased the Sass rounding precision to 6 (by default it's 5) to prevent issues with browser rounding.

File structure

Whenever possible, avoid modifying Bootstrap's core files. For Sass, that means creating your own stylesheet that imports Bootstrap so you can modify and extend it. Assuming you're using a package manager like npm, you'll have a file structure that looks like this:

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

Copy

If you've downloaded our source files and aren't using a package manager, you'll want to manually setup something similar to that structure, keeping Bootstrap's source files separate from your own.

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

Copy

Importing

In your `custom.scss`, you'll import Bootstrap's source Sass files. You have two options: include all of Bootstrap, or pick the parts you need. We encourage the latter, though be aware there are some requirements and dependencies across our components. You also will need to include some JavaScript for our plugins.

```
// Custom.scss
// Option A: Include all of Bootstrap

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

Copy

```
// Custom.scss
// Option B: Include parts of Bootstrap

// Required
@import "../node_modules/bootstrap/scss/functions";
@import "../node_modules/bootstrap/scss/variables";
@import "../node_modules/bootstrap/scss/mixins";

// Optional
@import "../node_modules/bootstrap/scss/reboot";
@import "../node_modules/bootstrap/scss/type";
@import "../node_modules/bootstrap/scss/images";
@import "../node_modules/bootstrap/scss/code";
@import "../node_modules/bootstrap/scss/grid";
```

Copy

With that setup in place, you can begin to modify any of the Sass variables and maps in your `custom.scss`. You can also start to add parts of Bootstrap under the `// Optional` section as needed. We suggest using the full import stack from our `bootstrap.scss` file as your starting point.

Variable defaults

Every Sass variable in Bootstrap 4 includes the `!default` flag allowing you to override the variable's default value in your own Sass without modifying Bootstrap's source code. Copy and paste variables as needed, modify their values, and remove the `!default` flag. If a variable has already been assigned, then it won't be re-assigned by the default values in Bootstrap.

You will find the complete list of Bootstrap's variables in `scss/_variables.scss`. Some variables are set to `null`, these variables don't output the property unless they are overridden in your configuration.

Variable overrides within the same Sass file can come before or after the default variables. However, when overriding across Sass files, your overrides must come before you import Bootstrap's Sass files.

Here's an example that changes the `background-color` and `color` for the `<body>` when importing and compiling Bootstrap via npm:

```
// Your variable overrides
$body-bg: #000;
$body-color: #111;

// Bootstrap and its default variables
@import "../node_modules/bootstrap/scss/bootstrap";
```

Copy

Repeat as necessary for any variable in Bootstrap, including the global options below.

Maps and loops

Bootstrap 4 includes a handful of Sass maps, key value pairs that make it easier to generate families of related CSS. We use Sass maps for our colors, grid breakpoints, and more. Just like Sass variables, all Sass maps include the `!default` flag and can be overridden and extended.

Some of our Sass maps are merged into empty ones by default. This is done to allow easy expansion of a given Sass map, but comes at the cost of making *removing* items from a map slightly more difficult.

Modify map

To modify an existing color in our `$theme-colors` map, add the following to your custom Sass file:

Copy

```
$theme-colors: (
  "primary": #0074d9,
  "danger": #ff4136
);
```

Add to map

To add a new color to `$theme-colors`, add the new key and value:

```
$theme-colors: (
  "custom-color": #900
);
```

Copy

Remove from map

To remove colors from `$theme-colors`, or any other map, use `map-remove`. Be aware you must insert it between our requirements and options:

```
// Required
@import "../node_modules/bootstrap/scss/functions";
@import "../node_modules/bootstrap/scss/variables";
@import "../node_modules/bootstrap/scss/mixins";

$theme-colors: map-remove($theme-colors, "info", "light", "dark");

// Optional
@import "../node_modules/bootstrap/scss/root";
@import "../node_modules/bootstrap/scss/reboot";
@import "../node_modules/bootstrap/scss/type";
...
```

Copy

Required keys

Bootstrap assumes the presence of some specific keys within Sass maps as we used and extend these ourselves. As you customize the included maps, you may encounter errors where a specific Sass map's key is being used.

For example, we use the `primary`, `success`, and `danger` keys from `$theme-colors` for links, buttons, and form states. Replacing the values of these keys should present no issues, but removing them may cause Sass compilation issues. In these instances, you'll need to modify the Sass code that makes use of those values.

Functions

Bootstrap utilizes several Sass functions, but only a subset are applicable to general theming. We've included three functions for getting values from the color maps:

```
@function color($key: "blue") {
  @return map-get($colors, $key);
}

@function theme-color($key: "primary") {
  @return map-get($theme-colors, $key);
}

@function gray($key: "100") {
  @return map-get($grays, $key);
}
```

Copy

These allow you to pick one color from a Sass map much like how you'd use a color variable from v3.

Copy

```
.custom-element {  
  color: gray("100");  
  background-color: theme-color("dark");  
}
```

We also have another function for getting a particular *level* of color from the `$theme-colors` map. Negative level values will lighten the color, while higher levels will darken.

```
@function theme-color-level($color-name: "primary", $level: 0) {  
  $color: theme-color($color-name);  
  $color-base: if($level > 0, #000, #fff);  
  $level: abs($level);  
  
  @return mix($color-base, $color, $level * $theme-color-interval);  
}
```

Copy

In practice, you'd call the function and pass in two parameters: the name of the color from `$theme-colors` (e.g., primary or danger) and a numeric level.

```
.custom-element {  
  color: theme-color-level(primary, -10);  
}
```

Copy

Additional functions could be added in the future or your own custom Sass to create level functions for additional Sass maps, or even a generic one if you wanted to be more verbose.

Color contrast

An additional function we include in Bootstrap is the color contrast function, `color-yiq`. It utilizes the [YIQ color space](#) to automatically return a light (#fff) or dark (#111) contrast color based on the specified base color. This function is especially useful for mixins or loops where you're generating multiple classes.

For example, to generate color swatches from our `$theme-colors` map:

```
@each $color, $value in $theme-colors {  
  .swatch-#{$color} {  
    color: color-yiq($value);  
  }  
}
```

Copy

It can also be used for one-off contrast needs:

```
.custom-element {  
  color: color-yiq(#000); // returns `color: #fff`  
}
```

Copy

You can also specify a base color with our color map functions:

```
.custom-element {  
  color: color-yiq(theme-color("dark")); // returns `color: #fff`  
}
```

Copy

Escape SVG

We use the `escape-svg` function to escape the `<`, `>` and `#` characters for SVG background images. These characters need to be escaped to properly render the background images in IE.

Add and Subtract functions

We use the `add` and `subtract` functions to wrap the CSS `calc` function. The primary purpose of these functions is to avoid errors when a “unitless” `0` value is passed into a `calc` expression. Expressions like `calc(10px - 0)` will return an error in all browsers, despite being mathematically correct.

Example where the calc is valid:

```
$border-radius: .25rem;  
$border-width: 1px;  
  
.element {  
  // Output calc(.25rem - 1px) is valid  
  border-radius: calc($border-radius - $border-width);  
}  
  
.element {  
  // Output the same calc(.25rem - 1px) as above  
  border-radius: subtract($border-radius, $border-width);  
}
```

Copy

Example where the calc is invalid:

```
$border-radius: .25rem;  
$border-width: 0;  
  
.element {  
  // Output calc(.25rem - 0) is invalid  
  border-radius: calc($border-radius - $border-width);  
}  
  
.element {  
  // Output .25rem  
  border-radius: subtract($border-radius, $border-width);  
}
```

Copy

Sass options

Customize Bootstrap 4 with our built-in custom variables file and easily toggle global CSS preferences with new `$enable-*` Sass variables. Override a variable’s value and recompile with `npm run test` as needed.

You can find and customize these variables for key global options in Bootstrap’s `scss/_variables.scss` file.

Variable	Values	Description
<code>\$spacer</code>	<code>1rem</code> (default), or any value > 0	Specifies the default spacer value to programmatically generate our spacer utilities .
<code>\$enable-rounded</code>	<code>true</code> (default) or <code>false</code>	Enables predefined <code>border-radius</code> styles on various components.
<code>\$enable-shadows</code>	<code>true</code> or <code>false</code> (default)	Enables predefined <code>box-shadow</code> styles on various components.
<code>\$enable-gradients</code>	<code>true</code> or <code>false</code> (default)	Enables predefined gradients via <code>background-image</code> styles on various components.

Variable	Values	Description
<code>\$enable-transitions</code>	<code>true</code> (default) or <code>false</code>	Enables predefined <code>transitions</code> on various components.
<code>\$enable-prefers-reduced-motion-media-query</code>	<code>true</code> (default) or <code>false</code>	Enables the prefers-reduced-motion media query , which suppresses certain animations/transitions based on the users' browser/operating system preferences.
<code>\$enable-hover-media-query</code>	<code>true</code> or <code>false</code> (default)	Deprecated
<code>\$enable-grid-classes</code>	<code>true</code> (default) or <code>false</code>	Enables the generation of CSS classes for the grid system (e.g., <code>.container</code> , <code>.row</code> , <code>.col-md-1</code> , etc.).
<code>\$enable-caret</code>	<code>true</code> (default) or <code>false</code>	Enables pseudo element caret on <code>.dropdown-toggle</code> .
<code>\$enable-pointer-cursor-for-buttons</code>	<code>true</code> (default) or <code>false</code>	Add "hand" cursor to non-disabled button elements.
<code>\$enable-print-styles</code>	<code>true</code> (default) or <code>false</code>	Enables styles for optimizing printing.
<code>\$enable-responsive-font-sizes</code>	<code>true</code> or <code>false</code> (default)	Enables responsive font sizes .
<code>\$enable-validation-icons</code>	<code>true</code> (default) or <code>false</code>	Enables <code>background-image</code> icons within textual inputs and some custom forms for validation states.
<code>\$enable-deprecation-messages</code>	<code>true</code> or <code>false</code> (default)	Set to <code>true</code> to show warnings when using any of the deprecated mixins and functions that are planned to be removed in <code>v5</code> .

Color

Many of Bootstrap's various components and utilities are built through a series of colors defined in a Sass map. This map can be looped over in Sass to quickly generate a series of rulesets.

All colors

All colors available in Bootstrap 4, are available as Sass variables and a Sass map in `scss/_variables.scss` file. This will be expanded upon in subsequent minor releases to add additional shades, much like the [grayscale palette](#) we already include.

Blue

Indigo

Purple

Pink

Red

Orange

Yellow

Green

Teal

Cyan

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

```
// With variable  
.alpha { color: $purple; }  
  
// From the Sass map with our `color()` function  
.beta { color: color("purple"); }
```

Copy

[Color utility classes](#) are also available for setting `color` and `background-color`.

In the future, we'll aim to provide Sass maps and variables for shades of each color as we've done with the grayscale colors below.

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

Grays

An expansive set of gray variables and a Sass map in `scss/_variables.scss` for consistent shades of gray across your project. Note that these are "cool grays", which tend towards a subtle blue tone, rather than neutral grays.

100

200

300

400

500

600

700

800

900

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,
  "white": $white,
  "gray": $gray-600,
  "gray-dark": $gray-800
) !default;
```

Copy

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.

Components

Many of Bootstrap's components and utilities are built with `@each` loops that 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.

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` component and all our `.bg-*` background utilities.

```
// Generate alert modifier classes
@each $color, $value in $theme-colors {
  .alert-#{$color} {
    @include alert-variant(theme-color-level($color, -10), theme-color-level($color, -9), theme-color-level($color, 6));
  }
}

// Generate `bg-*` color utilities
@each $color, $value in $theme-colors {
  @include bg-variant('.bg-#{$color}', $value);
}
```

Copy

Responsive

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

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

    .text#{$infix}-left { text-align: left !important; }
    .text#{$infix}-right { text-align: right !important; }
    .text#{$infix}-center { text-align: center !important; }
  }
}
```

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

CSS variables

Bootstrap 4 includes around two dozen [CSS custom properties \(variables\)](#) in its compiled CSS. 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.

Available variables

Here are the variables we include (note that the `:root` is required). They're located in our `_root.scss` file.

```
Copy
:root {
  --blue: #007bff;
  --indigo: #6610f2;
  --purple: #6f42c1;
  --pink: #e83e8c;
  --red: #dc3545;
  --orange: #fd7e14;
  --yellow: #ffc107;
  --green: #28a745;
  --teal: #20c997;
  --cyan: #17a2b8;
  --white: #fff;
  --gray: #6c757d;
  --gray-dark: #343a40;
  --primary: #007bff;
  --secondary: #6c757d;
  --success: #28a745;
  --info: #17a2b8;
  --warning: #ffc107;
  --danger: #dc3545;
  --light: #f8f9fa;
  --dark: #343a40;
  --breakpoint-xs: 0;
  --breakpoint-sm: 576px;
  --breakpoint-md: 768px;
  --breakpoint-lg: 992px;
  --breakpoint-xl: 1200px;
  --font-family-sans-serif: -apple-system, BlinkMacSystemFont, "Segoe UI",
  Roboto, "Helvetica Neue", Arial, "Noto Sans", 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;
}
```

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(--font-family-sans-serif);  
}  
a {  
  color: var(--blue);  
}
```

Copy

Breakpoint variables

While we originally included breakpoints in our CSS variables (e.g., `--breakpoint-md`), **these are not supported in media queries**, but they can still be used *within* rulesets in media queries. These breakpoint variables remain in the compiled CSS for backward compatibility given they can be utilized by JavaScript. [Learn more in the spec](#).

Here's an example of **what's not supported**:

```
@media (min-width: var(--breakpoint-sm)) {  
  ...  
}
```

Copy

And here's an example of **what is supported**:

```
@media (min-width: 768px) {  
  .custom-element {  
    color: var(--primary);  
  }  
}
```

Copy

Webpack

Search...

Getting started

[Introduction](#)

[Download](#)

[Contents](#)

[Browsers & devices](#)

[JavaScript](#)

[Theming](#)

[Build tools](#)

[Webpack](#)

[Accessibility](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Installing Bootstrap

[Install bootstrap](#) as a Node.js module using npm.

Importing JavaScript

Import [Bootstrap's JavaScript](#) by adding this line to your app's entry point (usually `index.js` or `app.js`):

```
import 'bootstrap';
```

Copy

Alternatively, you may **import plugins individually** as needed:

```
import 'bootstrap/js/dist/util';
import 'bootstrap/js/dist/alert';
...
```

Copy

Bootstrap depends on [jQuery](#) and [Popper](#), which are specified in the `peerDependencies` property; this means that you will have to make sure to add both of them to your `package.json` using `npm install --save jquery popper.js`.

Importing Styles

Importing Precompiled Sass

To enjoy the full potential of Bootstrap and customize it to your needs, use the source files as a part of your project's bundling process.

First, create your own `_custom.scss` and use it to override the [built-in custom variables](#). Then, use your main Sass file to import your custom variables, followed by Bootstrap:

```
@import "custom";
@import "~bootstrap/scss/bootstrap";
```

Copy

For Bootstrap to compile, make sure you install and use the required loaders: [sass-loader](#), [postcss-loader](#) with [Autoprefixer](#). With minimal setup, your webpack config should include this rule or similar:

Copy

```
...
{
  test: /\.scss$/,
  use: [
    {
      loader: 'style-loader', // inject CSS to page
    },
    {
      loader: 'css-loader', // translates CSS into CommonJS modules
    },
    {
      loader: 'postcss-loader', // Run postcss actions
      options: {
        plugins: function () { // postcss plugins, can be exported to
          postcss.config.js
        }
      }
    },
    {
      loader: 'sass-loader' // compiles Sass to CSS
    }
  ],
}
...
```

Importing Compiled CSS

Alternatively, you may use Bootstrap's ready-to-use CSS by simply adding this line to your project's entry point:

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

Copy

In this case you may use your existing rule for `css` without any special modifications to webpack config, except you don't need `sass-loader` just `style-loader` and `css-loader`.

```
...
module: {
  rules: [
    {
      test: /\.css$/,
      use: ['style-loader', 'css-loader']
    }
  ]
}
...
```

Copy

Code

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Reboot](#)

[Typography](#)

[Code](#)

[Images](#)

[Tables](#)

[Figures](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Inline code

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

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

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

Copy

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. You may optionally add the `.pre-scrollable` class, which will set a max-height of 340px and provide a y-axis scrollbar.

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

```
<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>
```

Copy

Variables

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

```
y = mx + b
```

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

Copy

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 + ,
```

Copy

To switch directories, type `<kbd>cd</kbd>` followed by the name of the directory.
`
`
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.

Copy

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

Figures

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

[Getting started](#)

[Layout](#)

[Content](#)

[Reboot](#)

[Typography](#)

[Code](#)

[Images](#)

[Tables](#)

[Figures](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)

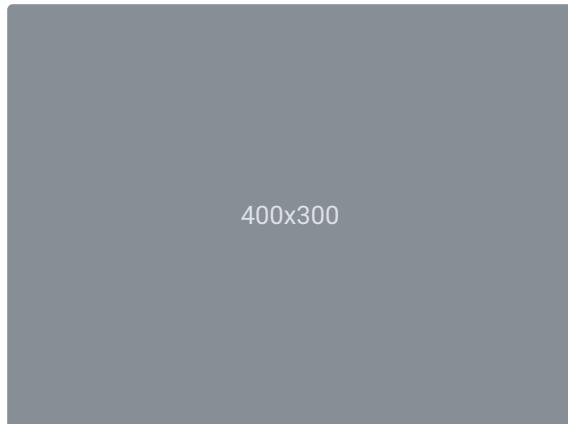


Adobe Creative Cloud for Teams starting at \$33.99 per month.

ads via Carbon

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 `` to make it responsive.

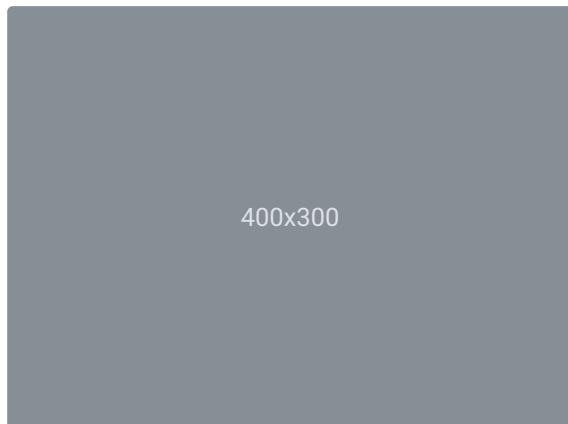


A caption for the above image.

Copy

```
<figure class="figure">
  
  <figcaption class="figure-caption">A caption for the above image.</figcaption>
</figure>
```

Aligning the figure's caption is easy with our [text utilities](#).



A caption for the above image.

Copy

```
<figure class="figure">
  
  <figcaption class="figure-caption text-right">A caption for the above image.
</figcaption>
</figure>
```

Images

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Reboot](#)

[Typography](#)

[Code](#)

[Images](#)

[Tables](#)

[Figures](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Build and deploy your app
with \$100 in credit.

ads via Carbon

Responsive images

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



Responsive image

```

```

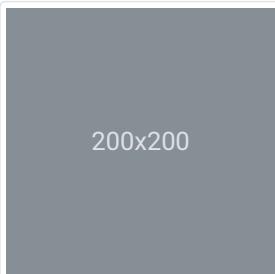
Copy

SVG images and IE 10

In Internet Explorer 10, SVG images with `.img-fluid` are disproportionately sized. To fix this, add `width: 100% \9;` where necessary. This fix improperly sizes other image formats, so Bootstrap doesn't apply it automatically.

Image thumbnails

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



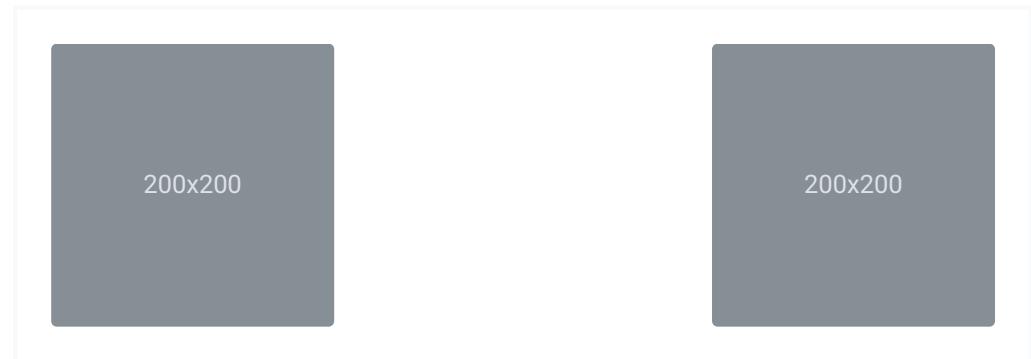
```

```

Copy

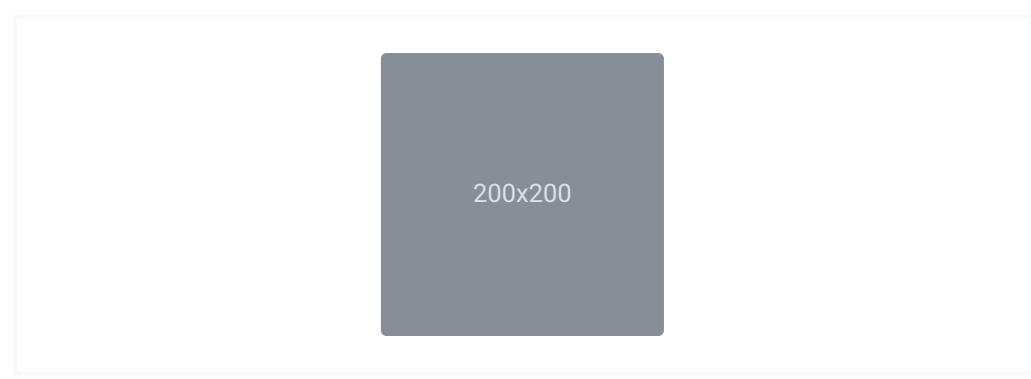
Aligning images

Align images with the [helper float classes](#) or [text alignment classes](#). **block**-level images can be centered using [the .mx-auto margin utility class](#).



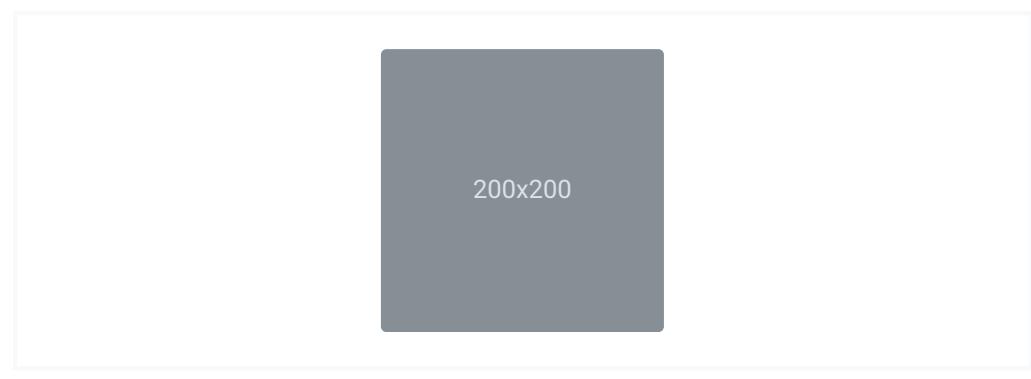
```
</pre>
```

Copy



```
</pre>
```

Copy



```
<div class="text-center"></div></pre>
```

Copy

Picture

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

```
<picture><source srcset="..." type="image/svg+xml"></picture></pre>
```

Copy

Reboot

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Reboot](#)

[Typography](#)

[Code](#)

[Images](#)

[Tables](#)

[Figures](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10
free Adobe Stock images.

ads via Carbon

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 `rems` instead of `ems` 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 `rems` for `margins`.
- Keep declarations of `font`-related properties to a minimum, using `inherit` whenever possible.

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.
- The `<body>` also sets a global `font-family`, `line-height`, and `text-align`. 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

The default web fonts (Helvetica Neue, Helvetica, and Arial) have been dropped in Bootstrap 4 and replaced with a “native font stack” for optimum text rendering on every device and OS. Read more about [native font stacks in this Smashing Magazine article](#).

Copy

```

$font-family-sans-serif:
  // Safari for macOS and iOS (San Francisco)
  -apple-system,
  // Chrome < 56 for macOS (San Francisco)
  BlinkMacSystemFont,
  // Windows
  "Segoe UI",
  // Android
  Roboto,
  // Basic web fallback
  "Helvetica Neue", Arial,
  // Linux
  "Noto Sans",
  // Sans serif fallback
  sans-serif,
  // Emoji fonts
  "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji"
!default;

```

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

Lists

All lists—``, ``, and `<dl>`—have their `margin-top` removed and a `margin-bottom: 1rem`. Nested lists have no `margin-bottom`.

- Lorem ipsum dolor sit amet
- Consectetur adipiscing elit
- Integer molestie lorem at massa
- Facilisis in pretium nisl aliquet
- Nulla volutpat aliquam velit
 - Phasellus iaculis neque
 - Purus sodales ultricies
 - Vestibulum laoreet porttitor sem
 - Ac tristique libero volutpat at
- Faucibus porta lacus fringilla vel
- Aenean sit amet erat nunc

- Eget porttitor lorem
 1. Lorem ipsum dolor sit amet
 2. Consectetur adipiscing elit
 3. Integer molestie lorem at massa
 4. Facilisis in pretium nisl aliquet
 5. Nulla volutpat aliquam velit
 6. Faucibus porta lacus fringilla vel
 7. Aenean sit amet erat nunc
 8. Eget porttitor lorem

For simpler styling, clear hierarchy, and better spacing, description lists have updated `margins`. `<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.

Euismod

Vestibulum id ligula porta felis euismod semper eget lacinia odio sem.

Donec id elit non mi porta gravida at eget metus.

Malesuada porta

Etiam porta sem malesuada magna mollis euismod.

Preformatted text

The `<pre>` element is reset to remove its `margin-top` and use `rem` units for its `margin-bottom`.

```
.example-element {
  margin-bottom: 1rem;
}
```

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](#).

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

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

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.

Example legend

Example input

Example select ▾

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

Example textarea

Example date

Example time

Example output 100

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

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

Copy

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 `
`.

Twitter, Inc.
1355 Market St, Suite 900
San Francisco, CA 94103
P: (Phone) (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.

`Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere
 erat a ante.`

`Someone famous in Source Title`

Inline elements

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

`Nulla attr (attribute) vitae elit libero, a pharetra augue.`

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.

- ▶ Some details
 - ▼ Even more details
- 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. While `[hidden]` isn't natively supported by IE10, the explicit declaration in our CSS gets around that problem.

`<input type="text" hidden>`

Copy

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.

Tables

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Reboot](#)

[Typography](#)

[Code](#)

[Images](#)

[Tables](#)

[Figures](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Examples

Due to the widespread use of tables across third-party widgets like calendars and date pickers, we've designed our tables to be **opt-in**. Just add the base class `.table` to any `<table>`, then extend with custom styles or our various included modifier classes.

Using the most basic table markup, here's how `.table`-based tables look in Bootstrap. **All table styles are inherited in Bootstrap 4**, meaning any nested tables will be styled in the same manner as the parent.

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

Copy

```

<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>Larry</td>
      <td>the Bird</td>
      <td>@twitter</td>
    </tr>
  </tbody>
</table>

```

You can also invert the colors—with light text on dark backgrounds—with `.table-dark`.

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

Copy

```

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

```

Table head options

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

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

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

Copy

```

<table class="table">
  <thead class="thead-dark">
    <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>

<table class="table">
  <thead class="thead-light">
    <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>

```

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

Copy

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

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

Copy

```

<table class="table table-striped table-dark">
  <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>

```

Bordered table

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

Copy

```

<table class="table table-bordered">
  <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>

```

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

Copy

```

<table class="table table-bordered table-dark">
  <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>

```

Borderless table

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

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

Copy

```

<table class="table table-borderless">
  <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>

```

.table-borderless can also be used on dark tables.

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

Copy

```

<table class="table table-borderless table-dark">
  <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>

```

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

[Copy](#)

```

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

```

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

Copy

```

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

```

Small table

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

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

Copy

```

<table class="table table-sm">
  <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>

```

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

```

<table class="table table-sm table-dark">
  <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>

```

Copy

Contextual classes

Use contextual classes to color table rows or individual cells.

Class	Heading	Heading
Active	Cell	Cell
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 rows -->
<tr class="table-active">...</tr>

<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-active">...</td>

  <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>
```

Copy

Regular table background variants are not available with the dark table, however, you may use [text or background utilities](#) to achieve similar styles.

#	Heading	Heading

#	Heading	Heading
1	Cell	Cell
2	Cell	Cell
3	Cell	Cell
4	Cell	Cell
5	Cell	Cell
6	Cell	Cell
7	Cell	Cell
8	Cell	Cell
9	Cell	Cell

Copy

```
<!-- On rows -->
<tr class="bg-primary">...</tr>
<tr class="bg-success">...</tr>
<tr class="bg-warning">...</tr>
<tr class="bg-danger">...</tr>
<tr class="bg-info">...</tr>

<!-- On cells (`td` or `th`) -->
<tr>
  <td class="bg-primary">...</td>
  <td class="bg-success">...</td>
  <td class="bg-warning">...</td>
  <td class="bg-danger">...</td>
  <td class="bg-info">...</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 `.sr-only` class.

Create responsive tables by wrapping any `.table` with `.table-responsive{-sm|-md|-lg|-xl}`, making the table scroll horizontally at each `max-width` breakpoint of up to (but not including) 576px, 768px, 992px, and 1120px, respectively.

Note that since browsers do not currently support [range context queries](#), 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 for these comparisons.

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.

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

```
<table class="table">
  <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>
```

Copy

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}`.

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 Head

| # | Heading |
|---|---------|---------|---------|---------|---------|---------|---------|---------|
| 1 | Cell |
| 2 | Cell |
| 3 | Cell |

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

Copy

Breakpoint specific

Use `.table-responsive{-sm|-md|-lg|-xl}` 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 |
|---|---------|---------|---------|---------|---------|---------|---------|---------|
| 1 | Cell |
| 2 | Cell |
| 3 | Cell |

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

Copy

| # | Heading |
|---|---------|---------|---------|---------|---------|---------|---------|---------|
| 1 | Cell |
| 2 | Cell |
| 3 | Cell |

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

Copy

| # | Heading | Heac |
|---|---------|---------|---------|---------|---------|---------|---------|---------|------|
| 1 | Cell | Cell |
| 2 | Cell | Cell |
| 3 | Cell | Cell |

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

Copy

| # | Heading | Heac |
|---|---------|---------|---------|---------|---------|---------|---------|---------|------|
| 1 | Cell | Cell |
| 2 | Cell | Cell |
| 3 | Cell | Cell |

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

Copy

Typography

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Reboot](#)

[Typography](#)

[Code](#)

[Images](#)

[Tables](#)

[Figures](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Power-up the lifetime value of your app users with rewarded video ads

ads via Carbon

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 assume the browser 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` and apply link underlines only on `:hover`.
- Use `$body-bg` to set a `background-color` on the `<body>` (#ffff 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
<code><h1></h1></code>	h1. Bootstrap heading
<code><h2></h2></code>	h2. Bootstrap heading
<code><h3></h3></code>	h3. Bootstrap heading
<code><h4></h4></code>	h4. Bootstrap heading
<code><h5></h5></code>	h5. Bootstrap heading
<code><h6></h6></code>	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>
```

Copy

.**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

```
<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>
```

Copy

Customizing headings

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

Fancy display heading With faded secondary text

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

Copy

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. Keep in mind these headings are not responsive by default, but it's possible to enable [responsive font sizes](#).

Display 1

Display 2

Display 3

Display 4

Copy

```
<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>
```

Lead

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

Copy

Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Duis mollis, est non commodo luctus.

```
<p class="lead">
    Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Duis
    mollis, est non commodo luctus.
</p>
```

Inline text elements

Styling for common inline HTML5 elements.

Copy

You can use the mark tag to `highlight` 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.

```
<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>
```

`.mark` and `.small` classes are also available to apply the same styles as `<mark>` and `<small>` while avoiding any unwanted semantic implications that the tags would bring.

While not shown above, feel free to use `` and `<i>` in HTML5. `` 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, 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.

```
attr (attribute)  
HTML (HYPertext Markup Language)
```

Copy
`<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 (HyperText Markup Language)` as the quote.

```
 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere  
erat a ante.
```

Copy
`<blockquote class="blockquote">
 <p class="mb-0">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
 Integer posuere erat a ante.</p>
</blockquote>`

Naming a source

Add a `<footer class="blockquote-footer">` for identifying the source. Wrap the name of the source work in `<cite>`.

```
 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere  
erat a ante.  
— Someone famous in Source Title
```

Copy
`<blockquote class="blockquote">
 <p class="mb-0">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
 Integer posuere erat a ante.</p>
 <footer class="blockquote-footer">Someone famous in <cite title="Source
Title">Source Title</cite></footer>
</blockquote>`

Alignment

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

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere
erat a ante.

— Someone famous in *Source Title*

Copy

```
<blockquote class="blockquote text-center">  
  <p class="mb-0">Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
  Integer posuere erat a ante.</p>  
  <footer class="blockquote-footer">Someone famous in <cite title="Source  
  Title">Source Title</cite></footer>  
</blockquote>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere
erat a ante.

— Someone famous in *Source Title*

Copy

```
<blockquote class="blockquote text-right">  
  <p class="mb-0">Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
  Integer posuere erat a ante.</p>  
  <footer class="blockquote-footer">Someone famous in <cite title="Source  
  Title">Source Title</cite></footer>  
</blockquote>
```

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.

 Lorem ipsum dolor sit amet
 Consectetur adipiscing elit
 Integer molestie lorem at massa
 Facilisis in pretium nisl aliquet
 Nulla volutpat aliquam velit

- Phasellus iaculis neque
- Purus sodales ultricies
- Vestibulum laoreet porttitor sem
- Ac tristique libero volutpat at

 Faucibus porta lacus fringilla vel
 Aenean sit amet erat nunc
 Eget porttitor lorem

Copy

```

<ul class="list-unstyled">
  <li>Lorem ipsum dolor sit amet</li>
  <li>Consectetur adipiscing elit</li>
  <li>Integer molestie lorem at massa</li>
  <li>Facilisis in pretium nisl aliquet</li>
  <li>Nulla volutpat aliquam velit
    <ul>
      <li>Phasellus iaculis neque</li>
      <li>Purus sodales ultricies</li>
      <li>Vestibulum laoreet porttitor sem</li>
      <li>Ac tristique libero volutpat at</li>
    </ul>
  </li>
  <li>Faucibus porta lacus fringilla vel</li>
  <li>Aenean sit amet erat nunc</li>
  <li>Eget porttitor lorem</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`.

Copy
 Lorem ipsum Phasellus iaculis Nulla volutpat

```

<ul class="list-inline">
  <li class="list-inline-item">Lorem ipsum</li>
  <li class="list-inline-item">Phasellus iaculis</li>
  <li class="list-inline-item">Nulla volutpat</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.	
Euismod	Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.	
	Donec id elit non mi porta gravida at eget metus.	
Malesuada porta	Etiam porta sem malesuada magna mollis euismod.	
Truncated term is ...	Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.	
Nesting	Nested definition	Aenean posuere, tortor sed cursus feugiat, nunc augue blandit nunc.
	list	

```

<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">Euismod</dt>
  <dd class="col-sm-9">
    <p>Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.</p>
    <p>Donec id elit non mi porta gravida at eget metus.</p>
  </dd>

  <dt class="col-sm-3">Malesuada porta</dt>
  <dd class="col-sm-9">Etiam porta sem malesuada magna mollis euismod.</dd>

  <dt class="col-sm-3 text-truncate">Truncated term is truncated</dt>
  <dd class="col-sm-9">Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.</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">Aenean posuere, tortor sed cursus feugiat, nunc augue blandit nunc.</dd>
    </dl>
  </dd>
</dl>

```

Responsive font sizes

Bootstrap v4.3 ships with the option to enable responsive font sizes, allowing text to scale more naturally across device and viewport sizes. [RFS \(Responsive font sizes\)](#) can be enabled by changing the `$enable-responsive-font-sizes` Sass variable to `true` and recompiling Bootstrap.

To support [RFS \(Responsive font sizes\)](#), we use a Sass mixin to replace our normal `font-size` properties. Responsive font sizes will be compiled into `calc()` functions with a mix of `rem` and viewport units to enable the responsive scaling behavior. More about [RFS \(Responsive font sizes\)](#) and its configuration can be found on its [GitHub repository](#).

Alerts

Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages.

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Examples

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!

Copy

```
<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 `.sr-only` class.

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.

```

<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.

[Copy](#)

```

<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>

```

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.
- If you're building our JavaScript from source, it [requires util.js](#). The compiled version includes this.

- Add a dismiss button and the `.alert-dismissible` class, which adds extra padding to the right of the alert and positions the `.close` button.
- On the dismiss button, add the `data-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. ✖

```
<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="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
</div>
```

Copy

JavaScript behavior

Triggers

Enable dismissal of an alert via JavaScript:

```
$('.alert').alert()
```

Copy

Or with `data` attributes on a button **within the alert**, as demonstrated above:

```
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
  <span aria-hidden="true">&times;</span>
</button>
```

Copy

Note that closing an alert will remove it from the DOM.

Methods

Method	Description
<code>\$(().alert())</code>	Makes an alert listen for click events on descendant elements which have the <code>data-dismiss="alert"</code> attribute. (Not necessary when using the data-api's auto-initialization.)
<code>\$(().alert('close'))</code>	Closes an alert by removing it from the DOM. If the <code>.fade</code> and <code>.show</code> classes are present on the element, the alert will fade out before it is removed.
<code>\$(().alert('dispose'))</code>	Destroys an element's alert.

```
$('.alert').alert('close')
```

Copy

Events

Bootstrap's alert plugin exposes a few events for hooking into alert functionality.

Event	Description
<code>close.bs.alert</code>	This event fires immediately when the <code>close</code> instance method is called.
<code>closed.bs.alert</code>	This event is fired when the alert has been closed (will wait for CSS transitions to complete).

```
$('#myAlert').on('closed.bs.alert', function () {  
  // do something...  
})
```

Copy

Badges

Search...

Getting started

Layout

Content

Components

Alerts

Badge

Breadcrumb

Buttons

Button group

Card

Carousel

Collapse

Dropdowns

Forms

Input group

Jumbotron

List group

Media object

Modal

Navs

Navbar

Pagination

Popovers

Progress

Scrollspy

Spinners

Toasts

Tooltips

Utilities

Extend

Migration

About



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Example

Badges scale to match the size of the immediate parent element by using relative font sizing and `em` units.

Example heading New

Copy

```
<h1>Example heading <span class="badge badge-secondary">New</span></h1>
<h2>Example heading <span class="badge badge-secondary">New</span></h2>
<h3>Example heading <span class="badge badge-secondary">New</span></h3>
<h4>Example heading <span class="badge badge-secondary">New</span></h4>
<h5>Example heading <span class="badge badge-secondary">New</span></h5>
<h6>Example heading <span class="badge badge-secondary">New</span></h6>
```

Badges can be used as part of links or buttons to provide a counter.

Notifications 4

Copy

```
<button type="button" class="btn btn-primary">
  Notifications <span class="badge badge-light">4</span>
</button>
```

Note that depending on how they are used, badges may be confusing for users of screen readers and similar assistive technologies. While the styling of badges provides a visual cue as to their purpose, these users will simply be presented with the content of the badge. Depending on the specific situation, these badges may seem like random additional words or numbers at the end of a sentence, link, or button.

Unless the context is clear (as with the “Notifications” example, where it is understood that the “4” is the number of notifications), consider including additional context with a visually hidden piece of additional text.

Profile 9

Copy

```
<button type="button" class="btn btn-primary">  
  Profile <span class="badge badge-light">9</span>  
  <span class="sr-only">unread messages</span>  
</button>
```

Contextual variations

Add any of the below mentioned modifier classes to change the appearance of a badge.

Primary Secondary Success Danger Warning Info Light Dark

Copy

```
<span class="badge badge-primary">Primary</span>  
<span class="badge badge-secondary">Secondary</span>  
<span class="badge badge-success">Success</span>  
<span class="badge badge-danger">Danger</span>  
<span class="badge badge-warning">Warning</span>  
<span class="badge badge-info">Info</span>  
<span class="badge badge-light">Light</span>  
<span class="badge badge-dark">Dark</span>
```

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 `.sr-only` class.

Pill badges

Use the `.badge-pill` modifier class to make badges more rounded (with a larger `border-radius` and additional horizontal `padding`). Useful if you miss the badges from v3.

Primary Secondary Success Danger Warning Info Light Dark

Copy

```
<span class="badge badge-pill badge-primary">Primary</span>  
<span class="badge badge-pill badge-secondary">Secondary</span>  
<span class="badge badge-pill badge-success">Success</span>  
<span class="badge badge-pill badge-danger">Danger</span>  
<span class="badge badge-pill badge-warning">Warning</span>  
<span class="badge badge-pill badge-info">Info</span>  
<span class="badge badge-pill badge-light">Light</span>  
<span class="badge badge-pill badge-dark">Dark</span>
```

Links

Using the contextual `.badge-*` classes on an `<a>` element quickly provide *actionable* badges with hover and focus states.

Primary Secondary Success Danger Warning Info Light Dark

Copy

```
<a href="#" class="badge badge-primary">Primary</a>
<a href="#" class="badge badge-secondary">Secondary</a>
<a href="#" class="badge badge-success">Success</a>
<a href="#" class="badge badge-danger">Danger</a>
<a href="#" class="badge badge-warning">Warning</a>
<a href="#" class="badge badge-info">Info</a>
<a href="#" class="badge badge-light">Light</a>
<a href="#" class="badge badge-dark">Dark</a>
```

Breadcrumb

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

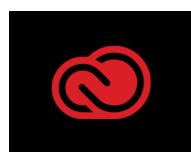
[Toolips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Adobe Creative Cloud for
Teams starting at \$33.99
per month.

ads via Carbon

Example

Home

[Home](#) / Library

[Home](#) / [Library](#) / Data

```
<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>
```

Copy

Changing the separator

Separtors are automatically added in CSS through `:before` and `content`. They can be changed by changing `$breadcrumb-divider`. The `quote` function is needed to generate the quotes around a string, so if you want `>` as separator, you can use this:

```
$breadcrumb-divider: quote(">");
```

Copy

It's also possible to use a **base64 embedded SVG icon**:

Copy

```
$breadcrumb-divider:  
url(data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdm  
ciIHdpZHRoPSI4IiBoZWlnaHQ9IjgiPjxwYXR0IGQ9Ik0yLjUgMEwxIDEuNSAzLjUgNCAxIDYuNSAyLj  
UgOGw0LTQtNC00eiIgZmlsbD0iY3VycmVudENvbG9yIi8+PC9zdmc+);
```

The separator can be removed by setting `$breadcrumb-divider` to `none`:

Copy

```
$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](#).

Button group

Search...

Getting started

Layout

Content

Components

Alerts

Badge

Breadcrumb

Buttons

Button group

Card

Carousel

Collapse

Dropdowns

Forms

Input group

Jumbotron

List group

Media object

Modal

Navs

Navbar

Pagination

Popovers

Progress

Scrollspy

Spinners

Toasts

Tooltips

Utilities

Extend

Migration

About



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Basic example

Wrap a series of buttons with `.btn` in `.btn-group`. Add on optional JavaScript radio and checkbox style behavior with [our buttons plugin](#).

Left Middle Right

```
<div class="btn-group" role="group" aria-label="Basic example">
  <button type="button" class="btn btn-secondary">Left</button>
  <button type="button" class="btn btn-secondary">Middle</button>
  <button type="button" class="btn btn-secondary">Right</button>
</div>
```

Copy

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.

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.

1 2 3 4 5 6 7 8

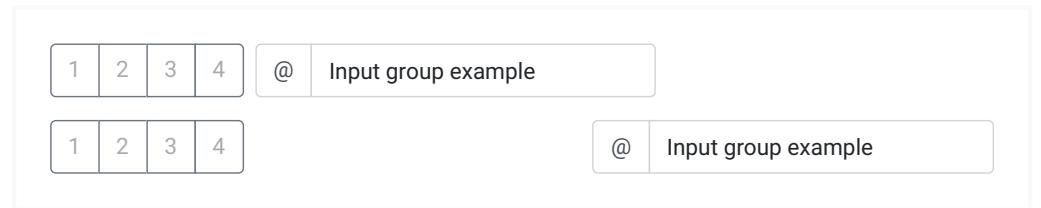
Copy

```

<div class="btn-toolbar" role="toolbar" aria-label="Toolbar with button groups">
  <div class="btn-group mr-2" role="group" aria-label="First group">
    <button type="button" class="btn btn-secondary">1</button>
    <button type="button" class="btn btn-secondary">2</button>
    <button type="button" class="btn btn-secondary">3</button>
    <button type="button" class="btn btn-secondary">4</button>
  </div>
  <div class="btn-group mr-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-secondary">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.



```

<div class="btn-toolbar mb-3" role="toolbar" aria-label="Toolbar with button groups">
  <div class="btn-group mr-2" role="group" aria-label="First group">
    <button type="button" class="btn btn-secondary">1</button>
    <button type="button" class="btn btn-secondary">2</button>
    <button type="button" class="btn btn-secondary">3</button>
    <button type="button" class="btn btn-secondary">4</button>
  </div>
  <div class="input-group">
    <div class="input-group-prepend">
      <div class="input-group-text" id="btnGroupAddon">@</div>
    </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-secondary">1</button>
    <button type="button" class="btn btn-secondary">2</button>
    <button type="button" class="btn btn-secondary">3</button>
    <button type="button" class="btn btn-secondary">4</button>
  </div>
  <div class="input-group">
    <div class="input-group-prepend">
      <div class="input-group-text" id="btnGroupAddon2">@</div>
    </div>
    <input type="text" class="form-control" placeholder="Input group example" aria-label="Input group example" aria-describedby="btnGroupAddon2">
  </div>
</div>

```

[Copy](#)

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>
```

Copy

Nesting

Place a `.btn-group` within another `.btn-group` when you want dropdown menus mixed with a series of buttons.



```
<div class="btn-group" role="group" aria-label="Button group with nested
dropdown">
  <button type="button" class="btn btn-secondary">1</button>
  <button type="button" class="btn btn-secondary">2</button>

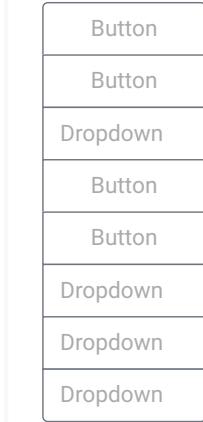
  <div class="btn-group" role="group">
    <button id="btnGroupDrop1" type="button" class="btn btn-secondary dropdown-
toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
      Dropdown
    </button>
    <div class="dropdown-menu" aria-labelledby="btnGroupDrop1">
      <a class="dropdown-item" href="#">Dropdown link</a>
      <a class="dropdown-item" href="#">Dropdown link</a>
    </div>
  </div>
</div>
```

Copy

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>
```

Copy

Buttons

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Toolips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Grow your app revenue
with quality demand from
Facebook's global
advertisers
ads via Carbon

Examples

Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control.

Primary Secondary Success Danger Warning Info Light Dark

Link

Copy

```
<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 `.sr-only` 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.

```
Link Button Input Submit Reset
```

Copy

```
<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.

```
Primary Secondary Success Danger Warning Info Light Dark
```

Copy

```
<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>
```

Sizes

Fancy larger or smaller buttons? Add `.btn-lg` or `.btn-sm` for additional sizes.

```
Large button Large button
```

Copy

```
<button type="button" class="btn btn-primary btn-lg">Large button</button>
<button type="button" class="btn btn-secondary btn-lg">Large button</button>
```

```
Small button Small button
```

Copy

```
<button type="button" class="btn btn-primary btn-sm">Small button</button>
<button type="button" class="btn btn-secondary btn-sm">Small button</button>
```

Create block level buttons—those that span the full width of a parent—by adding `.btn-block`.

```
Block level button
```

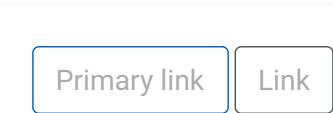
```
Block level button
```

Copy

```
<button type="button" class="btn btn-primary btn-lg btn-block">Block level  
button</button>  
<button type="button" class="btn btn-secondary btn-lg btn-block">Block level  
button</button>
```

Active state

Buttons will appear pressed (with a darker background, darker border, and inset shadow) when active. There's no need to add a class to `<button>`s as they use a **pseudo-class**. However, you can still force the same active appearance with `.active` (and include the `aria-pressed="true"` attribute) should you need to replicate the state programmatically.



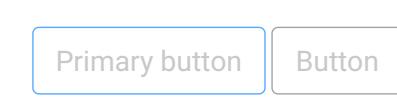
Primary link Link

Copy

```
<a href="#" class="btn btn-primary btn-lg active" role="button" aria-  
pressed="true">Primary link</a>  
<a href="#" class="btn btn-secondary btn-lg active" role="button" aria-  
pressed="true">Link</a>
```

Disabled state

Make buttons look inactive by adding the `disabled` boolean attribute to any `<button>` element.



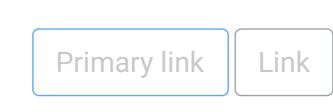
Primary button Button

Copy

```
<button type="button" class="btn btn-primary btn-lg" disabled>Primary  
button</button>  
<button type="button" class="btn btn-secondary btn-lg" 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. In browsers which support that property, you won't see the disabled cursor at all.
- Disabled buttons should include the `aria-disabled="true"` attribute to indicate the state of the element to assistive technologies.



Primary link Link

Copy

```
<a href="#" class="btn btn-primary btn-lg disabled" tabindex="-1" role="button"  
aria-disabled="true">Primary link</a>  
<a href="#" class="btn btn-secondary btn-lg disabled" tabindex="-1"  
role="button" aria-disabled="true">Link</a>
```

Link functionality caveat

The `.disabled` class uses `pointer-events: none` to try to disable the link functionality of `<a>`s, but that CSS property is not yet standardized. 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, add a `tabindex="-1"` attribute on these links (to prevent them from receiving keyboard focus) and use custom JavaScript to disable their functionality.

Button plugin

Do more with buttons. Control button states or create groups of buttons for more components like toolbars.

Toggle states

Add `data-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 the `<button>`.

Single toggle

```
<button type="button" class="btn btn-primary" data-toggle="button" aria-pressed="false">  
  Single toggle  
</button>
```

Copy

Checkbox and radio buttons

Bootstrap's `.button` styles can be applied to other elements, such as `<label>`s, to provide checkbox or radio style button toggling. Add `data-toggle="buttons"` to a `.btn-group` containing those modified buttons to enable their toggling behavior via JavaScript and add `.btn-group-toggle` to style the `<input>`s within your buttons. **Note that you can create single input-powered buttons or groups of them.**

The checked state for these buttons is **only updated via `click` event** on the button. If you use another method to update the input—e.g., with `<input type="reset">` or by manually applying the input's `checked` property—you'll need to toggle `.active` on the `<label>` manually.

Note that pre-checked buttons require you to manually add the `.active` class to the input's `<label>`.

Checked

```
<div class="btn-group-toggle" data-toggle="buttons">  
  <label class="btn btn-secondary active">  
    <input type="checkbox" checked> Checked  
  </label>  
</div>
```

Copy

Active Radio Radio

Copy

```
<div class="btn-group btn-group-toggle" data-toggle="buttons">
  <label class="btn btn-secondary active">
    <input type="radio" name="options" id="option1" checked> Active
  </label>
  <label class="btn btn-secondary">
    <input type="radio" name="options" id="option2"> Radio
  </label>
  <label class="btn btn-secondary">
    <input type="radio" name="options" id="option3"> Radio
  </label>
</div>
```

Methods

Method	Description
<code>\$(()).button('toggle')</code>	Toggles push state. Gives the button the appearance that it has been activated.
<code>\$(()).button('dispose')</code>	Destroys an element's button.

Cards

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Adobe Creative Cloud for Teams starting at \$33.99 per month.

ads via Carbon

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](#).



Image cap

Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

[Go somewhere](#)

```
<div class="card" style="width: 18rem;">
  
  <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>
```

Copy

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.

Copy

```
<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](#)

Copy

```
<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.

Image cap

Some quick example text to build on the card title and make up the bulk of the card's content.

Copy

```
<div class="card" style="width: 18rem;">
  
  <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.

Cras justo odio

Dapibus ac facilisis in

Vestibulum at eros

Copy

```
<div class="card" style="width: 18rem;">
  <ul class="list-group list-group-flush">
    <li class="list-group-item">Cras justo odio</li>
    <li class="list-group-item">Dapibus ac facilisis in</li>
    <li class="list-group-item">Vestibulum at eros</li>
  </ul>
</div>
```

Featured

Cras justo odio

Dapibus ac facilisis in

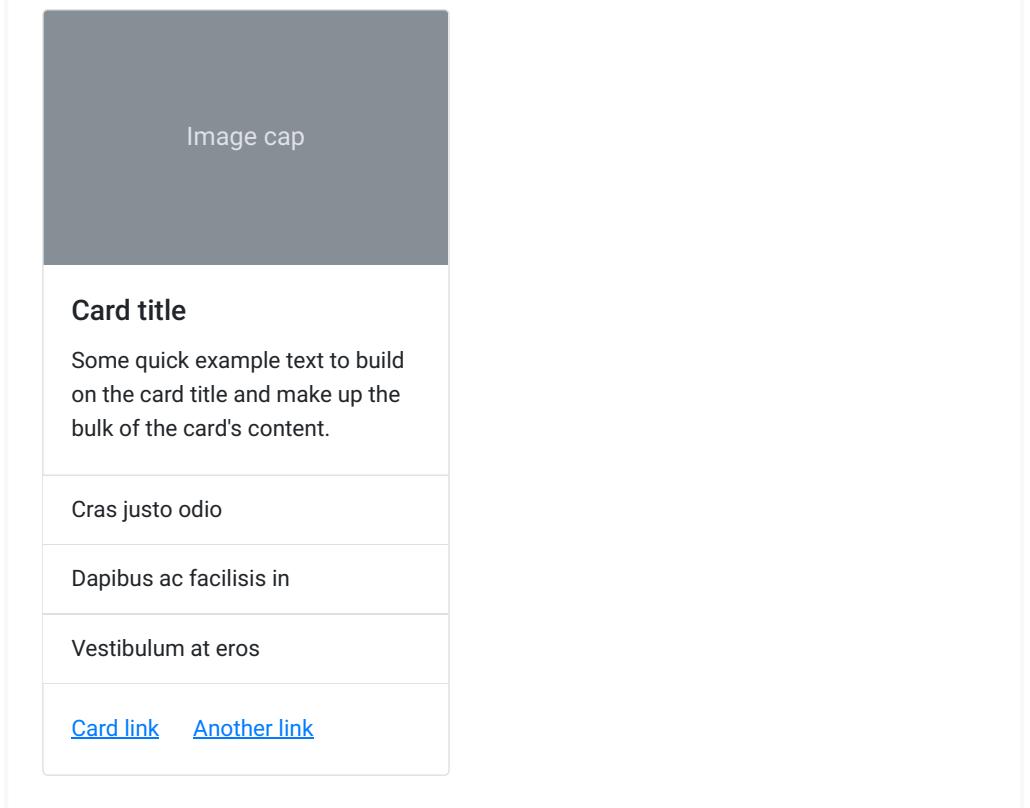
Vestibulum at eros

Copy

```
<div class="card" style="width: 18rem;">
  <div class="card-header">
    Featured
  </div>
  <ul class="list-group list-group-flush">
    <li class="list-group-item">Cras justo odio</li>
    <li class="list-group-item">Dapibus ac facilisis in</li>
    <li class="list-group-item">Vestibulum at eros</li>
  </ul>
</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.

Cras justo odio

Dapibus ac facilisis in

Vestibulum at eros

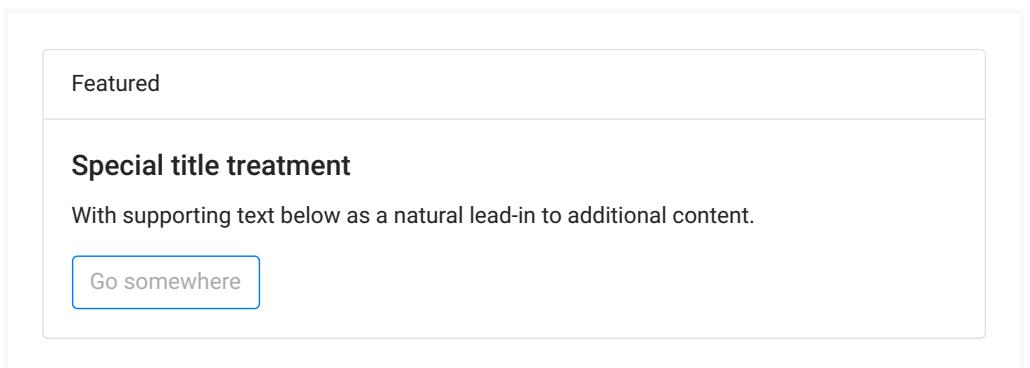
[Card link](#) [Another link](#)

Copy

```
<div class="card" style="width: 18rem;">
  
  <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">Cras justo odio</li>
    <li class="list-group-item">Dapibus ac facilisis in</li>
    <li class="list-group-item">Vestibulum at eros</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.

[Go somewhere](#)

Copy

```
<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.

Go somewhere

```
<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>
```

Copy

Quote

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.

 — Someone famous in *Source Title*

```
<div class="card">
  <div class="card-header">
    Quote
  </div>
  <div class="card-body">
    <blockquote class="blockquote mb-0">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.</p>
      <footer class="blockquote-footer">Someone famous in <cite title="Source Title">Source Title</cite></footer>
    </blockquote>
  </div>
</div>
```

Copy

Featured

Special title treatment

With supporting text below as a natural lead-in to additional content.

[Go somewhere](#)

2 days ago

Copy

```
<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.

[Go somewhere](#)

Special title treatment

With supporting text below as a natural lead-in to additional content.

[Go somewhere](#)

Copy

```

<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.

[Button](#)

Card title

With supporting text below as a natural lead-in to additional content.

[Button](#)

Copy

```

<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.

[Go somewhere](#)

Copy

```
<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.

[Go somewhere](#)

Special title treatment

With supporting text below as a natural lead-in to additional content.

[Go somewhere](#)

Special title treatment

With supporting text below as a natural lead-in to additional content.

[Go somewhere](#)

Copy

```

<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-right" 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](#).

The screenshot shows a card component with a tabbed header. The tabs are labeled "Active", "Link", and "Disabled". The "Active" tab is currently selected. The main content area contains the heading "Special title treatment" and the text "With supporting text below as a natural lead-in to additional content.". A blue-outlined button labeled "Go somewhere" is located at the bottom right of the content area.

```

<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" 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" href="#" tabindex="-1" aria-disabled="true">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.

[Go somewhere](#)

Copy

```
<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" href="#" tabindex="-1" aria-
disabled="true">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.

Image cap

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

Image cap

Copy

```
<div class="card mb-3">
  
  <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>
  
</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

Card image

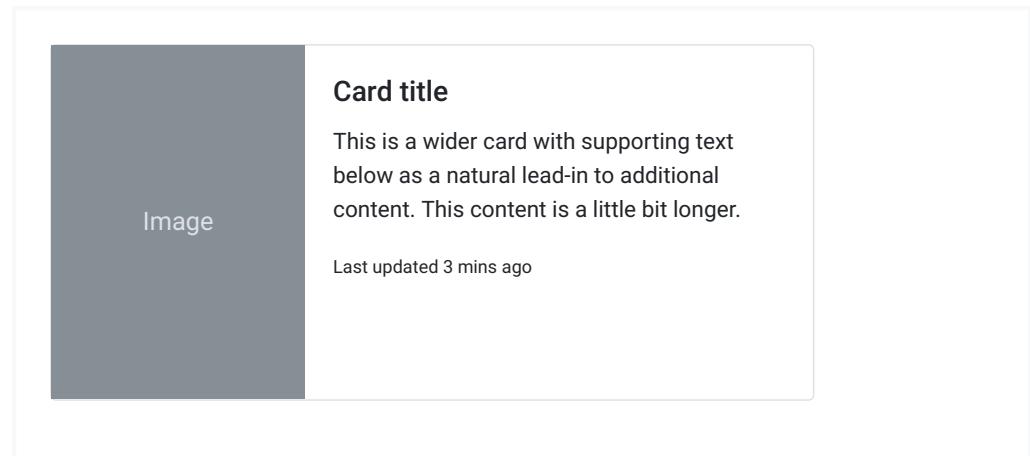
Copy

```
<div class="card bg-dark text-white">
  
  <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 `.no-gutters` and use `.col-md-*` classes to make the card horizontal at the `md` breakpoint. Further adjustments may be needed depending on your card content.



```
<div class="card mb-3" style="max-width: 540px;">
  <div class="row no-gutters">
    <div class="col-md-4">
      
    </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>
```

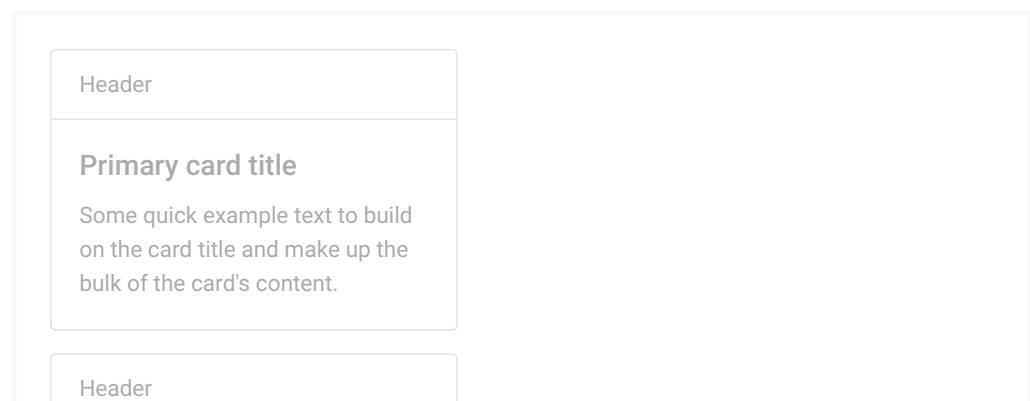
Copy

Card styles

Cards include various options for customizing their backgrounds, borders, and color.

Background and color

Use [text and background utilities](#) to change the appearance of a card.



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.

Copy

```
<div class="card text-white 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-white 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-white 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-white 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-white 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-white 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 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-white 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 `.sr-only` 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.

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.

Copy

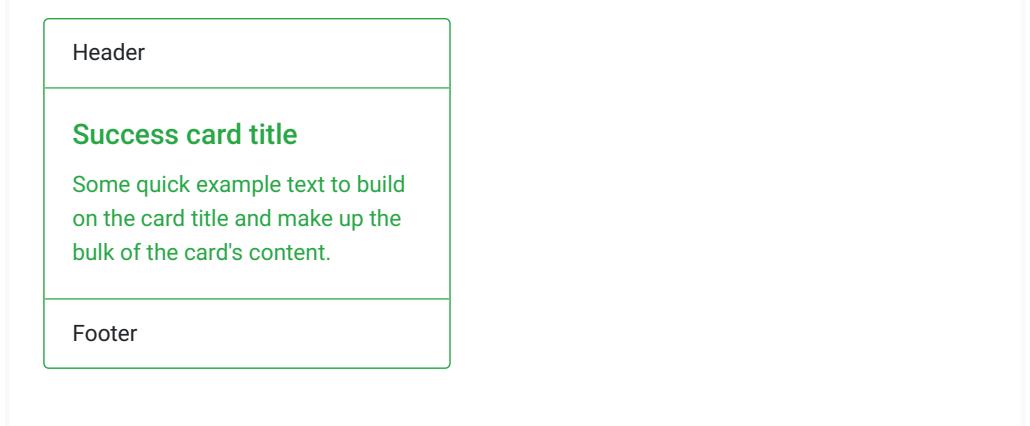
```

<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 text-warning">
    <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 text-info">
    <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`.



Copy

```
<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

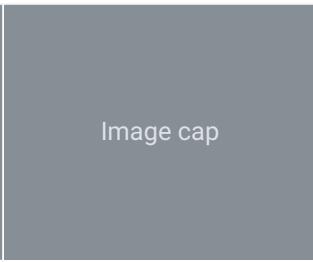
Copy

```

<div class="card-group">
  <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>
  </div>
  <div class="card">
    
    <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">
    
    <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.	 Card title This card has supporting text below as a natural lead-in to additional content.	 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	Last updated 3 mins ago	Last updated 3 mins ago

Copy

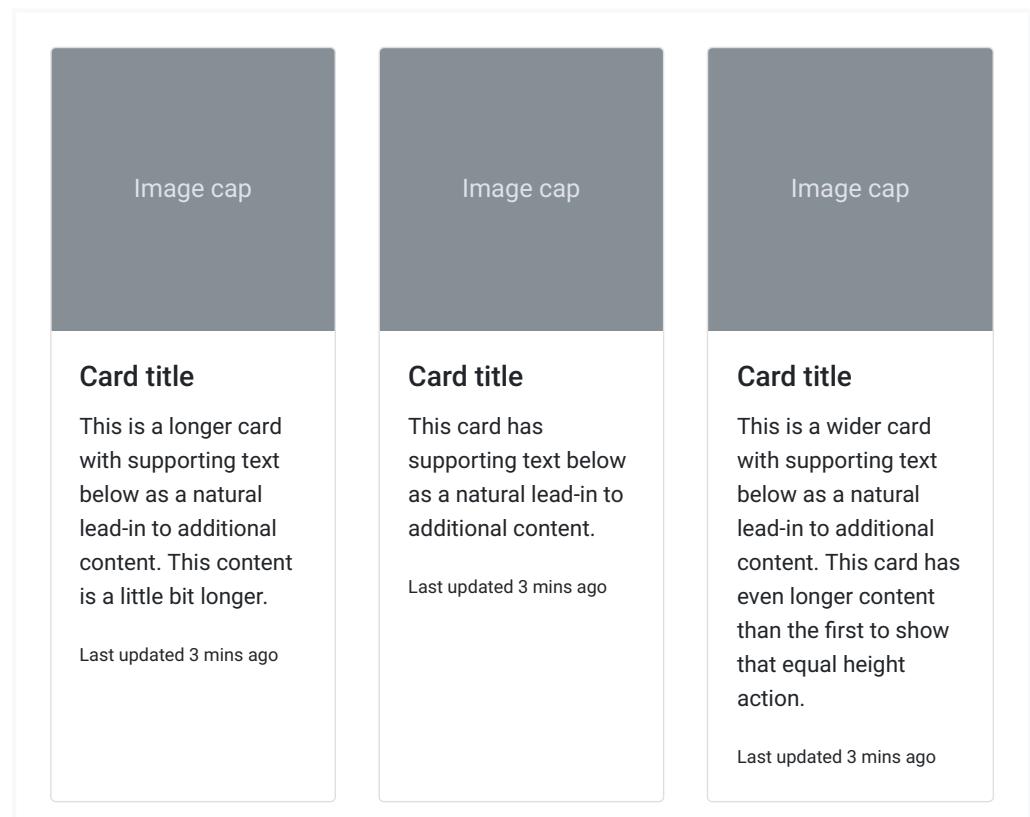
```

<div class="card-group">
  <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>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 3 mins ago</small>
    </div>
  </div>
  <div class="card">
    
    <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">
    
    <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>

```

Card decks

Need a set of equal width and height cards that aren't attached to one another? Use card decks.



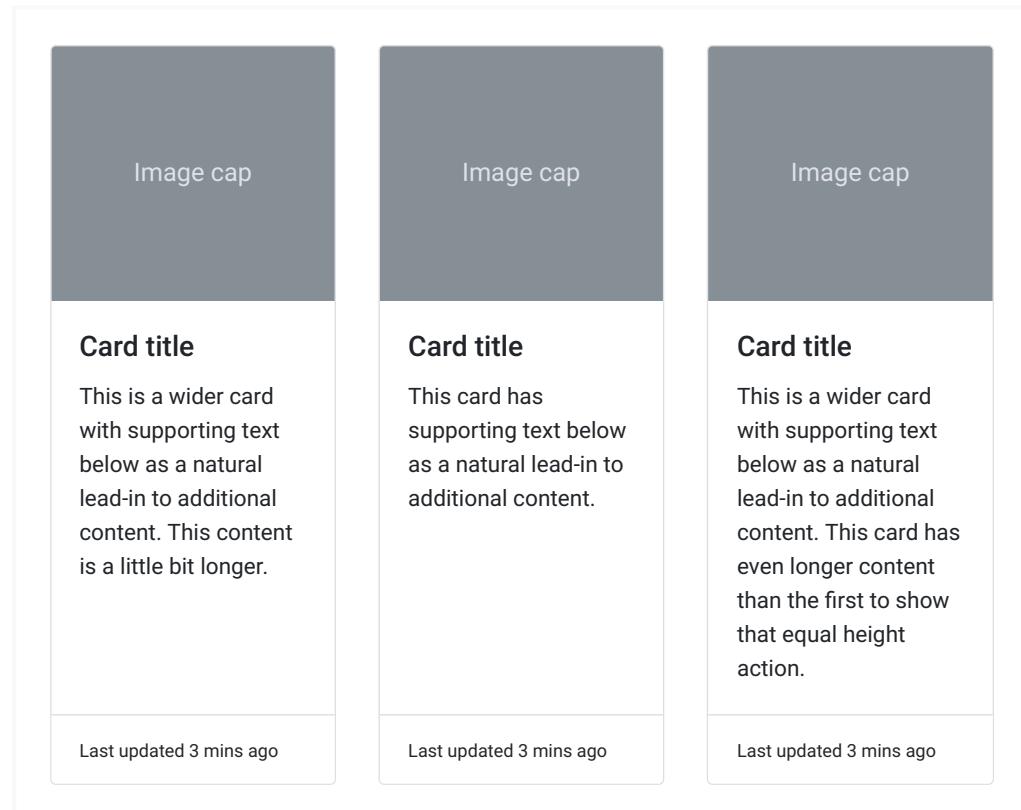
The image shows a row of three cards, each with a dark gray header and a white body. The headers are labeled "Image cap". The bodies contain the following content:

- 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.
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

[Copy](#)

```
<div class="card-deck">
  <div class="card">
    
    <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>
      <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 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">
    
    <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>
```

Just like with card groups, card footers in decks will automatically line up.

[Copy](#)

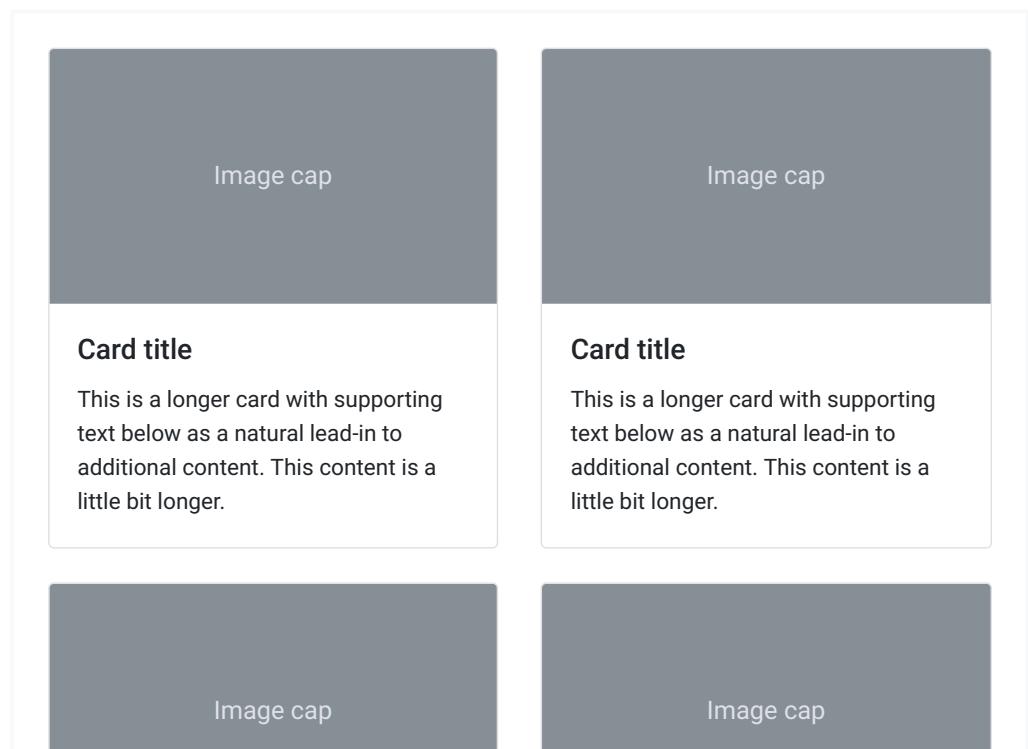
```

<div class="card-deck">
  <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>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 3 mins ago</small>
    </div>
  </div>
  <div class="card">
    
    <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">
    
    <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.

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.

```
<div class="row row-cols-1 row-cols-md-2">
  <div class="col mb-4">
    <div class="card">
      
      <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 mb-4">
    <div class="card">
      
      <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 mb-4">
    <div class="card">
      
      <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 mb-4">
    <div class="card">
      
      <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>
```

Copy

Change it to `.row-cols-3` and you'll see the fourth card wrap.

Image cap

Image cap

Image cap

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.

Image cap

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.

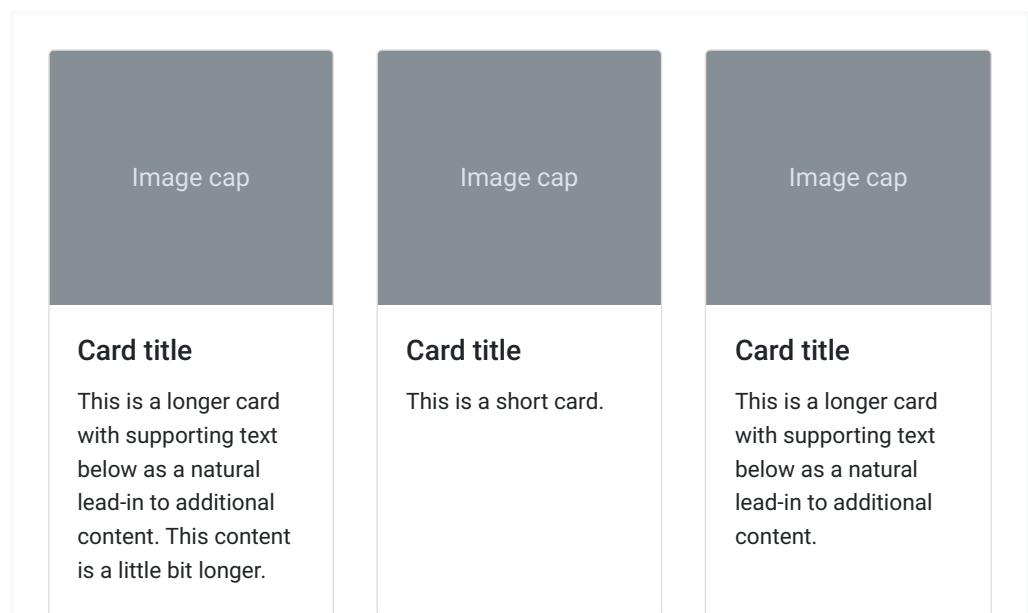
Copy

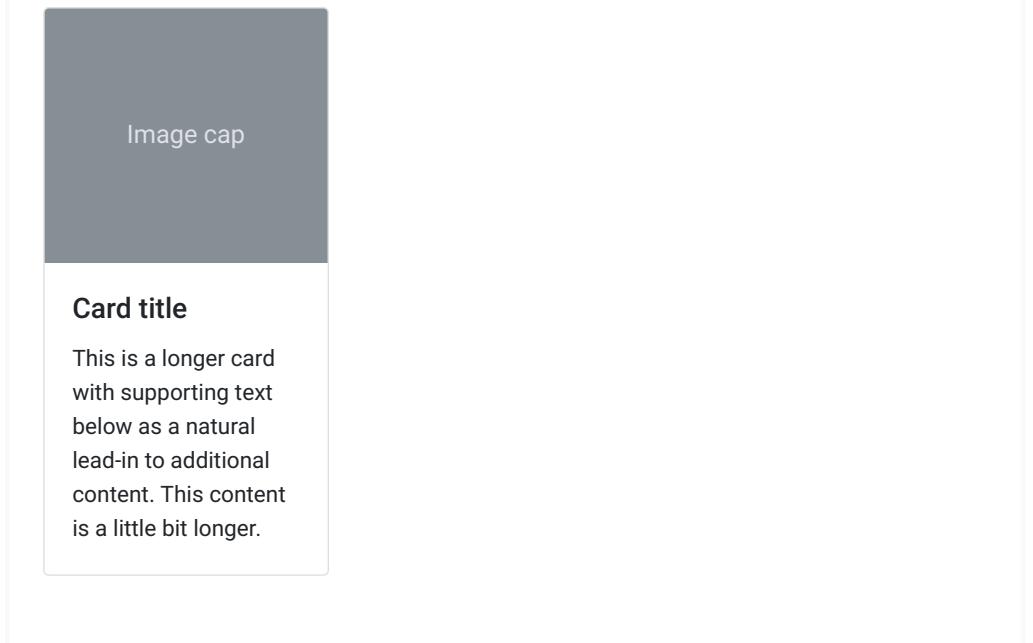
```

<div class="row row-cols-1 row-cols-md-3">
  <div class="col mb-4">
    <div class="card">
      
      <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 mb-4">
    <div class="card">
      
      <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 mb-4">
    <div class="card">
      
      <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>

```

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.





Copy

```
<div class="row row-cols-1 row-cols-md-3">
  <div class="col mb-4">
    <div class="card h-100">
      
      <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 mb-4">
    <div class="card h-100">
      
      <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 mb-4">
    <div class="card h-100">
      
      <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 mb-4">
    <div class="card h-100">
      
      <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>
```

Card columns

Cards can be organized into [Masonry](#)-like columns with just CSS by wrapping them in `.card-columns`. Cards are built with CSS `column` properties instead of flexbox for easier alignment. Cards are ordered from top to bottom and left to right.

Heads up! Your mileage with card columns may vary. To prevent cards breaking across columns, we must set them to `display: inline-block` as `column-break-inside: avoid` isn't a bulletproof solution yet.

The image shows a grid of cards arranged in three columns. The first column contains two cards: one with an image cap and another with a title, subtitle, and supporting text. The second column contains two cards: one with an image cap and another with a title, subtitle, and a quote. The third column contains two cards: one with an image and another with a large block of text and a quote.

Card Type	Content
Image Cap	Image cap
Card title that wraps to a new line	<p>Card title that wraps to a new line</p> <p>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>
Image Cap	Image cap
Card title	<p>Card title</p> <p>This card has supporting text below as a natural lead-in to additional content.</p> <p>Last updated 3 mins ago</p>
Card image	Card image
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.	<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.</p> <p>— Someone famous in <i>Source Title</i></p>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.	<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.</p> <p>— Someone famous in <i>Source Title</i></p>
Card title	<p>Card title</p> <p>This card has a regular title and short paragraph of text below it.</p> <p>Last updated 3 mins ago</p>
Card title	<p>Card title</p> <p>This is another card with title and supporting text below. This card has some additional content to make it slightly taller overall.</p> <p>Last updated 3 mins ago</p>

Copy

```
<div class="card-columns">
  <div class="card">
    
    <div class="card-body">
      <h5 class="card-title">Card title that wraps to a new line</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 class="card p-3">
    <blockquote class="blockquote mb-0 card-body">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.</p>
      <footer class="blockquote-footer">
        <small class="text-muted">
          Someone famous in <cite title="Source Title">Source Title</cite>
        </small>
      </footer>
    </blockquote>
  </div>
  <div class="card">
    
    <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 bg-primary text-white text-center p-3">
    <blockquote class="blockquote mb-0">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat.</p>
      <footer class="blockquote-footer text-white">
        <small>
          Someone famous in <cite title="Source Title">Source Title</cite>
        </small>
      </footer>
    </blockquote>
  </div>
  <div class="card text-center">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This card has a regular title and short paragraph of text below it.</p>
      <p class="card-text"><small class="text-muted">Last updated 3 mins ago</small></p>
    </div>
  </div>
  <div class="card">
    
  </div>
  <div class="card p-3 text-right">
    <blockquote class="blockquote mb-0">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.</p>
      <footer class="blockquote-footer">
        <small class="text-muted">
          Someone famous in <cite title="Source Title">Source Title</cite>
        </small>
      </footer>
    </blockquote>
  </div>
  <div class="card">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This is another card with title and supporting text below. This card has some additional content to make it slightly taller overall.</p>
    </div>
  </div>
</div>
```

```
<p class="card-text"><small class="text-muted">Last updated 3 mins  
ago</small></p>  
</div>  
</div>  
</div>
```

Card columns can also be extended and customized with some additional code. Shown below is an extension of the `.card-columns` class using the same CSS we use—CSS columns—to generate a set of responsive tiers for changing the number of columns.

```
.card-columns {  
  @include media-breakpoint-only(lg) {  
    column-count: 4;  
  }  
  @include media-breakpoint-only(xl) {  
    column-count: 5;  
  }  
}
```

Copy

Carousel

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

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.

Lastly, if you're building our JavaScript from source, it [requires util.js](#).

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-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.

Third slide

```
<div id="carouselExampleSlidesOnly" class="carousel slide" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      
    </div>
    <div class="carousel-item">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
```

Copy

With controls

Adding in the previous and next controls:

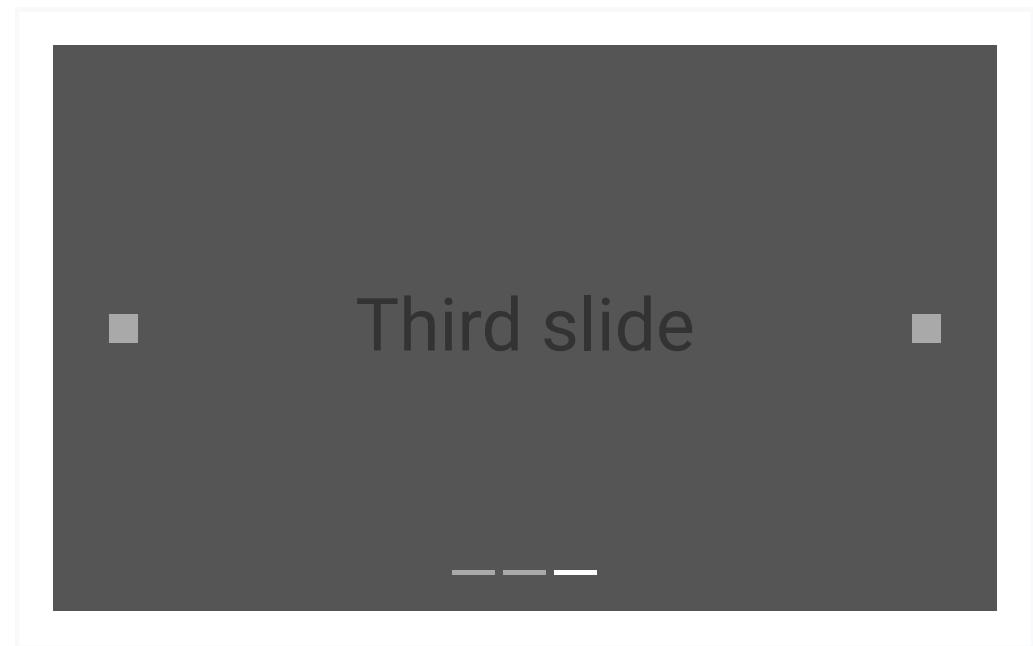
Third slide

Copy

```
<div id="carouselExampleControls" class="carousel slide" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      
    </div>
    <div class="carousel-item">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
  <a class="carousel-control-prev" href="#carouselExampleControls" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleControls" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

With indicators

You can also add the indicators to the carousel, alongside the controls, too.



Copy

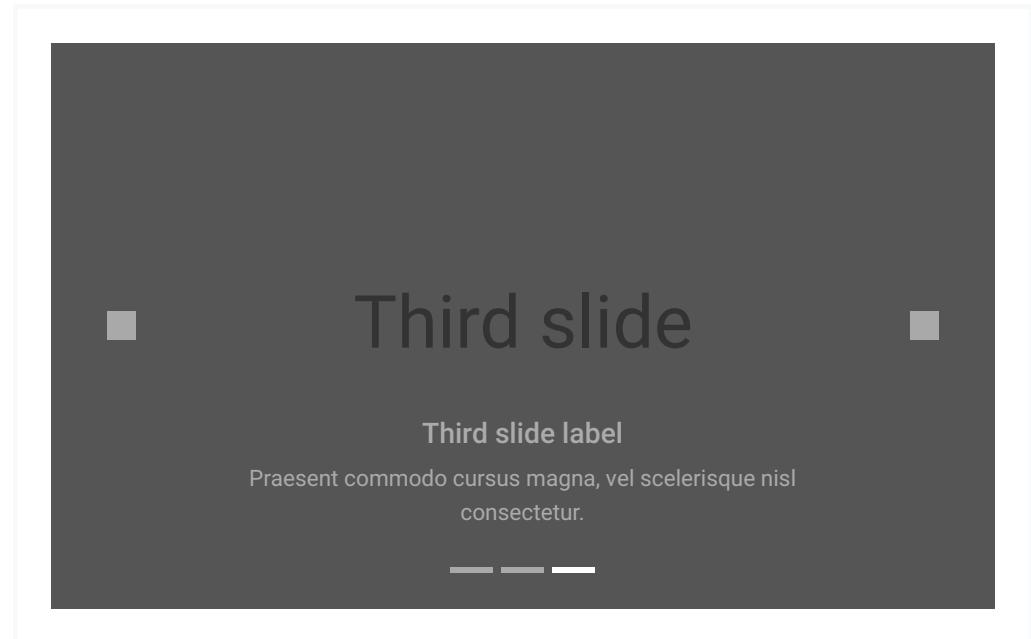
```

<div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">
  <ol class="carousel-indicators">
    <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>
    <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
    <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
  </ol>
  <div class="carousel-inner">
    <div class="carousel-item active">
      
    </div>
    <div class="carousel-item">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
  <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</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`.



[Copy](#)

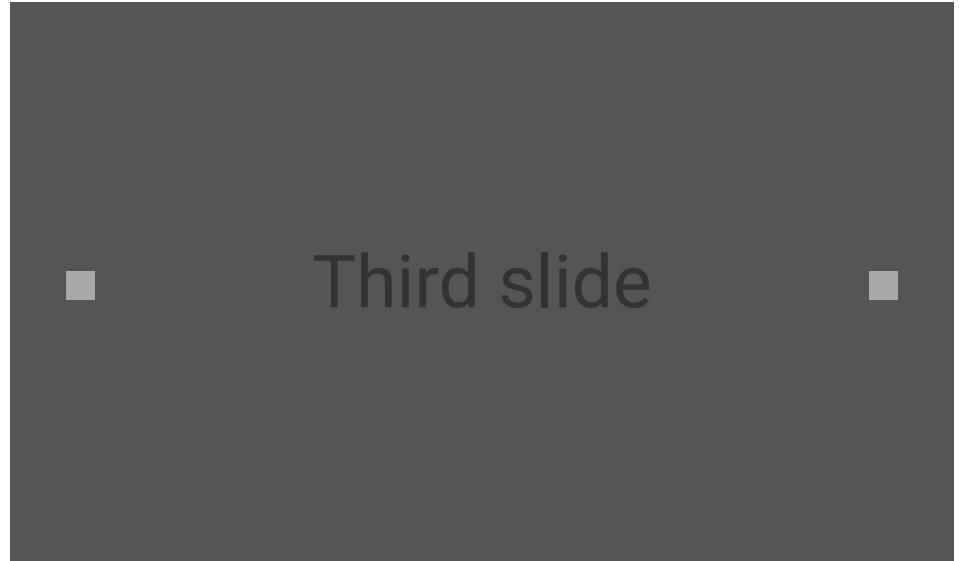
```

<div id="carouselExampleCaptions" class="carousel slide" data-ride="carousel">
  <ol class="carousel-indicators">
    <li data-target="#carouselExampleCaptions" data-slide-to="0" class="active"></li>
    <li data-target="#carouselExampleCaptions" data-slide-to="1"></li>
    <li data-target="#carouselExampleCaptions" data-slide-to="2"></li>
  </ol>
  <div class="carousel-inner">
    <div class="carousel-item active">
      
      <div class="carousel-caption d-none d-md-block">
        <h5>First slide label</h5>
        <p>Nulla vitae elit libero, a pharetra augue mollis interdum.</p>
      </div>
    </div>
    <div class="carousel-item">
      
      <div class="carousel-caption d-none d-md-block">
        <h5>Second slide label</h5>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
      </div>
    </div>
    <div class="carousel-item">
      
      <div class="carousel-caption d-none d-md-block">
        <h5>Third slide label</h5>
        <p>Praesent commodo cursus magna, vel scelerisque nisl consectetur.</p>
      </div>
    </div>
    <a class="carousel-control-prev" href="#carouselExampleCaptions" role="button" data-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="sr-only">Previous</span>
    </a>
    <a class="carousel-control-next" href="#carouselExampleCaptions" role="button" data-slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="sr-only">Next</span>
    </a>
  </div>

```

Crossfade

Add `.carousel-fade` to your carousel to animate slides with a fade transition instead of a slide.



```
<div id="carouselExampleFade" class="carousel slide carousel-fade" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      
    </div>
    <div class="carousel-item">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
  <a class="carousel-control-prev" href="#carouselExampleFade" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleFade" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

Individual .carousel-item interval

Add `data-interval=""` to a `.carousel-item` to change the amount of time to delay between automatically cycling to the next item.



Copy

```

<div id="carouselExampleInterval" class="carousel slide" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active" data-interval="10000">
      
    </div>
    <div class="carousel-item" data-interval="2000">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
  <a class="carousel-control-prev" href="#carouselExampleInterval" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleInterval" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>

```

Usage

Via data attributes

Use data attributes to easily control the position of the carousel. `data-slide` accepts the keywords `prev` or `next`, which alters the slide position relative to its current position. Alternatively, use `data-slide-to` to pass a raw slide index to the carousel `data-slide-to="2"`, which shifts the slide position to a particular index beginning with `0`.

The `data-ride="carousel"` attribute is used to mark a carousel as animating starting at page load. If you don't use `data-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:

```
$('.carousel').carousel()
```

[Copy](#)

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-interval=""`.

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.

Name	Type	Default	Description
pause	string boolean	"hover"	If set to "hover", pauses the cycling of the carousel on <code>mouseenter</code> and resumes the cycling of the carousel on <code>mouseleave</code> . If set to <code>false</code> , hovering over the carousel won't pause it. On touch-enabled devices, when set to "hover", cycling will pause on <code>touchend</code> (once the user finished interacting with the carousel) for two intervals, before automatically resuming. Note that this is in addition to the above mouse behavior.
ride	string	false	Autoplays the carousel after the user manually cycles the first item. If "carousel", autoplays the carousel on load.
wrap	boolean	true	Whether the carousel should cycle continuously or have hard stops.
touch	boolean	true	Whether the carousel should support left/right swipe interactions on touchscreen devices.

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.](#)

`.carousel(options)`

Initializes the carousel with an optional options `object` and starts cycling through items.

```
$('.carousel').carousel({
  interval: 2000
})
```

Copy

`.carousel('cycle')`

Cycles through the carousel items from left to right.

`.carousel('pause')`

Stops the carousel from cycling through items.

`.carousel(number)`

Cycles the carousel to a particular frame (0 based, similar to an array). **Returns to the caller before the target item has been shown** (i.e. before the `slid.bs.carousel` event occurs).

`.carousel('prev')`

Cycles to the previous item. **Returns to the caller before the previous item has been shown** (i.e. before the `slid.bs.carousel` event occurs).

`.carousel('next')`

Cycles to the next item. **Returns to the caller before the next item has been shown** (i.e. before the `slid.bs.carousel` event occurs).

`.carousel('dispose')`

Destroys an element's carousel.

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	This event fires immediately when the <code>slide</code> instance method is invoked.
slid.bs.carousel	This event is fired when the carousel has completed its slide transition.

```
$('#myCarousel').on('slide.bs.carousel', function () {  
  // do something...  
})
```

Copy

Change transition duration

The transition duration of `.carousel-item` can be changed with the `$carousel-transition` 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`).

Collapse

Search...

Getting started

[Layout](#)

[Content](#)

Components

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

Utilities

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

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

You can use a link with the `href` attribute, or a button with the `data-target` attribute. In both cases, the `data-toggle="collapse"` is required.

[Link with href](#) Button with data-target

Copy

```
<p>
  <a class="btn btn-primary" data-toggle="collapse" href="#collapseExample"
  role="button" aria-expanded="false" aria-controls="collapseExample">
    Link with href
  </a>
  <button class="btn btn-primary" type="button" data-toggle="collapse" data-
  target="#collapseExample" aria-expanded="false" aria-controls="collapseExample">
    Button with data-target
  </button>
</p>
<div class="collapse" id="collapseExample">
  <div class="card card-body">
    Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry
    richardson ad squid. Nihil anim keffiyeh helvetica, craft beer labore wes
    anderson cred nesciunt sapiente ea proident.
  </div>
</div>
```

Multiple targets

A `<button>` or `<a>` can show and hide multiple elements by referencing them with a JQuery selector in its `href` or `data-target` attribute. Multiple `<button>` or `<a>` can show and hide an element if they each reference it with their `href` or `data-target` attribute

[Toggle first element](#) [Toggle second element](#) [Toggle both elements](#)

Copy

```
<p>
  <a class="btn btn-primary" data-toggle="collapse"
    href="#multiCollapseExample1" role="button" aria-expanded="false" aria-
    controls="multiCollapseExample1">Toggle first element</a>
  <button class="btn btn-primary" type="button" data-toggle="collapse" data-
    target="#multiCollapseExample2" aria-expanded="false" aria-
    controls="multiCollapseExample2">Toggle second element</button>
  <button class="btn btn-primary" type="button" data-toggle="collapse" data-
    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">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus
        terry richardson ad squid. Nihil anim keffiyeh helvetica, craft beer labore wes
        anderson cred nesciunt sapiente ea proident.
      </div>
    </div>
  </div>
  <div class="col">
    <div class="collapse multi-collapse" id="multiCollapseExample2">
      <div class="card card-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus
        terry richardson ad squid. Nihil anim keffiyeh helvetica, craft beer labore wes
        anderson cred nesciunt sapiente ea proident.
      </div>
    </div>
  </div>
</div>
```

Accordion example

Using the `card` component, you can extend the default collapse behavior to create an accordion. To properly achieve the accordion style, be sure to use `.accordion` as a wrapper.

Collapsible Group Item #1

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

Collapsible Group Item #2

Collapsible Group Item #3

Copy

```

<div class="accordion" id="accordionExample">
  <div class="card">
    <div class="card-header" id="headingOne">
      <h2 class="mb-0">
        <button class="btn btn-link btn-block text-left" type="button" data-
        toggle="collapse" data-target="#collapseOne" aria-expanded="true" aria-
        controls="collapseOne">
          Collapsible Group Item #1
        </button>
      </h2>
    </div>

    <div id="collapseOne" class="collapse show" aria-labelledby="headingOne"
    data-parent="#accordionExample">
      <div class="card-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus
        terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard
        dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon
        tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda
        shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson
        cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo.
        Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt
        you probably haven't heard of them accusamus labore sustainable VHS.
      </div>
    </div>
  </div>
  <div class="card">
    <div class="card-header" id="headingTwo">
      <h2 class="mb-0">
        <button class="btn btn-link btn-block text-left collapsed" type="button"
        data-toggle="collapse" data-target="#collapseTwo" aria-expanded="false" aria-
        controls="collapseTwo">
          Collapsible Group Item #2
        </button>
      </h2>
    </div>
    <div id="collapseTwo" class="collapse" aria-labelledby="headingTwo" data-
    parent="#accordionExample">
      <div class="card-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus
        terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard
        dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon
        tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda
        shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson
        cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo.
        Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt
        you probably haven't heard of them accusamus labore sustainable VHS.
      </div>
    </div>
  </div>
  <div class="card">
    <div class="card-header" id="headingThree">
      <h2 class="mb-0">
        <button class="btn btn-link btn-block text-left collapsed" type="button"
        data-toggle="collapse" data-target="#collapseThree" aria-expanded="false" aria-
        controls="collapseThree">
          Collapsible Group Item #3
        </button>
      </h2>
    </div>
    <div id="collapseThree" class="collapse" aria-labelledby="headingThree"
    data-parent="#accordionExample">
      <div class="card-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus
        terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard
        dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon
        tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda
        shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson
        cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo.
        Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt
      </div>
    </div>
  </div>
</div>

```

```
you probably haven't heard of them accusamus labore sustainable VHS.  
      </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-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 keyboard interactions described in the [WAI-ARIA Authoring Practices 1.1 accordion pattern](#) - you will need to include these yourself with custom JavaScript.

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-toggle="collapse"` and a `data-target` to the element to automatically assign control of one or more collapsible elements. The `data-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-parent="#selector"`. Refer to the demo to see this in action.

Via JavaScript

Enable manually with:

```
$('.collapse').collapse()
```

Copy

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-parent=""`.

Name	Type	Default	Description

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 <code>card</code> 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.](#)

`.collapse(options)`

Activates your content as a collapsible element. Accepts an optional options `object`.

```
$('#myCollapsible').collapse({
  toggle: false
})
```

Copy

`.collapse('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).

`.collapse('show')`

Shows a collapsible element. **Returns to the caller before the collapsible element has actually been shown** (i.e. before the `shown.bs.collapse` event occurs).

`.collapse('hide')`

Hides a collapsible element. **Returns to the caller before the collapsible element has actually been hidden** (i.e. before the `hidden.bs.collapse` event occurs).

`.collapse('dispose')`

Destroys an element's collapse.

Events

Bootstrap's collapse class exposes a few events for hooking into collapse functionality.

Event Type	Description
<code>show.bs.collapse</code>	This event fires immediately when the <code>show</code> instance method is called.
<code>shown.bs.collapse</code>	This event is fired when a collapse element has been made visible to the user (will wait for CSS transitions to complete).
<code>hide.bs.collapse</code>	This event is fired immediately when the <code>hide</code> method has been called.
<code>hidden.bs.collapse</code>	This event is fired when a collapse element has been hidden from the user (will wait for CSS transitions to complete).

```
$('#myCollapsible').on('hidden.bs.collapse', function () {  
    // do something...  
})
```

Copy

Dropdowns

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Adobe Creative Cloud for
Teams starting at \$33.99
per month.

ads via Carbon

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.js](#), 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.js. Popper.js isn't used to position dropdowns in navbars though as dynamic positioning isn't required.

If you're building our JavaScript from source, it [requires util.js](#).

Accessibility

The [WAI \(Web Accessibility Initiative\) ARIA \(Accessible Rich Internet Applications\)](#) standard defines an actual [role="menu" widget](#), but this is specific to application-like menus which trigger actions or functions. ARIA (Accessible Rich Internet Applications) 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 ARIA (Accessible Rich Internet Applications) 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.

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:

Dropdown button

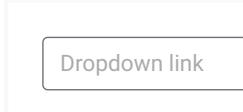
Copy

```

<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle" type="button"
  id="dropdownMenuButton" data-toggle="dropdown" aria-haspopup="true" aria-
  expanded="false">
    Dropdown button
  </button>
  <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
    <a class="dropdown-item" href="#">Action</a>
    <a class="dropdown-item" href="#">Another action</a>
    <a class="dropdown-item" href="#">Something else here</a>
  </div>
</div>

```

And with `<a>` elements:



Dropdown link

Copy

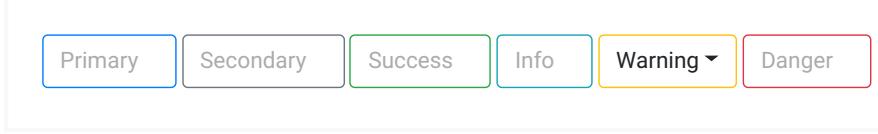
```

<div class="dropdown">
  <a class="btn btn-secondary dropdown-toggle" href="#" role="button"
  id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-
  expanded="false">
    Dropdown link
  </a>

  <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
    <a class="dropdown-item" href="#">Action</a>
    <a class="dropdown-item" href="#">Another action</a>
    <a class="dropdown-item" href="#">Something else here</a>
  </div>
</div>

```

The best part is you can do this with any button variant, too:



Primary Secondary Success Info Warning ▾ Danger

Copy

```

<!-- Example single danger button --&gt;
&lt;div class="btn-group"&gt;
  &lt;button type="button" class="btn btn-danger dropdown-toggle" data-
  toggle="dropdown" aria-haspopup="true" aria-expanded="false"&gt;
    Action
  &lt;/button&gt;
  &lt;div class="dropdown-menu"&gt;
    &lt;a class="dropdown-item" href="#"&gt;Action&lt;/a&gt;
    &lt;a class="dropdown-item" href="#"&gt;Another action&lt;/a&gt;
    &lt;a class="dropdown-item" href="#"&gt;Something else here&lt;/a&gt;
    &lt;div class="dropdown-divider"&gt;&lt;/div&gt;
    &lt;a class="dropdown-item" href="#"&gt;Separated link&lt;/a&gt;
  &lt;/div&gt;
&lt;/div&gt;
</pre>

```

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.

Primary Secondary Success Info Warning ▾

Danger

Copy

```
<!-- 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-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    <span class="sr-only">Toggle Dropdown</span>
  </button>
  <div class="dropdown-menu">
    <a class="dropdown-item" href="#">Action</a>
    <a class="dropdown-item" href="#">Another action</a>
    <a class="dropdown-item" href="#">Something else here</a>
    <div class="dropdown-divider"></div>
    <a class="dropdown-item" href="#">Separated link</a>
  </div>
</div>
```

Sizing

Button dropdowns work with buttons of all sizes, including default and split dropdown buttons.

Large button Large split button

Small button Small split button

Copy

```

<!-- Large button groups (default and split) -->


<button class="btn btn-secondary btn-lg dropdown-toggle" type="button" data-
  toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Large button
  </button>
  <div class="dropdown-menu">
    ...
  </div>



<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-toggle="dropdown" aria-haspopup="true" aria-
  expanded="false">
    <span class="sr-only">Toggle Dropdown</span>
  </button>
  <div class="dropdown-menu">
    ...
  </div>



<!-- Small button groups (default and split) -->


<button class="btn btn-secondary btn-sm dropdown-toggle" type="button" data-
  toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Small button
  </button>
  <div class="dropdown-menu">
    ...
  </div>



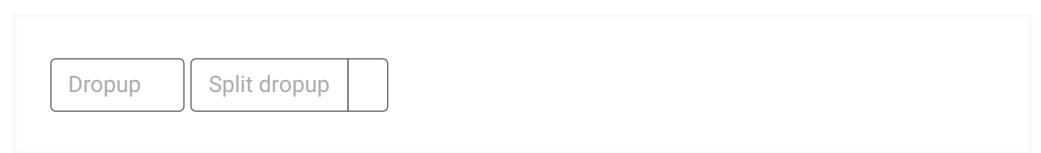
<button class="btn btn-secondary btn-sm" type="button">
    Small split button
  </button>
  <button type="button" class="btn btn-sm btn-secondary dropdown-toggle
  dropdown-toggle-split" data-toggle="dropdown" aria-haspopup="true" aria-
  expanded="false">
    <span class="sr-only">Toggle Dropdown</span>
  </button>
  <div class="dropdown-menu">
    ...
  </div>


```

Directions

Dropup

Trigger dropdown menus above elements by adding `.dropup` to the parent element.



[Copy](#)

```

<!-- Default dropdown button -->


<button type="button" class="btn btn-secondary dropdown-toggle" data-
  toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Dropdown
  </button>
  <div class="dropdown-menu">
    <!-- Dropdown menu links -->
  </div>
</div>

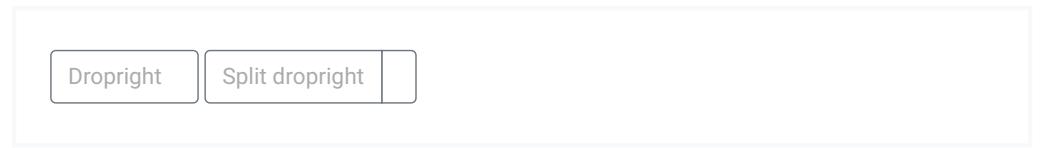
<!-- Split dropdown button -->


<button type="button" class="btn btn-secondary">
    Split dropdown
  </button>
  <button type="button" class="btn btn-secondary dropdown-toggle dropdown-
  toggle-split" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    <span class="sr-only">Toggle Dropdown</span>
  </button>
  <div class="dropdown-menu">
    <!-- Dropdown menu links -->
  </div>
</div>


```

Dropright

Trigger dropdown menus at the right of the elements by adding `.dropright` to the parent element.



```

<!-- Default dropright button -->


<button type="button" class="btn btn-secondary dropdown-toggle" data-
  toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Dropright
  </button>
  <div class="dropdown-menu">
    <!-- Dropdown menu links -->
  </div>
</div>

<!-- Split dropright button -->

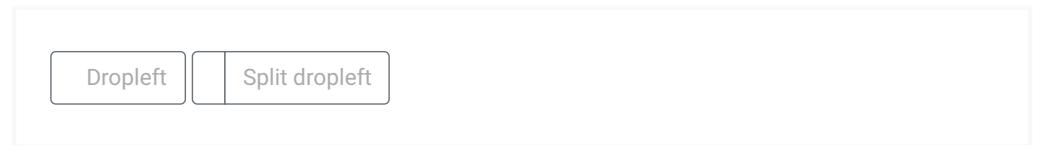

<button type="button" class="btn btn-secondary">
    Split dropright
  </button>
  <button type="button" class="btn btn-secondary dropdown-toggle dropdown-
  toggle-split" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    <span class="sr-only">Toggle Dropright</span>
  </button>
  <div class="dropdown-menu">
    <!-- Dropdown menu links -->
  </div>
</div>


```

[Copy](#)

Dropleft

Trigger dropdown menus at the left of the elements by adding `.dropleft` to the parent element.



[Copy](#)

```
<!-- Default dropleft button -->
<div class="btn-group dropleft">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-
  toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Dropleft
  </button>
  <div class="dropdown-menu">
    <!-- Dropdown menu links -->
  </div>
</div>

<!-- Split dropleft button -->
<div class="btn-group">
  <div class="btn-group dropleft" role="group">
    <button type="button" class="btn btn-secondary dropdown-toggle dropdown-
    toggle-split" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
      <span class="sr-only">Toggle Dropleft</span>
    </button>
    <div class="dropdown-menu">
      <!-- Dropdown menu links -->
    </div>
  </div>
  <button type="button" class="btn btn-secondary">
    Split dropleft
  </button>
</div>
```

Menu items

Historically dropdown menu contents *had* to be links, but that's no longer the case with v4. Now you can optionally use `<button>` elements in your dropdowns instead of just `<a>`s.

[Copy](#)

```
<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle" type="button"
  id="dropdownMenu2" data-toggle="dropdown" aria-haspopup="true" aria-
  expanded="false">
    Dropdown
  </button>
  <div class="dropdown-menu" aria-labelledby="dropdownMenu2">
    <button class="dropdown-item" type="button">Action</button>
    <button class="dropdown-item" type="button">Another action</button>
    <button class="dropdown-item" type="button">Something else here</button>
  </div>
</div>
```

You can also create non-interactive dropdown items with `.dropdown-item-text`. Feel free to style further with custom CSS or text utilities.

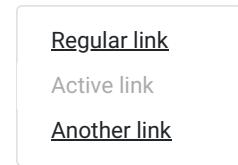
[Copy](#)

```
Dropdown item text
Action
Another action
Something else here
```

```
<div class="dropdown-menu">
  <span class="dropdown-item-text">Dropdown item text</span>
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
  <a class="dropdown-item" href="#">Something else here</a>
</div>
```

Active

Add `.active` to items in the dropdown to **style them as active**.



Regular link
Active link
Another link

Copy

```
<div class="dropdown-menu">
  <a class="dropdown-item" href="#">Regular link</a>
  <a class="dropdown-item active" href="#">Active link</a>
  <a class="dropdown-item" href="#">Another link</a>
</div>
```

Disabled

Add `.disabled` to items in the dropdown to **style them as disabled**.



Regular link
Disabled link
Another link

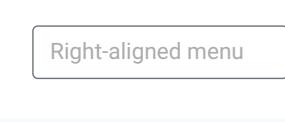
Copy

```
<div class="dropdown-menu">
  <a class="dropdown-item" href="#">Regular link</a>
  <a class="dropdown-item disabled" href="#" tabindex="-1" aria-
disabled="true">Disabled link</a>
  <a class="dropdown-item" href="#">Another link</a>
</div>
```

Menu alignment

By default, a dropdown menu is automatically positioned 100% from the top and along the left side of its parent. Add `.dropdown-menu-right` to a `.dropdown-menu` to right align the dropdown menu.

Heads up! Dropdowns are positioned thanks to Popper.js (except when they are contained in a navbar).



Right-aligned menu

Copy

```
<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-
  toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Right-aligned menu
  </button>
  <div class="dropdown-menu dropdown-menu-right">
    <button class="dropdown-item" type="button">Action</button>
    <button class="dropdown-item" type="button">Another action</button>
    <button class="dropdown-item" type="button">Something else here</button>
  </div>
</div>
```

Responsive alignment

If you want to use responsive alignment, disable dynamic positioning by adding the `data-
 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}-right`.

Left-aligned but right aligned when large screen

Copy

```
<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-
  toggle="dropdown" data-display="static" aria-haspopup="true" aria-
  expanded="false">
    Left-aligned but right aligned when large screen
  </button>
  <div class="dropdown-menu dropdown-menu-lg-right">
    <button class="dropdown-item" type="button">Action</button>
    <button class="dropdown-item" type="button">Another action</button>
    <button class="dropdown-item" type="button">Something else here</button>
  </div>
</div>
```

To align **left** the dropdown menu with the given breakpoint or larger, add `.dropdown-menu-right` and `.dropdown-menu{-sm|-md|-lg|-xl}-left`.

Right-aligned but left aligned when large screen

Copy

```
<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-
  toggle="dropdown" data-display="static" aria-haspopup="true" aria-
  expanded="false">
    Right-aligned but left aligned when large screen
  </button>
  <div class="dropdown-menu dropdown-menu-right dropdown-menu-lg-left">
    <button class="dropdown-item" type="button">Action</button>
    <button class="dropdown-item" type="button">Another action</button>
    <button class="dropdown-item" type="button">Something else here</button>
  </div>
</div>
```

Note that you don't need to add a `data-display="static"` attribute to dropdown buttons in navbars, since Popper.js isn't used in navbars.

Menu content

Headers

Add a header to label sections of actions in any dropdown menu.

Dropdown header

Action

Another action

Copy

```
<div class="dropdown-menu">
  <h6 class="dropdown-header">Dropdown header</h6>
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
</div>
```

Dividers

Separate groups of related menu items with a divider.

Action

Another action

Something else here

Separated link

Copy

```
<div class="dropdown-menu">
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
  <a class="dropdown-item" href="#">Something else here</a>
  <div class="dropdown-divider"></div>
  <a class="dropdown-item" href="#">Separated link</a>
</div>
```

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.

Copy

```
<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.

The screenshot shows a sign-in form within a dropdown menu. The form consists of the following elements:

- Email address input field containing "email@example.com"
- Password input field containing "Password"
- A "Remember me" checkbox labeled "Remember me"
- A "Sign in" button

Below the form, there are two links:

- [New around here? Sign up](#)
- [Forgot password?](#)

```
<div class="dropdown-menu">
  <form class="px-4 py-3">
    <div class="form-group">
      <label for="exampleDropdownFormEmail1">Email address</label>
      <input type="email" class="form-control" id="exampleDropdownFormEmail1" placeholder="email@example.com">
    </div>
    <div class="form-group">
      <label for="exampleDropdownFormPassword1">Password</label>
      <input type="password" class="form-control" id="exampleDropdownFormPassword1" placeholder="Password">
    </div>
    <div class="form-group">
      <div class="form-check">
        <input type="checkbox" class="form-check-input" id="dropdownCheck">
        <label class="form-check-label" for="dropdownCheck">
          Remember me
        </label>
      </div>
      <button type="submit" class="btn btn-primary">Sign in</button>
    </div>
    <div class="dropdown-divider"></div>
    <a class="dropdown-item" href="#">New around here? Sign up</a>
    <a class="dropdown-item" href="#">Forgot password?</a>
  </form>
</div>
```

The screenshot shows a simplified sign-in form within a dropdown menu, featuring only an "Email address" input field.

The form consists of a single column. At the top is an input field for the email address, containing "email@example.com". Below it is a password input field with the placeholder "Password". Underneath the password field is a checkbox labeled "Remember me". At the bottom is a blue-outlined button labeled "Sign in".

Copy

```
<form class="dropdown-menu p-4">
  <div class="form-group">
    <label for="exampleDropdownFormEmail2">Email address</label>
    <input type="email" class="form-control" id="exampleDropdownFormEmail2"
placeholder="email@example.com">
  </div>
  <div class="form-group">
    <label for="exampleDropdownFormPassword2">Password</label>
    <input type="password" class="form-control"
id="exampleDropdownFormPassword2" placeholder="Password">
  </div>
  <div class="form-group">
    <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>
```

Dropdown options

Use `data-offset` or `data-reference` to change the location of the dropdown.

The dropdown menu is open and contains three items: "Offset", "Reference", and an empty item. The "Offset" and "Reference" items are visible, while the third item is partially visible on the right.

Copy

```

<div class="d-flex">
  <div class="dropdown mr-1">
    <button type="button" class="btn btn-secondary dropdown-toggle"
id="dropdownMenuOffset" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false" data-offset="10,20">
      Offset
    </button>
    <div class="dropdown-menu" aria-labelledby="dropdownMenuOffset">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else here</a>
    </div>
  </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-toggle="dropdown" aria-
haspopup="true" aria-expanded="false" data-reference="parent">
      <span class="sr-only">Toggle Dropdown</span>
    </button>
    <div class="dropdown-menu" aria-labelledby="dropdownMenuReference">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else here</a>
      <div class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">Separated link</a>
    </div>
  </div>
</div>

```

Usage

Via data attributes or JavaScript, the dropdown plugin toggles hidden content (dropdown menus) by toggling the `.show` class on the parent list item. The `data-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 (`$.noop`) `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-toggle="dropdown"` to a link or button to toggle a dropdown.

<div class="dropdown">
Copy

```

<div class="dropdown">
  <button id="dLabel" type="button" data-toggle="dropdown" aria-haspopup="true"
aria-expanded="false">
    Dropdown trigger
  </button>
  <div class="dropdown-menu" aria-labelledby="dLabel">
    ...
  </div>
</div>

```

Via JavaScript

Call the dropdowns via JavaScript:

```
$('.dropdown-toggle').dropdown()
```

Copy

`data-toggle="dropdown"` still required

Regardless of whether you call your dropdown via JavaScript or instead use the data-api, `data-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-`, as in `data-offset=""`.

Name	Type	Default	Description
offset	number string function	0	<p>Offset of the dropdown relative to its target.</p> <p>When a function is used to determine the offset, it is called with an object containing the offset data as its first argument. The function must return an object with the same structure. The triggering element DOM node is passed as the second argument.</p> <p>For more information refer to Popper.js's offset docs.</p>
flip	boolean	true	Allow Dropdown to flip in case of an overlapping on the reference element. For more information refer to Popper.js's flip docs .
boundary	string element	'scrollParent'	Overflow constraint boundary of the dropdown menu. Accepts the values of ' <code>viewport</code> ', ' <code>window</code> ', ' <code>scrollParent</code> ', or an HTMLElement reference (JavaScript only). For more information refer to Popper.js's preventOverflow docs .
reference	string element	'toggle'	Reference element of the dropdown menu. Accepts the values of ' <code>toggle</code> ', ' <code>parent</code> ', or an HTMLElement reference. For more information refer to Popper.js's referenceObject docs .
display	string	'dynamic'	By default, we use Popper.js for dynamic positioning. Disable this with <code>static</code> .
popperConfig	null object	null	To change Bootstrap's default Popper.js config, see Popper.js's configuration

Note when `boundary` is set to any value other than '`scrollParent`', the style `position: static` is applied to the `.dropdown` container.

Methods

Method	Description
<code>\$().dropdown('toggle')</code>	Toggles the dropdown menu of a given navbar or tabbed navigation.
<code>\$().dropdown('show')</code>	Shows the dropdown menu of a given navbar or tabbed navigation.
<code>\$().dropdown('hide')</code>	Hides the dropdown menu of a given navbar or tabbed navigation.
<code>\$().dropdown('update')</code>	Updates the position of an element's dropdown.

Method	Description
<code>\$().dropdown('dispose')</code>	Destroys an element's dropdown.

Events

All dropdown events are fired at the `.dropdown-menu`'s parent element and have a `relatedTarget` property, whose value is the toggling anchor 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	Description
<code>show.bs.dropdown</code>	This event fires immediately when the show instance method is called.
<code>shown.bs.dropdown</code>	This event is fired when the dropdown has been made visible to the user (will wait for CSS transitions, to complete).
<code>hide.bs.dropdown</code>	This event is fired immediately when the hide instance method has been called.
<code>hidden.bs.dropdown</code>	This event is fired when the dropdown has finished being hidden from the user (will wait for CSS transitions, to complete).

```
$('#myDropdown').on('show.bs.dropdown', function () {
  // do something...
})
```

Copy

Forms

Search...

Getting started

Layout

Content

Components

Alerts

Badge

Breadcrumb

Buttons

Button group

Card

Carousel

Collapse

Dropdowns

Forms

Input group

Jumbotron

List group

Media object

Modal

Navs

Navbar

Pagination

Popovers

Progress

Scrollspy

Spinners

Toasts

Tooltips

Utilities

Extend

Migration

About



Diversify your revenue beyond in-app purchases with ad monetization

ads via Carbon

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.

Email address

We'll never share your email with anyone else.

Password

Check me out

Submit

Copy

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp">
    <small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="form-group 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 controls

Textual form controls—like `<input>`s, `<select>`s, and `<textarea>`s—are styled with the `.form-control` class. Included are styles for general appearance, focus state, sizing, and more.

Be sure to explore our [custom forms](#) to further style `<select>`s.

Email address

Example select

1

Example multiple select

0 selected

Example textarea

Copy

```
<form>
  <div class="form-group">
    <label for="exampleFormControlInput1">Email address</label>
    <input type="email" class="form-control" id="exampleFormControlInput1" placeholder="name@example.com">
  </div>
  <div class="form-group">
    <label for="exampleFormControlSelect1">Example select</label>
    <select class="form-control" id="exampleFormControlSelect1">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      <option>4</option>
      <option>5</option>
    </select>
  </div>
  <div class="form-group">
    <label for="exampleFormControlSelect2">Example multiple select</label>
    <select multiple class="form-control" id="exampleFormControlSelect2">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      <option>4</option>
      <option>5</option>
    </select>
  </div>
  <div class="form-group">
    <label for="exampleFormControlTextarea1">Example textarea</label>
    <textarea class="form-control" id="exampleFormControlTextarea1" rows="3"></textarea>
  </div>
</form>
```

For file inputs, swap the `.form-control` for `.form-control-file`.

Example file input

 No file chosen

Copy

```
<form>
  <div class="form-group">
    <label for="exampleFormControlFile1">Example file input</label>
    <input type="file" class="form-control-file" id="exampleFormControlFile1">
  </div>
</form>
```

Sizing

Set heights using classes like `.form-control-lg` and `.form-control-sm`.

`.form-control-lg`

Default input

`.form-control-sm`

```
<input class="form-control form-control-lg" type="text" placeholder=".form-control-lg">
<input class="form-control" type="text" placeholder="Default input">
<input class="form-control form-control-sm" type="text" placeholder=".form-control-sm">
```

Copy

Large select ▼

Default select ▼

Small select ▼

```
<select class="form-control form-control-lg">
  <option>Large select</option>
</select>
<select class="form-control">
  <option>Default select</option>
</select>
<select class="form-control form-control-sm">
  <option>Small select</option>
</select>
```

Copy

Readonly

Add the `readonly` boolean attribute on an input to prevent modification of the input's value. Read-only inputs appear lighter (just like disabled inputs), but retain the standard cursor.

Readonly input here...

```
<input class="form-control" type="text" placeholder="Readonly input here..." readonly>
```

Copy

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.

Email email@example.com

Password

Copy

```
<form>
  <div class="form-group 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="form-group 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>
</form>
```

email@example.com

Password

Confirm identity

Copy

```
<form class="form-inline">
  <div class="form-group mb-2">
    <label for="staticEmail2" class="sr-only">Email</label>
    <input type="text" readonly class="form-control-plaintext" id="staticEmail2" value="email@example.com">
  </div>
  <div class="form-group mx-sm-3 mb-2">
    <label for="inputPassword2" class="sr-only">Password</label>
    <input type="password" class="form-control" id="inputPassword2" placeholder="Password">
  </div>
  <button type="submit" class="btn btn-primary mb-2">Confirm identity</button>
</form>
```

Range Inputs

Set horizontally scrollable range inputs using `.form-control-range`.

Example Range input



Copy

```
<form>
  <div class="form-group">
    <label for="formControlRange">Example Range input</label>
    <input type="range" class="form-control-range" id="formControlRange">
  </div>
</form>
```

Checkboxes and radios

Default checkboxes and radios are improved upon with the help of `.form-check`, a single class for both input types that improves the layout and behavior of their HTML elements. Checkboxes are for selecting one or several options in a list, while radios are for selecting one option from many.

Disabled checkboxes and radios are supported. The `disabled` attribute will apply a lighter color to help indicate the input's state.

Checkboxes and radio buttons support HTML-based form validation and provide concise, accessible labels. As such, 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>`.

Default (stacked)

By default, any number of checkboxes and radios that are immediate sibling will be vertically stacked and appropriately spaced with `.form-check`.

- Default checkbox
- Disabled checkbox

Copy

```
<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>
```

- Default radio
- Second default radio
- Disabled radio

Copy

```
<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`.

1 2 3 (disabled)

Copy

```
<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>
```

1 2 3 (disabled)

Copy

```
<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>
```

Without labels

Add `.position-static` to inputs within `.form-check` that don't have any label text. Remember to still provide some form of label for assistive technologies (for instance, using `aria-label`).

Copy

```
<div class="form-check">
  <input class="form-check-input position-static" type="checkbox" id="blankCheckbox" value="option1" aria-label="..."/>
</div>
<div class="form-check">
  <input class="form-check-input position-static" type="radio" name="blankRadio" id="blankRadio1" value="option1" aria-label="..."/>
</div>
```

Layout

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.

Form groups

The `.form-group` class is the easiest way to add some structure to forms. It provides a flexible class that encourages proper grouping of labels, controls, optional help text, and form validation messaging. By default it only applies `margin-bottom`, but it picks up additional styles in `.form-inline` as needed. Use it with `<fieldset>`s, `<div>`s, or nearly any other element.

Example label

Example input placeholder

Another label

Another input placeholder

Copy

```
<form>
  <div class="form-group">
    <label for="formGroupExampleInput">Example label</label>
    <input type="text" class="form-control" id="formGroupExampleInput"
placeholder="Example input placeholder">
  </div>
  <div class="form-group">
    <label for="formGroupExampleInput2">Another label</label>
    <input type="text" class="form-control" id="formGroupExampleInput2"
placeholder="Another input placeholder">
  </div>
</form>
```

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.

First name

Last name

Copy

```
<form>
  <div class="row">
    <div class="col">
      <input type="text" class="form-control" placeholder="First name">
    </div>
    <div class="col">
      <input type="text" class="form-control" placeholder="Last name">
    </div>
  </div>
</form>
```

Form row

You may also swap `.row` for `.form-row`, a variation of our standard grid row that overrides the default column gutters for tighter and more compact layouts.

First name

Last name

Copy

```
<form>
  <div class="form-row">
    <div class="col">
      <input type="text" class="form-control" placeholder="First name">
    </div>
    <div class="col">
      <input type="text" class="form-control" placeholder="Last name">
    </div>
  </div>
</form>
```

More complex layouts can also be created with the grid system.

Email	Password	
<input type="text"/>	<input type="password"/>	
Address		
<input type="text" value="1234 Main St"/>		
Address 2		
<input type="text" value="Apartment, studio, or floor"/>		
City	State	Zip
<input type="text"/>	<input type="text" value="Choose..."/> ▼	<input type="text"/>
<input type="checkbox"/> Check me out		
<input type="button" value="Sign in"/>		

Copy

```

<form>
  <div class="form-row">
    <div class="form-group col-md-6">
      <label for="inputEmail4">Email</label>
      <input type="email" class="form-control" id="inputEmail4">
    </div>
    <div class="form-group col-md-6">
      <label for="inputPassword4">Password</label>
      <input type="password" class="form-control" id="inputPassword4">
    </div>
  </div>
  <div class="form-group">
    <label for="inputAddress">Address</label>
    <input type="text" class="form-control" id="inputAddress" placeholder="1234 Main St">
  </div>
  <div class="form-group">
    <label for="inputAddress2">Address 2</label>
    <input type="text" class="form-control" id="inputAddress2" placeholder="Apartment, studio, or floor">
  </div>
  <div class="form-row">
    <div class="form-group col-md-6">
      <label for="inputCity">City</label>
      <input type="text" class="form-control" id="inputCity">
    </div>
    <div class="form-group col-md-4">
      <label for="inputState">State</label>
      <select id="inputState" class="form-control">
        <option selected>Choose...</option>
        <option>...</option>
      </select>
    </div>
    <div class="form-group col-md-2">
      <label for="inputZip">Zip</label>
      <input type="text" class="form-control" id="inputZip">
    </div>
  </div>
  <div class="form-group">
    <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>
  <button type="submit" class="btn btn-primary">Sign in</button>
</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.

Email	<input type="text"/>
Password	<input type="text"/>
Radios	<input checked="" type="radio"/> First radio <input type="radio"/> Second radio <input type="radio"/> Third disabled radio

Checkbox

 Example checkbox[Sign in](#)[Copy](#)

```

<form>
  <div class="form-group row">
    <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="form-group row">
    <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="form-group">
    <div class="row">
      <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>
    </div>
  </fieldset>
  <div class="form-group row">
    <div class="col-sm-2">Checkbox</div>
    <div class="col-sm-10">
      <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>
  <div class="form-group row">
    <div class="col-sm-10">
      <button type="submit" class="btn btn-primary">Sign in</button>
    </div>
  </div>
</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`.

Email	col-form-label-sm
Email	col-form-label
Email	col-form-label-lg

Copy

```
<form>
  <div class="form-group row">
    <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="form-group row">
    <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="form-group 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>
</form>
```

Column sizing

As shown in the previous examples, our grid system allows you to place any number of `.cols` within a `.row` or `.form-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 `.cols` equally split the rest, with specific column classes like `.col-7`.

City	State	Zip
------	-------	-----

Copy

```
<form>
  <div class="form-row">
    <div class="col-7">
      <input type="text" class="form-control" placeholder="City">
    </div>
    <div class="col">
      <input type="text" class="form-control" placeholder="State">
    </div>
    <div class="col">
      <input type="text" class="form-control" placeholder="Zip">
    </div>
  </div>
</form>
```

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.

A screenshot of a login form. It consists of two input fields: one for 'Name' containing 'Jane Doe' and another for 'Username' containing '@ Username'. Below the inputs is a checkbox labeled 'Remember me' and a blue-bordered 'Submit' button.

Copy

```
<form>
  <div class="form-row align-items-center">
    <div class="col-auto">
      <label class="sr-only" for="inlineFormInput">Name</label>
      <input type="text" class="form-control mb-2" id="inlineFormInput"
placeholder="Jane Doe">
    </div>
    <div class="col-auto">
      <label class="sr-only" for="inlineFormInputGroup">Username</label>
      <div class="input-group mb-2">
        <div class="input-group-prepend">
          <div class="input-group-text">@</div>
        </div>
        <input type="text" class="form-control" id="inlineFormInputGroup"
placeholder="Username">
      </div>
    </div>
    <div class="col-auto">
      <div class="form-check mb-2">
        <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 mb-2">Submit</button>
    </div>
  </div>
</form>
```

You can then remix that once again with size-specific column classes.

A screenshot of a login form, identical to the first one, but with the 'col-auto' class replaced by 'col-2' in the first two columns of the form row. This results in wider input fields and labels.

Copy

```

<form>
  <div class="form-row align-items-center">
    <div class="col-sm-3 my-1">
      <label class="sr-only" for="inlineFormInputName">Name</label>
      <input type="text" class="form-control" id="inlineFormInputName"
placeholder="Jane Doe">
    </div>
    <div class="col-sm-3 my-1">
      <label class="sr-only" for="inlineFormInputGroupUsername">Username</label>
      <div class="input-group">
        <div class="input-group-prepend">
          <div class="input-group-text">@</div>
        </div>
        <input type="text" class="form-control"
id="inlineFormInputGroupUsername" placeholder="Username">
      </div>
    </div>
    <div class="col-auto my-1">
      <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 my-1">
      <button type="submit" class="btn btn-primary">Submit</button>
    </div>
  </div>
</form>

```

And of course [custom form controls](#) are supported.

```

<form>
  <div class="form-row align-items-center">
    <div class="col-auto my-1">
      <label class="mr-sm-2 sr-only"
for="inlineFormCustomSelect">Preference</label>
      <select class="custom-select mr-sm-2" id="inlineFormCustomSelect">
        <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 my-1">
      <div class="custom-control custom-checkbox mr-sm-2">
        <input type="checkbox" class="custom-control-input"
id="customControlAutosizing">
        <label class="custom-control-label"
for="customControlAutosizing">Remember my preference</label>
      </div>
    </div>
    <div class="col-auto my-1">
      <button type="submit" class="btn btn-primary">Submit</button>
    </div>
  </div>
</form>

```

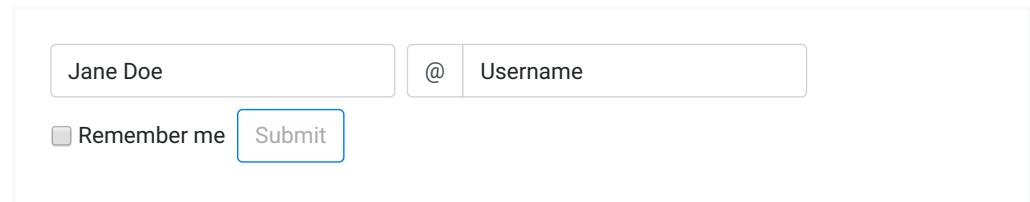
[Copy](#)

Inline forms

Use the `.form-inline` class to display a series of labels, form controls, and buttons on a single horizontal row. Form controls within inline forms vary slightly from their default states.

- Controls are `display: flex`, collapsing any HTML white space and allowing you to provide alignment control with `spacing` and `flexbox` utilities.
- Controls and input groups receive `width: auto` to override the Bootstrap default `width: 100%`.
- Controls **only appear inline in viewports that are at least 576px wide** to account for narrow viewports on mobile devices.

You may need to manually address the width and alignment of individual form controls with `spacing utilities` (as shown below). Lastly, be sure to always include a `<label>` with each form control, even if you need to hide it from non-screenreader visitors with `.sr-only`.



A screenshot of a login form demonstrating inline form usage. It contains two text input fields: one for 'Name' with placeholder 'Jane Doe' and another for 'Username' with placeholder '@ Username'. Below these is a checkbox labeled 'Remember me' and a blue-outlined 'Submit' button.

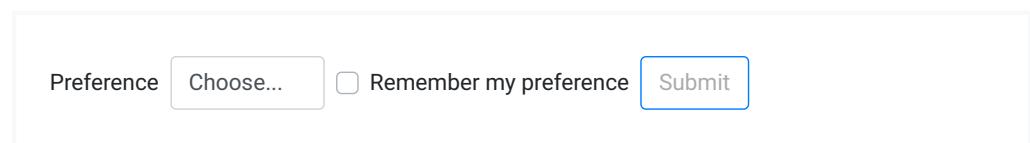
```
<form class="form-inline">
  <label class="sr-only" for="inlineFormInputName2">Name</label>
  <input type="text" class="form-control mb-2 mr-sm-2" id="inlineFormInputName2"
placeholder="Jane Doe">

  <label class="sr-only" for="inlineFormInputGroupUsername2">Username</label>
  <div class="input-group mb-2 mr-sm-2">
    <div class="input-group-prepend">
      <div class="input-group-text">@</div>
    </div>
    <input type="text" class="form-control" id="inlineFormInputGroupUsername2"
placeholder="Username">
  </div>

  <div class="form-check mb-2 mr-sm-2">
    <input class="form-check-input" type="checkbox" id="inlineFormCheck">
    <label class="form-check-label" for="inlineFormCheck">
      Remember me
    </label>
  </div>

  <button type="submit" class="btn btn-primary mb-2">Submit</button>
</form>
```

Custom form controls and selects are also supported.



A screenshot of a form featuring a 'Preference' input field with a placeholder 'Choose...', a dropdown menu, a checkbox labeled 'Remember my preference', and a blue-outlined 'Submit' button.

Copy

```

<form class="form-inline">
  <label class="my-1 mr-2" for="inlineFormCustomSelectPref">Preference</label>
  <select class="custom-select my-1 mr-sm-2" id="inlineFormCustomSelectPref">
    <option selected>Choose...</option>
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </select>

  <div class="custom-control custom-checkbox my-1 mr-sm-2">
    <input type="checkbox" class="custom-control-input" id="customControlInline">
    <label class="custom-control-label" for="customControlInline">Remember my preference</label>
  </div>

  <button type="submit" class="btn btn-primary my-1">Submit</button>
</form>

```

Alternatives to hidden labels

Assistive technologies such as screen readers will have trouble with your forms if you don't include a label for every input. For these inline forms, you can hide the labels using the `.sr-only` class. There are further alternative methods of providing a label for assistive technologies, such as the `aria-label`, `aria-labelledby` or `title` attribute. If none of these are present, assistive technologies may resort to using the `placeholder` attribute, if present, but note that use of `placeholder` as a replacement for other labelling methods is not advised.

Help text

Block-level help text in forms can be created using `.form-text` (previously known as `.help-block` in v3). Inline help text can be flexibly implemented using any inline HTML element and utility classes like `.text-muted`.

Associating help text with form controls

Help 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 help text when the user focuses or enters the control.

Help text below inputs can be styled with `.form-text`. This class includes `display: block` and adds some top margin for easy spacing from the inputs above.

Password

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

`<label for="inputPassword5">Password</label>`

`<input type="password" id="inputPassword5" class="form-control" aria-describedby="passwordHelpBlock">`

`<small id="passwordHelpBlock" class="form-text text-muted">`

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

`</small>`

Copy

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

Password Must be 8-20 characters long.

Copy

```
<form class="form-inline">
  <div class="form-group">
    <label for="inputPassword6">Password</label>
    <input type="password" id="inputPassword6" class="form-control mx-sm-3" aria-describedby="passwordHelpInline">
    <small id="passwordHelpInline" class="text-muted">
      Must be 8-20 characters long.
    </small>
  </div>
</form>
```

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>
```

Copy

Add the `disabled` attribute to a `<fieldset>` to disable all the controls within.

Disabled input

Disabled input

Disabled select menu

Disabled select



Can't check this

```
<form>
  <fieldset disabled>
    <div class="form-group">
      <label for="disabledTextInput">Disabled input</label>
      <input type="text" id="disabledTextInput" class="form-control" placeholder="Disabled input">
    </div>
    <div class="form-group">
      <label for="disabledSelect">Disabled select menu</label>
      <select id="disabledSelect" class="form-control">
        <option>Disabled select</option>
      </select>
    </div>
    <div class="form-group">
      <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>
```

Copy

Caveat with anchors

By default, browsers will 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 `<a ... class="btn btn-*">` elements, these will only be given a style of `pointer-events: none`. As noted in the section about [disabled state for buttons](#) (and specifically in the sub-section for anchor elements), this CSS property is not yet standardized and isn't fully supported in Internet Explorer 10, and won't prevent keyboard users from being able to focus or activate these links. So to be safe, use custom JavaScript to disable such links.

Cross-browser compatibility

While Bootstrap will apply these styles in all browsers, Internet Explorer 11 and below don't fully support the `disabled` attribute on a `<fieldset>`. Use custom JavaScript to disable the fieldset in these browsers.

Validation

Provide valuable, actionable feedback to your users with HTML5 form validation—[available in all our supported browsers](#). Choose from the browser default validation feedback, or implement custom messages with our built-in classes and starter JavaScript.

We currently recommend using custom validation styles, as native browser default validation messages are not consistently exposed to assistive technologies in all browsers (most notably, Chrome on desktop and mobile).

Input group validation

Input groups have difficulty with validation styles, unfortunately. Our recommendation is to place feedback messages as sibling elements of the `.input-group` that has `.is-{valid|invalid}`. Placing feedback messages within input groups breaks the `border-radius`. [See this workaround](#).

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 unstylistable 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 `.custom-select`, and not `.form-control`.

First name	Last name	
<input type="text" value="Mark"/>	<input type="text" value="Otto"/>	
City	State	Zip
<input type="text"/>	<input type="text" value="Choose..."/>	<input type="text"/>
<input type="checkbox"/> Agree to terms and conditions		
<input type="button" value="Submit form"/>		

[Copy](#)

```

<form class="needs-validation" novalidate>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationCustom01">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-6 mb-3">
      <label for="validationCustom02">Last name</label>
      <input type="text" class="form-control" id="validationCustom02"
      value="Otto" required>
      <div class="valid-feedback">
        Looks good!
      </div>
    </div>
  </div>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationCustom03">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 mb-3">
      <label for="validationCustom04">State</label>
      <select class="custom-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 mb-3">
      <label for="validationCustom05">Zip</label>
      <input type="text" class="form-control" id="validationCustom05" required>
      <div class="invalid-feedback">
        Please provide a valid zip.
      </div>
    </div>
  </div>
  <div class="form-group">
    <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>
  <button class="btn btn-primary" type="submit">Submit form</button>
</form>

<script>
// Example starter JavaScript for disabling form submissions if there are
invalid fields
(function() {
  'use strict';
  window.addEventListener('load', function() {
    // Fetch all the forms we want to apply custom Bootstrap validation styles
    to
    var forms = document.getElementsByClassName('needs-validation');
    // Loop over them and prevent submission

```

```
var validation = Array.prototype.filter.call(forms, function(form) {
  form.addEventListener('submit', function(event) {
    if (form.checkValidity() === false) {
      event.preventDefault();
      event.stopPropagation();
    }
    form.classList.add('was-validated');
  }, false);
});
})();
</script>
```

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.

First name	Last name	
<input type="text" value="Mark"/>	<input type="text" value="Otto"/>	
City	State	Zip
<input type="text"/>	<input type="text" value="Choose..."/>	<input type="text"/>
<input type="checkbox"/> Agree to terms and conditions		
<input type="button" value="Submit form"/>		

Copy

```

<form>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationDefault01">First name</label>
      <input type="text" class="form-control" id="validationDefault01" value="Mark" required>
    </div>
    <div class="col-md-6 mb-3">
      <label for="validationDefault02">Last name</label>
      <input type="text" class="form-control" id="validationDefault02" value="Otto" required>
    </div>
  </div>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationDefault03">City</label>
      <input type="text" class="form-control" id="validationDefault03" required>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationDefault04">State</label>
      <select class="custom-select" id="validationDefault04" required>
        <option selected disabled value="">Choose...</option>
        <option>...</option>
      </select>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationDefault05">Zip</label>
      <input type="text" class="form-control" id="validationDefault05" required>
    </div>
  </div>
  <div class="form-group">
    <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>
  <button class="btn btn-primary" type="submit">Submit form</button>
</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.

First name	Last name	
<input style="border: 1px solid #ccc; width: 100%; height: 30px; border-radius: 5px; padding: 5px;" type="text" value="Mark"/> <div style="font-size: small; color: green; margin-top: 2px;">Looks good!</div>	<input style="border: 1px solid #ccc; width: 100%; height: 30px; border-radius: 5px; padding: 5px;" type="text" value="Otto"/> <div style="font-size: small; color: green; margin-top: 2px;">Looks good!</div>	
City	State	Zip
<input style="border: 1px solid red; width: 100%; height: 30px; border-radius: 5px; padding: 5px;" type="text"/> <div style="font-size: small; color: red; margin-top: 2px;">Please provide a valid city.</div>	<input style="border: 1px solid red; width: 100%; height: 30px; border-radius: 5px; padding: 5px;" type="text" value="Choose..."/> <div style="font-size: small; color: red; margin-top: 2px;">Please select a valid state.</div>	<input style="border: 1px solid red; width: 100%; height: 30px; border-radius: 5px; padding: 5px;" type="text"/> <div style="font-size: small; color: red; margin-top: 2px;">Please provide a valid zip.</div>
<input checked="" type="checkbox"/> Agree to terms and conditions <div style="font-size: small; color: red; margin-top: 2px;">You must agree before submitting.</div>		
<input style="border: 1px solid #007bff; border-radius: 5px; padding: 5px; background-color: #007bff; color: white; font-weight: bold; width: fit-content;" type="button" value="Submit form"/>		

```

<form>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationServer01">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-6 mb-3">
      <label for="validationServer02">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>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationServer03">City</label>
      <input type="text" class="form-control is-invalid" id="validationServer03" required>
      <div class="invalid-feedback">
        Please provide a valid city.
      </div>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationServer04">State</label>
      <select class="custom-select is-invalid" id="validationServer04" 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 mb-3">
      <label for="validationServer05">Zip</label>
      <input type="text" class="form-control is-invalid" id="validationServer05" required>
      <div class="invalid-feedback">
        Please provide a valid zip.
      </div>
    </div>
  </div>
  <div class="form-group">
    <div class="form-check">
      <input class="form-check-input is-invalid" type="checkbox" value="" id="invalidCheck3" required>
      <label class="form-check-label" for="invalidCheck3">
        Agree to terms and conditions
      </label>
      <div class="invalid-feedback">
        You must agree before submitting.
      </div>
    </div>
  </div>
  <button class="btn btn-primary" type="submit">Submit form</button>
</form>

```

Supported elements

Validation styles are available for the following form controls and components:

- `<input>`s and `<textarea>`s with `.form-control`
- `<select>`s with `.form-control` or `.custom-select`

- `.form-checks`
- `.custom-checkboxes` and `.custom-radios`
- `.custom-file`

Textarea

Required example textarea

Please enter a message in the textarea.

Check this custom checkbox

Example invalid feedback text

Toggle this custom radio

Or toggle this other custom radio

More example invalid feedback text

Choose...

Example invalid custom select feedback

Choose file...

Browse

Example invalid custom file feedback

@

Example invalid input group feedback

Options Choose...

Example invalid input group feedback

Choose file...

Browse

Button

Example invalid input group feedback

Copy

```

<form class="was-validated">
  <div class="mb-3">
    <label for="validationTextarea">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="custom-control custom-checkbox mb-3">
    <input type="checkbox" class="custom-control-input"
id="customControlValidation1" required>
    <label class="custom-control-label" for="customControlValidation1">Check
this custom checkbox</label>
    <div class="invalid-feedback">Example invalid feedback text</div>
  </div>

  <div class="custom-control custom-radio">
    <input type="radio" class="custom-control-input"
id="customControlValidation2" name="radio-stacked" required>
    <label class="custom-control-label" for="customControlValidation2">Toggle
this custom radio</label>
  </div>
  <div class="custom-control custom-radio mb-3">
    <input type="radio" class="custom-control-input"
id="customControlValidation3" name="radio-stacked" required>
    <label class="custom-control-label" for="customControlValidation3">Or toggle
this other custom radio</label>
    <div class="invalid-feedback">More example invalid feedback text</div>
  </div>

  <div class="mb-3">
    <select class="custom-select" required>
      <option value="">Choose...</option>
      <option value="1">One</option>
      <option value="2">Two</option>
      <option value="3">Three</option>
    </select>
    <div class="invalid-feedback">Example invalid custom select feedback</div>
  </div>

  <div class="custom-file mb-3">
    <input type="file" class="custom-file-input" id="validatedCustomFile"
required>
    <label class="custom-file-label" for="validatedCustomFile">Choose file...
  </label>
    <div class="invalid-feedback">Example invalid custom file feedback</div>
  </div>

  <div class="mb-3">
    <div class="input-group is-invalid">
      <div class="input-group-prepend">
        <span class="input-group-text" id="validatedInputGroupPrepend">@</span>
      </div>
      <input type="text" class="form-control is-invalid" aria-
describedby="validatedInputGroupPrepend" required>
    </div>
    <div class="invalid-feedback">
      Example invalid input group feedback
    </div>
  </div>

  <div class="mb-3">
    <div class="input-group is-invalid">
      <div class="input-group-prepend">
        <label class="input-group-text"
for="validatedInputGroupSelect">Options</label>
      </div>
      <select class="custom-select" id="validatedInputGroupSelect" required>

```

```

<option value="">Choose...</option>
<option value="1">One</option>
<option value="2">Two</option>
<option value="3">Three</option>
</select>
</div>
<div class="invalid-feedback">
  Example invalid input group feedback
</div>
</div>

<div class="input-group is-invalid">
  <div class="custom-file">
    <input type="file" class="custom-file-input" id="validatedInputGroupCustomFile" required>
    <label class="custom-file-label" for="validatedInputGroupCustomFile">Choose file...</label>
  </div>
  <div class="input-group-append">
    <button class="btn btn-outline-secondary" type="button">Button</button>
  </div>
</div>
<div class="invalid-feedback">
  Example invalid input group feedback
</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.

First name	Last name	
<input type="text" value="Mark"/>	<input type="text" value="Otto"/>	
City	State	Zip
<input type="text"/>	<input type="text" value="Choose..."/>	<input type="text"/>
<input type="button" value="Submit form"/>		

Copy

```

<form class="needs-validation" novalidate>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationTooltip01">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-6 mb-3">
      <label for="validationTooltip02">Last name</label>
      <input type="text" class="form-control" id="validationTooltip02" value="Otto" required>
      <div class="valid-tooltip">
        Looks good!
      </div>
    </div>
  </div>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationTooltip03">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 mb-3">
      <label for="validationTooltip04">State</label>
      <select class="custom-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 mb-3">
      <label for="validationTooltip05">Zip</label>
      <input type="text" class="form-control" id="validationTooltip05" required>
      <div class="invalid-tooltip">
        Please provide a valid zip.
      </div>
    </div>
  </div>
  <button class="btn btn-primary" type="submit">Submit form</button>
</form>

```

Customizing

Validation states can be customized via Sass with the `$form-validation-states` map. Located in our `_variables.scss` file, this Sass map is looped over to generate the default `valid`/`invalid` validation states. Included is a nested map for customizing each state's color and icon. While no other states are supported by browsers, those using custom styles can easily add more complex form feedback.

Please note that we do not recommend customizing these values without also modifying the `form-validation-state` mixin.

Copy

```

// Sass map from `_variables.scss`
// Override this and recompile your Sass to generate different states
$form-validation-states: map-merge(
  (
    "valid": (
      "color": $form-feedback-valid-color,
      "icon": $form-feedback-icon-valid
    ),
    "invalid": (
      "color": $form-feedback-invalid-color,
      "icon": $form-feedback-icon-invalid
    )
  ),
  $form-validation-states
);

// Loop from `_forms.scss`
// 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, map-get($data, color), map-get($data, icon));
}

```

Input group validation workaround

We're unable to resolve the broken `border-radius` of input groups with validation due to selector limitations, so manual overrides are required. When you're using a standard input group and don't customize the default border radius values, add `.rounded-right` to the elements with the broken `border-radius`.

Copy

```

<div class="input-group">
  <div class="input-group-prepend">
    <span class="input-group-text">@</span>
  </div>
  <input type="text" class="form-control rounded-right" required>
  <div class="invalid-feedback">
    Please choose a username.
  </div>
</div>

```



When you are using a small or large input group or customizing the default `border-radius` values, add custom CSS to the element with the busted `border-radius`.

Copy

```

/* Change values to match the radius of your form control */
.fix-rounded-right {
  border-top-right-radius: .2rem !important;
  border-bottom-right-radius: .2rem !important;
}

```

Copy

```
<div class="input-group input-group-sm">
  <div class="input-group-prepend">
    <span class="input-group-text">@</span>
  </div>
  <input type="text" class="form-control fix-rounded-right" required>
  <div class="invalid-feedback">
    Please choose a username.
  </div>
</div>
```



Custom forms

For even more customization and cross browser consistency, use our completely custom form elements to replace the browser defaults. They're built on top of semantic and accessible markup, so they're solid replacements for any default form control.

Checkboxes and radios

Each checkbox and radio `<input>` and `<label>` pairing is wrapped in a `<div>` to create our custom control. Structurally, this is the same approach as our default `.form-check`.

We use the sibling selector `(-)` for all our `<input>` states—like `:checked`—to properly style our custom form indicator. When combined with the `.custom-control-label` class, we can also style the text for each item based on the `<input>`'s state.

We hide the default `<input>` with `opacity` and use the `.custom-control-label` to build a new custom form indicator in its place with `::before` and `::after`. Unfortunately we can't build a custom one from just the `<input>` because CSS's `content` doesn't work on that element.

In the checked states, we use **base64 embedded SVG icons** from [Open Iconic](#). This provides us the best control for styling and positioning across browsers and devices.

Checkboxes

Check this custom checkbox

```
<div class="custom-control custom-checkbox">
  <input type="checkbox" class="custom-control-input" id="customCheck1">
  <label class="custom-control-label" for="customCheck1">Check this custom
checkbox</label>
</div>
```

Copy

Custom checkboxes can also utilize the `:indeterminate` pseudo class when manually set via JavaScript (there is no available HTML attribute for specifying it).

Check this custom checkbox

If you're using jQuery, something like this should suffice:

```
$('.your-checkbox').prop('indeterminate', true)
```

Copy

Radios

- Toggle this custom radio
- Or toggle this other custom radio

Copy

```
<div class="custom-control custom-radio">
  <input type="radio" id="customRadio1" name="customRadio" class="custom-control-input">
    <label class="custom-control-label" for="customRadio1">Toggle this custom
radio</label>
</div>
<div class="custom-control custom-radio">
  <input type="radio" id="customRadio2" name="customRadio" class="custom-control-input">
    <label class="custom-control-label" for="customRadio2">Or toggle this other
custom radio</label>
</div>
```

Inline

- Toggle this custom radio
- Or toggle this other custom radio

Copy

```
<div class="custom-control custom-radio custom-control-inline">
  <input type="radio" id="customRadioInline1" name="customRadioInline1"
class="custom-control-input">
    <label class="custom-control-label" for="customRadioInline1">Toggle this
custom radio</label>
</div>
<div class="custom-control custom-radio custom-control-inline">
  <input type="radio" id="customRadioInline2" name="customRadioInline1"
class="custom-control-input">
    <label class="custom-control-label" for="customRadioInline2">Or toggle this
other custom radio</label>
</div>
```

Disabled

Custom checkboxes and radios can also be disabled. Add the `disabled` boolean attribute to the `<input>` and the custom indicator and label description will be automatically styled.

- Check this custom checkbox
- Toggle this custom radio

Copy

```
<div class="custom-control custom-checkbox">
  <input type="checkbox" class="custom-control-input" id="customCheckDisabled1"
disabled>
    <label class="custom-control-label" for="customCheckDisabled1">Check this
custom checkbox</label>
</div>

<div class="custom-control custom-radio">
  <input type="radio" name="radioDisabled" id="customRadioDisabled2"
class="custom-control-input" disabled>
    <label class="custom-control-label" for="customRadioDisabled2">Toggle this
custom radio</label>
</div>
```

Switches

A switch has the markup of a custom checkbox but uses the `.custom-switch` class to render a toggle switch. Switches also support the `disabled` attribute.

- Toggle this switch element
- Disabled switch element

Copy

```
<div class="custom-control custom-switch">
  <input type="checkbox" class="custom-control-input" id="customSwitch1">
  <label class="custom-control-label" for="customSwitch1">Toggle this switch
element</label>
</div>
<div class="custom-control custom-switch">
  <input type="checkbox" class="custom-control-input" disabled
id="customSwitch2">
  <label class="custom-control-label" for="customSwitch2">Disabled switch
element</label>
</div>
```

Select menu

Custom `<select>` menus need only a custom class, `.custom-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.

Open this select menu

Copy

```
<select class="custom-select">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

You may also choose from small and large custom selects to match our similarly sized text inputs.

Open this select menu

Open this select menu

Copy

```
<select class="custom-select custom-select-lg mb-3">
  <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="custom-select custom-select-sm">
  <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:

Open this select menu

Copy

```
<select class="custom-select" multiple>
  <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:

Open this select menu

Copy

```
<select class="custom-select" size="3">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

Range

Create custom `<input type="range">` controls with `.custom-range`. The track (the background) and thumb (the value) are both styled to appear the same across browsers. As only IE and Firefox support “filling” their track from the left or right of the thumb as a means to visually indicate progress, we do not currently support it.

Example range

Copy

```
<label for="customRange1">Example range</label>
<input type="range" class="custom-range" id="customRange1">
```

Range inputs have implicit values for `min` and `max`—`0` and `100`, respectively. You may specify new values for those using the `min` and `max` attributes.

Example range

Copy

```
<label for="customRange2">Example range</label>
<input type="range" class="custom-range" min="0" max="5" id="customRange2">
```

By default, range inputs “snap” to integer values. To change this, you can specify a `step` value. In the example below, we double the number of steps by using `step="0.5"`.

Example range

Copy

```
<label for="customRange3">Example range</label>
<input type="range" class="custom-range" min="0" max="5" step="0.5"
id="customRange3">
```

File browser

The recommended plugin to animate custom file input: [bs-custom-file-input](#), that's what we are using currently here in our docs.

The file input is the most gnarly of the bunch and requires additional JavaScript if you'd like to hook them up with functional *Choose file...* and selected file name text.



```
<div class="custom-file">
  <input type="file" class="custom-file-input" id="customFile">
  <label class="custom-file-label" for="customFile">Choose file</label>
</div>
```

Copy

We hide the default file `<input>` via `opacity` and instead style the `<label>`. The button is generated and positioned with `::after`. Lastly, we declare a `width` and `height` on the `<input>` for proper spacing for surrounding content.

Translating or customizing the strings with SCSS

The [:`lang\(\)` pseudo-class](#) is used to allow for translation of the "Browse" text into other languages. Override or add entries to the `$custom-file-text` Sass variable with the relevant [language tag](#) and localized strings. The English strings can be customized the same way. For example, here's how one might add a Spanish translation (Spanish's language code is `es`):

```
$custom-file-text: (
  en: "Browse",
  es: "Elegir"
);
```

Copy

Here's `lang(es)` in action on the custom file input for a Spanish translation:



```
<div class="custom-file">
  <input type="file" class="custom-file-input" id="customFileLang" lang="es">
  <label class="custom-file-label" for="customFileLang">Seleccionar
  Archivo</label>
</div>
```

Copy

You'll need to set the language of your document (or subtree thereof) correctly in order for the correct text to be shown. This can be done using [the `lang` attribute](#) on the `<html>` element or the [Content-Language HTTP header](#), among other methods.

Translating or customizing the strings with HTML

Bootstrap also provides a way to translate the "Browse" text in HTML with the `data-browse` attribute which can be added to the custom input label (example in Dutch):

Voeg je document toe

Bestand kiezen

Copy

```
<div class="custom-file">
  <input type="file" class="custom-file-input" id="customFileLangHTML">
  <label class="custom-file-label" for="customFileLangHTML" data-browse="Bestand
kiezen">Voeg je document toe</label>
</div>
```

Input group

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Toolips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Adobe Creative Cloud for
Teams starting at \$33.99
per month.

ads via Carbon

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.

@	Username
---	----------

Recipient's username	@example.com
----------------------	--------------

Your vanity URL	
https://example.com/users/	

\$.00
----	-----

With textarea	
---------------	--

Copy

```

<div class="input-group mb-3">
  <div class="input-group-prepend">
    <span class="input-group-text" id="basic-addon1">@</span>
  </div>
  <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">
  <div class="input-group-append">
    <span class="input-group-text" id="basic-addon2">@example.com</span>
  </div>
</div>

<label for="basic-url">Your vanity URL</label>
<div class="input-group mb-3">
  <div class="input-group-prepend">
    <span class="input-group-text" id="basic-addon3">https://example.com/users/</span>
  </div>
  <input type="text" class="form-control" id="basic-url" aria-describedby="basic-addon3">
</div>

<div class="input-group mb-3">
  <div class="input-group-prepend">
    <span class="input-group-text">$</span>
  </div>
  <input type="text" class="form-control" aria-label="Amount (to the nearest dollar)">
  <div class="input-group-append">
    <span class="input-group-text">.00</span>
  </div>
</div>

<div class="input-group">
  <div class="input-group-prepend">
    <span class="input-group-text">With textarea</span>
  </div>
  <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 `.flexnowrap`.

```

<div class="input-group flexnowrap">
  <div class="input-group-prepend">
    <span class="input-group-text" id="addon-wrapping">@</span>
  </div>
  <input type="text" class="form-control" placeholder="Username" aria-label="Username" aria-describedby="addon-wrapping">
</div>

```

[Copy](#)

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	<input type="text"/>
Default	<input type="text"/>
Large	<input type="text"/>

Copy

```
<div class="input-group input-group-sm mb-3">
  <div class="input-group-prepend">
    <span class="input-group-text" id="inputGroup-sizing-sm">Small</span>
  </div>
  <input type="text" class="form-control" aria-label="Sizing example input"
aria-describedby="inputGroup-sizing-sm">
</div>

<div class="input-group mb-3">
  <div class="input-group-prepend">
    <span class="input-group-text" id="inputGroup-sizing-default">Default</span>
  </div>
  <input type="text" class="form-control" aria-label="Sizing example input"
aria-describedby="inputGroup-sizing-default">
</div>

<div class="input-group input-group-lg">
  <div class="input-group-prepend">
    <span class="input-group-text" id="inputGroup-sizing-lg">Large</span>
  </div>
  <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.

<input type="checkbox"/>	<input type="text"/>
<input type="radio"/>	<input type="text"/>

Copy

```
<div class="input-group mb-3">
  <div class="input-group-prepend">
    <div class="input-group-text">
      <input type="checkbox" aria-label="Checkbox for following text input">
    </div>
  </div>
  <input type="text" class="form-control" aria-label="Text input with checkbox">
</div>

<div class="input-group">
  <div class="input-group-prepend">
    <div class="input-group-text">
      <input type="radio" aria-label="Radio button for following text input">
    </div>
  </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>`.



```
<div class="input-group">
  <div class="input-group-prepend">
    <span class="input-group-text">First and last name</span>
  </div>
  <input type="text" aria-label="First name" class="form-control">
  <input type="text" aria-label="Last name" class="form-control">
</div>
```

Copy

Multiple addons

Multiple add-ons are supported and can be mixed with checkbox and radio input versions.



Copy

```
<div class="input-group mb-3">
  <div class="input-group-prepend">
    <span class="input-group-text">$</span>
    <span class="input-group-text">0.00</span>
  </div>
  <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)">
  <div class="input-group-append">
    <span class="input-group-text">$</span>
    <span class="input-group-text">0.00</span>
  </div>
</div>
```

Button addons



Copy

```

<div class="input-group mb-3">
  <div class="input-group-prepend">
    <button class="btn btn-outline-secondary" type="button" id="button-addon1">Button</button>
  </div>
  <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">
  <div class="input-group-append">
    <button class="btn btn-outline-secondary" type="button" id="button-addon2">Button</button>
  </div>
</div>

<div class="input-group mb-3">
  <div class="input-group-prepend" id="button-addon3">
    <button class="btn btn-outline-secondary" type="button">Button</button>
    <button class="btn btn-outline-secondary" type="button">Button</button>
  </div>
  <input type="text" class="form-control" placeholder="" aria-label="Example text with two button addons" aria-describedby="button-addon3">
</div>

<div class="input-group">
  <input type="text" class="form-control" placeholder="Recipient's username" aria-label="Recipient's username with two button addons" aria-describedby="button-addon4">
  <div class="input-group-append" id="button-addon4">
    <button class="btn btn-outline-secondary" type="button">Button</button>
    <button class="btn btn-outline-secondary" type="button">Button</button>
  </div>
</div>

```

Buttons with dropdowns



Copy

```

<div class="input-group mb-3">
  <div class="input-group-prepend">
    <button class="btn btn-outline-secondary dropdown-toggle" type="button"
data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">Dropdown</button>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else here</a>
      <div role="separator" class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">Separated link</a>
    </div>
  </div>
  <input type="text" class="form-control" aria-label="Text input with dropdown
button">
</div>

<div class="input-group">
  <input type="text" class="form-control" aria-label="Text input with dropdown
button">
  <div class="input-group-append">
    <button class="btn btn-outline-secondary dropdown-toggle" type="button"
data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">Dropdown</button>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else here</a>
      <div role="separator" class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">Separated link</a>
    </div>
  </div>
</div>

```

Segmented buttons

Action	▼	
		Action

Copy

```

<div class="input-group mb-3">
  <div class="input-group-prepend">
    <button type="button" class="btn btn-outline-secondary">Action</button>
    <button type="button" class="btn btn-outline-secondary dropdown-toggle dropdown-toggle-split" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
      <span class="sr-only">Toggle Dropdown</span>
    </button>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else here</a>
      <div role="separator" class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">Separated link</a>
    </div>
  </div>
  <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">
  <div class="input-group-append">
    <button type="button" class="btn btn-outline-secondary">Action</button>
    <button type="button" class="btn btn-outline-secondary dropdown-toggle dropdown-toggle-split" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
      <span class="sr-only">Toggle Dropdown</span>
    </button>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else here</a>
      <div role="separator" class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">Separated link</a>
    </div>
  </div>
</div>

```

Custom forms

Input groups include support for custom selects and custom file inputs. Browser default versions of these are not supported.

Custom select

Options	Choose...
Choose...	
Button	Choose...
Choose...	Button

Copy

```

<div class="input-group mb-3">
  <div class="input-group-prepend">
    <label class="input-group-text" for="inputGroupSelect01">Options</label>
  </div>
  <select class="custom-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="custom-select" id="inputGroupSelect02">
    <option selected>Choose...</option>
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </select>
  <div class="input-group-append">
    <label class="input-group-text" for="inputGroupSelect02">Options</label>
  </div>
</div>

<div class="input-group mb-3">
  <div class="input-group-prepend">
    <button class="btn btn-outline-secondary" type="button">Button</button>
  </div>
  <select class="custom-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="custom-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>
  <div class="input-group-append">
    <button class="btn btn-outline-secondary" type="button">Button</button>
  </div>
</div>

```

Custom file input

Upload	Choose file	Browse
Choose file		Browse Upload
Button	Choose file	Browse
Choose file		Browse Button

Copy

```

<div class="input-group mb-3">
  <div class="input-group-prepend">
    <span class="input-group-text" id="inputGroupFileAddon01">Upload</span>
  </div>
  <div class="custom-file">
    <input type="file" class="custom-file-input" id="inputGroupFile01" aria-describedby="inputGroupFileAddon01">
    <label class="custom-file-label" for="inputGroupFile01">Choose file</label>
  </div>
</div>

<div class="input-group mb-3">
  <div class="custom-file">
    <input type="file" class="custom-file-input" id="inputGroupFile02">
    <label class="custom-file-label" for="inputGroupFile02" aria-describedby="inputGroupFileAddon02">Choose file</label>
  </div>
  <div class="input-group-append">
    <span class="input-group-text" id="inputGroupFileAddon02">Upload</span>
  </div>
</div>

<div class="input-group mb-3">
  <div class="input-group-prepend">
    <button class="btn btn-outline-secondary" type="button" id="inputGroupFileAddon03">Button</button>
  </div>
  <div class="custom-file">
    <input type="file" class="custom-file-input" id="inputGroupFile03" aria-describedby="inputGroupFileAddon03">
    <label class="custom-file-label" for="inputGroupFile03">Choose file</label>
  </div>
</div>

<div class="input-group">
  <div class="custom-file">
    <input type="file" class="custom-file-input" id="inputGroupFile04" aria-describedby="inputGroupFileAddon04">
    <label class="custom-file-label" for="inputGroupFile04">Choose file</label>
  </div>
  <div class="input-group-append">
    <button class="btn btn-outline-secondary" type="button" id="inputGroupFileAddon04">Button</button>
  </div>
</div>

```

Accessibility

Screen readers will have trouble with your forms if you don't include a label for every input. For these input groups, ensure that any additional label or functionality is conveyed to assistive technologies.

The exact technique to be used (`<label>` elements hidden using the `.sr-only` class, or use of the `aria-label` and `aria-labelledby` attributes, possibly in combination with `aria-describedby`) and what additional information will need to be conveyed will vary depending on the exact type of interface widget you're implementing. The examples in this section provide a few suggested, case-specific approaches.

Jumbotron

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10
free Adobe Stock images.

ads via Carbon

A lightweight, flexible component that can optionally extend the entire viewport to showcase key marketing messages on your site.

Hello, world!

This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.

It uses utility classes for typography and spacing to space content out within the larger container.

[Learn more](#)

Copy

```
<div class="jumbotron">
  <h1 class="display-4">Hello, world!</h1>
  <p class="lead">This is a simple hero unit, a simple jumbotron-style component for calling extra
attention to featured content or information.</p>
  <hr class="my-4">
  <p>It uses utility classes for typography and spacing to space content out within the larger container.
</p>
  <a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a>
</div>
```

To make the jumbotron full width, and without rounded corners, add the `.jumbotron-fluid` modifier class and add a `.container` or `.container-fluid` within.

Fluid jumbotron

This is a modified jumbotron that occupies the entire horizontal space of its parent.

Copy

```
<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h1 class="display-4">Fluid jumbotron</h1>
    <p class="lead">This is a modified jumbotron that occupies the entire horizontal space of its parent.
  </p>
  </div>
</div>
```

List group

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Toolips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

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.

Cras justo odio

Dapibus ac facilisis in

Morbi leo risus

Porta ac consectetur ac

Vestibulum at eros

Copy

```
<ul class="list-group">
  <li class="list-group-item">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
  <li class="list-group-item">Porta ac consectetur ac</li>
  <li class="list-group-item">Vestibulum at eros</li>
</ul>
```

Active items

Add `.active` to a `.list-group-item` to indicate the current active selection.

Cras justo odio

Dapibus ac facilisis in

Morbi leo risus

Porta ac consectetur ac

Vestibulum at eros

Copy

```
<ul class="list-group">
  <li class="list-group-item active">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
  <li class="list-group-item">Porta ac consectetur ac</li>
  <li class="list-group-item">Vestibulum at eros</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).

Cras justo odio

Dapibus ac facilisis in

Morbi leo risus

Porta ac consectetur ac

Vestibulum at eros

```
<ul class="list-group">
  <li class="list-group-item disabled" aria-disabled="true">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
  <li class="list-group-item">Porta ac consectetur ac</li>
  <li class="list-group-item">Vestibulum at eros</li>
</ul>
```

Copy

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 ``s or `<div>`s) don't provide a click or tap affordance.

Be sure to **not use the standard `.btn` classes here**.

Cras justo odio

Dapibus ac facilisis in

Morbi leo risus

Porta ac consectetur ac

Vestibulum at eros

Copy

```
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action active">
    Cras justo odio
  </a>
  <a href="#" class="list-group-item list-group-item-action">Dapibus ac
facilisis in</a>
  <a href="#" class="list-group-item list-group-item-action">Morbi leo risus</a>
  <a href="#" class="list-group-item list-group-item-action">Porta ac
consectetur ac</a>
  <a href="#" class="list-group-item list-group-item-action disabled"
tabindex="-1" aria-disabled="true">Vestibulum at eros</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.

Cras justo odio

Dapibus ac facilisis in

Morbi leo risus

Porta ac consectetur ac

Vestibulum at eros

Copy

```
<div class="list-group">
  <button type="button" class="list-group-item list-group-item-action active">
    Cras justo odio
  </button>
  <button type="button" class="list-group-item list-group-item-action">Dapibus
ac facilisis in</button>
  <button type="button" class="list-group-item list-group-item-action">Morbi leo
risus</button>
  <button type="button" class="list-group-item list-group-item-action">Porta ac
consectetur ac</button>
  <button type="button" class="list-group-item list-group-item-action"
disabled>Vestibulum at eros</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).

Cras justo odio

Dapibus ac facilisis in

Morbi leo risus

Porta ac consectetur ac

Vestibulum at eros

Copy

```
<ul class="list-group list-group-flush">
  <li class="list-group-item">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
  <li class="list-group-item">Porta ac consectetur ac</li>
  <li class="list-group-item">Vestibulum at eros</li>
</ul>
```

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|x1}` 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.

```
Cras justo odio Dapibus ac facilisis in Morbi leo risus
```

Copy

```
<ul class="list-group list-group-horizontal">
  <li class="list-group-item">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
</ul>
```

```
Cras justo odio Dapibus ac facilisis in Morbi leo risus
```

Copy

```
<ul class="list-group list-group-horizontal-sm">
  <li class="list-group-item">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
</ul>
```

```
Cras justo odio Dapibus ac facilisis in Morbi leo risus
```

Copy

```
<ul class="list-group list-group-horizontal-md">
  <li class="list-group-item">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
</ul>
```

```
Cras justo odio
```

```
Dapibus ac facilisis in
```

```
Morbi leo risus
```

Copy

```
<ul class="list-group list-group-horizontal-lg">
  <li class="list-group-item">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
</ul>
```

Cras justo odio

Dapibus ac facilisis in

Morbi leo risus

Copy

```
<ul class="list-group list-group-horizontal-xl">
  <li class="list-group-item">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
</ul>
```

Contextual classes

Use contextual classes to style list items with a stateful background and color.

Dapibus ac facilisis in

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

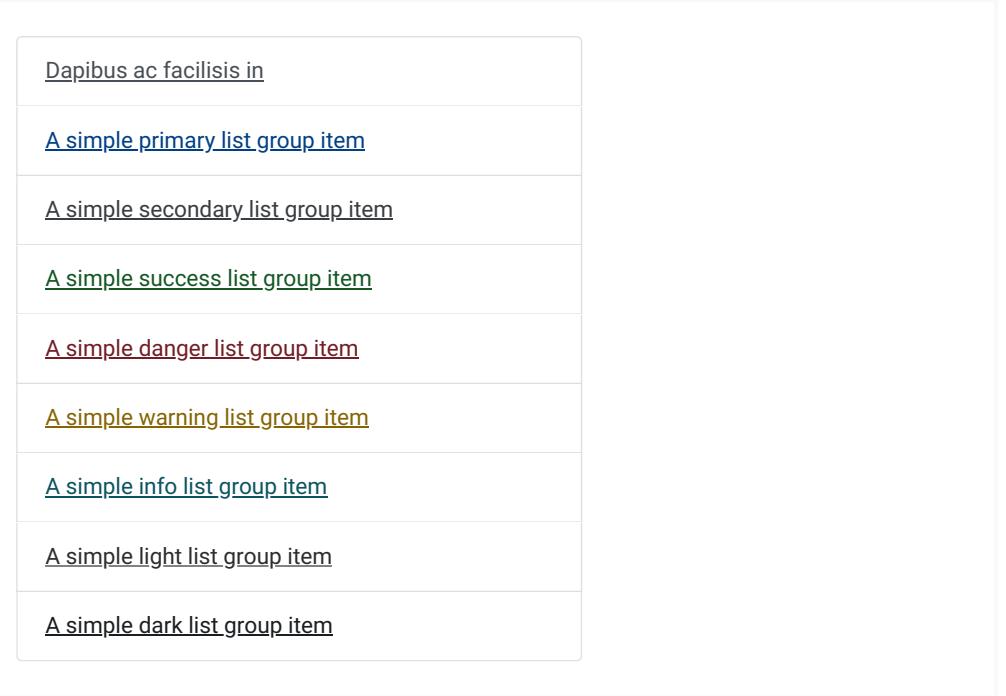
A simple dark list group item

Copy

```
<ul class="list-group">
  <li class="list-group-item">Dapibus ac facilisis in</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.



Dapibus ac facilisis in
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

Copy

```

<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action">Dapibus ac
facilisis in</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 `.sr-only` class.

With badges

Add badges to any list group item to show unread counts, activity, and more with the help of some [utilities](#).

Cras justo odio	(14)
Dapibus ac facilisis in	(2)
Morbi leo risus	(1)

```

<ul class="list-group">
  <li class="list-group-item d-flex justify-content-between align-items-center">
    Cras justo odio
    <span class="badge badge-primary badge-pill">14</span>
  </li>
  <li class="list-group-item d-flex justify-content-between align-items-center">
    Dapibus ac facilisis in
    <span class="badge badge-primary badge-pill">2</span>
  </li>
  <li class="list-group-item d-flex justify-content-between align-items-center">
    Morbi leo risus
    <span class="badge badge-primary badge-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

Donec id elit non mi porta gravida at eget metus.
Maecenas sed diam eget risus varius blandit.
Donec id elit non mi porta.

List group item heading 3 days ago

Donec id elit non mi porta gravida at eget metus.
Maecenas sed diam eget risus varius blandit.
Donec id elit non mi porta.

List group item heading 3 days ago

Donec id elit non mi porta gravida at eget metus.
Maecenas sed diam eget risus varius blandit.
Donec id elit non mi porta.

Copy

```
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action active">
    <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">Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit.</p>
    <small>Donec id elit non mi porta.</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">Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit.</p>
    <small class="text-muted">Donec id elit non mi porta.</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">Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit.</p>
    <small class="text-muted">Donec id elit non mi porta.</small>
  </a>
</div>
```

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

Velit aute mollit ipsum ad dolor consectetur nulla officia culpa adipisicing exercitation fugiat tempor. Voluptate deserunt sit sunt nisi aliqua fugiat proident ea ut. Mollit voluptate reprehenderit occaecat nisi ad non minim tempor sunt voluptate consectetur exercitation id ut nulla. Ea et

Settings

fugiat aliquip nostrud sunt incididunt consectetur culpa
aliquip eiusmod dolor. Anim ad Lorem aliqua in cupidatat nisi
enim eu nostrud do aliquip veniam minim.

```
<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-toggle="list" href="#list-home" role="tab" aria-controls="home">Home</a>
      <a class="list-group-item list-group-item-action" id="list-profile-list" data-toggle="list" href="#list-profile" role="tab" aria-controls="profile">Profile</a>
      <a class="list-group-item list-group-item-action" id="list-messages-list" data-toggle="list" href="#list-messages" role="tab" aria-controls="messages">Messages</a>
      <a class="list-group-item list-group-item-action" id="list-settings-list" data-toggle="list" href="#list-settings" role="tab" aria-controls="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>
```

Copy

Using data attributes

You can activate a list group navigation without writing any JavaScript by simply specifying `data-toggle="list"` or on an element. Use these data attributes on `.list-group-item`.

```
<!-- List group -->
<div class="list-group" id="myList" role="tablist">
  <a class="list-group-item list-group-item-action active" data-toggle="list" href="#home" role="tab">Home</a>
  <a class="list-group-item list-group-item-action" data-toggle="list" href="#profile" role="tab">Profile</a>
  <a class="list-group-item list-group-item-action" data-toggle="list" href="#messages" role="tab">Messages</a>
  <a class="list-group-item list-group-item-action" data-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>
```

Copy

Via JavaScript

Enable tabbable list item via JavaScript (each list item needs to be activated individually):

Copy

```
$(' myList a').on('click', function (e) {
  e.preventDefault()
  $(this).tab('show')
})
```

You can activate individual list item in several ways:

```
$('#myList a[href="#profile"]').tab('show') // Select tab by name
$('#myList a:first-child').tab('show') // Select first tab
$('#myList a:last-child').tab('show') // Select last tab
$('#myList a:nth-child(3)').tab('show') // Select third tab
```

[Copy](#)

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>
```

[Copy](#)

Methods

`$().tab`

Activates a list item element and content container. Tab should have either a `data-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-toggle="list"
  href="#home" role="tab">Home</a>
  <a class="list-group-item list-group-item-action" data-toggle="list"
  href="#profile" role="tab">Profile</a>
  <a class="list-group-item list-group-item-action" data-toggle="list"
  href="#messages" role="tab">Messages</a>
  <a class="list-group-item list-group-item-action" data-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>
  $(function () {
    $('#myList a:last-child').tab('show')
  })
</script>
```

[Copy](#)

`.tab('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).

```
$('#someListItem').tab('show')
```

[Copy](#)

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
<code>show.bs.tab</code>	This event fires on tab show, but before the new tab has been shown. Use <code>event.target</code> and <code>event.relatedTarget</code> to target the active tab and the previous active tab (if available) respectively.
<code>shown.bs.tab</code>	This event fires on tab show after a tab has been shown. Use <code>event.target</code> and <code>event.relatedTarget</code> to target the active tab and the previous active tab (if available) respectively.
<code>hide.bs.tab</code>	This event fires when a new tab is to be shown (and thus the previous active tab is to be hidden). Use <code>event.target</code> and <code>event.relatedTarget</code> to target the current active tab and the new soon-to-be-active tab, respectively.
<code>hidden.bs.tab</code>	This event fires after a new tab is shown (and thus the previous active tab is hidden). Use <code>event.target</code> and <code>event.relatedTarget</code> to target the previous active tab and the new active tab, respectively.

```
$('a[data-toggle="list"]').on('shown.bs.tab', function (e) {  
    e.target // newly activated tab  
    e.relatedTarget // previous active tab  
})
```

Copy

Media object

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10
free Adobe Stock images.

ads via Carbon

Example

The [media object](#) helps build complex and repetitive components where some media is positioned alongside content that doesn't wrap around said media. Plus, it does this with only two required classes thanks to flexbox.

Below is an example of a single media object. Only two classes are required—the wrapping `.media` and the `.media-body` around your content. Optional padding and margin can be controlled through [spacing utilities](#).

64x64

Media heading

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

```
<div class="media">
  
  <div class="media-body">
    <h5 class="mt-0">Media heading</h5>
    Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.
  </div>
</div>
```

Copy

Flexbug #12: Inline elements aren't treated as flex items

Internet Explorer 10-11 do not render inline elements like links or images (or `::before` and `::after` pseudo-elements) as flex items. The only workaround is to set a non-inline `display` value (e.g., `block`, `inline-block`, or `flex`). We suggest using `.d-flex`, one of our [display utilities](#), as an easy fix.

Source: [Flexbugs on GitHub](#)

Nesting

Media objects can be infinitely nested, though we suggest you stop at some point. Place nested `.media` within the `.media-body` of a parent media object.



Media heading

64x64

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

64x64

Media heading

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

Copy

```
<div class="media">
  
  <div class="media-body">
    <h5 class="mt-0">Media heading</h5>
    Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

    <div class="media mt-3">
      <a class="mr-3" href="#">
        
      </a>
      <div class="media-body">
        <h5 class="mt-0">Media heading</h5>
        Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.
      </div>
    </div>
  </div>
</div>
```

Alignment

Media in a media object can be aligned with flexbox utilities to the top (default), middle, or end of your `.media-body` content.

64x64

Top-aligned media

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Copy

```
<div class="media">
  
  <div class="media-body">
    <h5 class="mt-0">Top-aligned media</h5>
    <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.</p>
    <p>Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.</p>
  </div>
</div>
```

Center-aligned media

64x64

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

```
<div class="media">
  
  <div class="media-body">
    <h5 class="mt-0">Center-aligned media</h5>
    <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.</p>
    <p class="mb-0">Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.</p>
  </div>
</div>
```

Bottom-aligned media

64x64

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

```
<div class="media">
  
  <div class="media-body">
    <h5 class="mt-0">Bottom-aligned media</h5>
    <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.</p>
    <p class="mb-0">Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.</p>
  </div>
</div>
```

Order

Change the order of content in media objects by modifying the HTML itself, or by adding some custom flexbox CSS to set the `order` property (to an integer of your choosing).

Media object

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

64x64

Copy

```
<div class="media">
  <div class="media-body">
    <h5 class="mt-0 mb-1">Media object</h5>
    Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.
  </div>
  
</div>
```

Media list

Because the media object has so few structural requirements, you can also use these classes on list HTML elements. On your `` or ``, add the `.list-unstyled` to remove any browser default list styles, and then apply `.media` to your ``s. As always, use spacing utilities wherever needed to fine tune.

64x64

List-based media object

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

64x64

List-based media object

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

64x64

List-based media object

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

Copy

```
<ul class="list-unstyled">
  <li class="media">
    
    <div class="media-body">
      <h5 class="mt-0 mb-1">List-based media object</h5>
      Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.
    </div>
  </li>
  <li class="media my-4">
    
    <div class="media-body">
      <h5 class="mt-0 mb-1">List-based media object</h5>
      Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.
    </div>
  </li>
  <li class="media">
    
    <div class="media-body">
      <h5 class="mt-0 mb-1">List-based media object</h5>
      Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.
    </div>
  </li>
</ul>
```

Modal

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

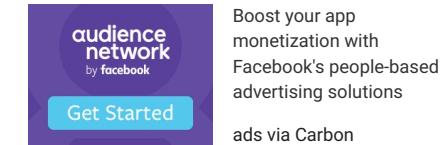
[Toolips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



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:

```
$('#myModal').on('shown.bs.modal', function () {
  $('#myInput').trigger('focus')
})
```

Copy

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

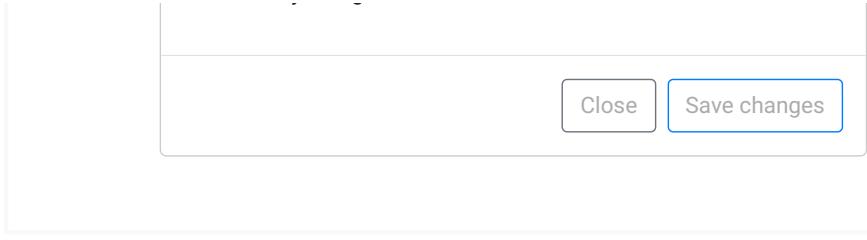
Modal components

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

x

Modal body text goes here.



Close

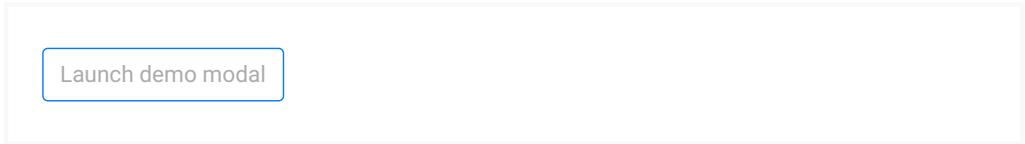
Save changes

Copy

```
<div class="modal" tabindex="-1" role="dialog">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Modal title</h5>
        <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
          <span aria-hidden="true">&times;</span>
        </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-
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.



Launch demo modal

Copy

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">
  Launch demo modal
</button>

<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" 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="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-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 it. Click the button below to try it.

Launch static backdrop modal

Copy

```

<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#staticBackdrop">
  Launch static backdrop modal
</button>

<!-- Modal -->
<div class="modal fade" id="staticBackdrop" data-backdrop="static" data-keyboard="false" tabindex="-1" role="dialog" 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="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-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.

[Launch demo modal](#)

You can also create a scrollable modal that allows scroll the modal body by adding `.modal-dialog-scrollable` to `.modal-dialog`.

[Launch demo modal](#)

```

<!-- Scrollable modal -->
<div class="modal-dialog modal-dialog-scrollable">
  ...
</div>

```

[Copy](#)

Vertically centered

Add `.modal-dialog-centered` to `.modal-dialog` to vertically center the modal.

Vertically centered modal

Vertically centered scrollable modal

[Copy](#)

```
<!-- 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.

Launch demo modal

Copy

```
<div class="modal-body">
  <h5>Popover in a modal</h5>
  <p>This <a href="#" role="button" class="btn btn-secondary popover-test" title="Popover title" data-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="#" class="tooltip-test" title="Tooltip">This link</a> and <a href="#" class="tooltip-test" 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.

Launch demo modal

Copy

```
<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 ml-auto">.col-md-4 .ml-auto</div>
    </div>
    <div class="row">
      <div class="col-md-3 ml-auto">.col-md-3 .ml-auto</div>
      <div class="col-md-2 ml-auto">.col-md-2 .ml-auto</div>
    </div>
    <div class="row">
      <div class="col-md-6 ml-auto">.col-md-6 .ml-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-* attributes](#) (possibly [via jQuery](#)) 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`.

Open modal for @mdo

Open modal for @fat

Open modal for @getbootstrap

Copy

```

<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal" data-whatever="@mdo">Open modal for @mdo</button>
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal" data-whatever="@fat">Open modal for @fat</button>
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal" data-whatever="@getbootstrap">Open modal for @getbootstrap</button>

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" 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="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <form>
          <div class="form-group">
            <label for="recipient-name" class="col-form-label">Recipient:</label>
            <input type="text" class="form-control" id="recipient-name">
          </div>
          <div class="form-group">
            <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-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Send message</button>
      </div>
    </div>
  </div>

```

Copy

```

$( '#exampleModal' ).on('show.bs.modal', function (event) {
  var button = $(event.relatedTarget) // Button that triggered the modal
  var recipient = button.data('whatever') // Extract info from data-* attributes
  // If necessary, you could initiate an AJAX request here (and then do the
  // updating in a callback).
  // Update the modal's content. We'll use jQuery here, but you could use a data
  // binding library or other methods instead.
  var modal = $(this)
  modal.find('.modal-title').text('New message to ' + recipient)
  modal.find('.modal-body input').val(recipient)
})

```

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" role="dialog" aria-labelledby="..." aria-hidden="true">
  ...
</div>
```

Dynamic heights

If the height of a modal changes while it is open, you should call `$('#myModal').modal('handleUpdate')` to readjust the modal's position in case a scrollbar appears.

Accessibility

Be sure to add `role="dialog"` and `aria-labelledby="..."`, referencing the modal title, to `.modal`. Additionally, you may give a description of your modal dialog with `aria-describedby` on `.modal`.

Embedding YouTube videos

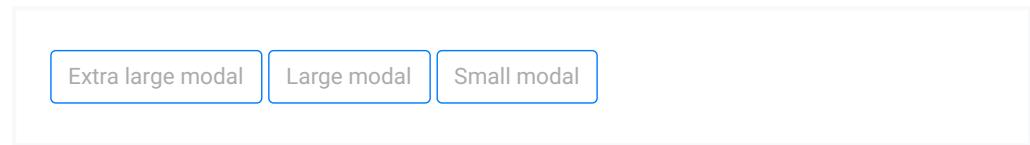
Embedding YouTube videos in modals requires additional JavaScript not in Bootstrap to automatically stop playback and more. [See this helpful Stack Overflow post](#) for more information.

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	<code>.modal-sm</code>	300px
Default	<code>None</code>	500px
Large	<code>.modal-lg</code>	800px
Extra large	<code>.modal-xl</code>	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>
```

Copy

Usage

The modal plugin toggles your hidden content on demand, via data attributes or JavaScript. It also adds `.modal-open` to the `<body>` to override 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

Activate a modal without writing JavaScript. Set `data-toggle="modal"` on a controller element, like a button, along with a `data-target="#foo"` or `href="#foo"` to target a specific modal to toggle.

```
<button type="button" data-toggle="modal" data-target="#myModal">Launch
modal</button>
```

Copy

Via JavaScript

Call a modal with id `myModal` with a single line of JavaScript:

```
$( '#myModal' ).modal(options)
```

Copy

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-backdrop=""`.

Name	Type	Default	Description
backdrop	boolean or the string ' <code>static</code> '	true	Includes a modal-backdrop element. Alternatively, specify <code>static</code> for a backdrop which doesn't close the modal on click.
keyboard	boolean	true	Closes the modal when escape key is pressed
focus	boolean	true	Puts the focus on the modal when initialized.
show	boolean	true	Shows 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.](#)

`.modal(options)`

Activates your content as a modal. Accepts an optional options `object`.

```
$( '#myModal' ).modal({  
  keyboard: false  
})
```

Copy

`.modal('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).

```
$( '#myModal' ).modal('toggle')
```

Copy

`.modal('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).

```
$( '#myModal' ).modal('show')
```

Copy

`.modal('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).

```
$( '#myModal' ).modal('hide')
```

Copy

.modal('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).

```
$( '#myModal' ).modal('handleUpdate')
```

Copy

.modal('dispose')

Destroys an element's modal.

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 Type	Description
show.bs.modal	This event fires immediately when the <code>show</code> instance method is called. If caused by a click, the clicked element is available as the <code>relatedTarget</code> 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 <code>relatedTarget</code> property of the event.
hide.bs.modal	This event is fired immediately when the <code>hide</code> 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 <code>static</code> and a click outside the modal or an escape key press is performed with the <code>keyboard</code> option or <code>data-keyboard</code> set to <code>false</code> .

```
$( '#myModal' ).on('hidden.bs.modal', function (e) {  
    // do something...  
})
```

Copy

Navbar

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Toolips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Adobe Creative Cloud for
Teams starting at \$33.99
per month.

ads via Carbon

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}` for responsive collapsing and [color scheme](#) classes.
- Navbars and their contents are fluid by default. Use [optional containers](#) to limit their horizontal width.
- 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.
- Navbars are hidden by default when printing. Force them to be printed by adding `.d-print` to the `.navbar`. See the [display](#) utility class.
- 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.

The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the [reduced motion section of our accessibility documentation](#).

Read on for an example and list of supported sub-components.

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.
- `.form-inline` 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.

Here's an example of all the sub-components included in a responsive light-themed navbar that automatically collapses at the `lg` (large) breakpoint.

Copy

```

<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-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 mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span>
      </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-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Dropdown
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>

```

This example uses [color](#) (`bg-light`) and [spacing](#) (`my-2, my-lg-0, mr-sm-0, my-sm-0`) 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.

```

<!-- As a link -->
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
</nav>

<!-- As a heading -->
<nav class="navbar navbar-light bg-light">
  <span class="navbar-brand mb-0 h1">Navbar</span>
</nav>

```

[Copy](#)

Adding images to the `.navbar-brand` will likely always require custom styles or utilities to properly size. Here are some examples to demonstrate.

```
<!-- Just an image -->
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">
    
  </a>
</nav>
```

Copy

```
<!-- Image and text -->
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">
    
      Bootstrap
  </a>
</nav>
```

Copy

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.

Active states—with `.active`—to indicate the current page can be applied directly to `.nav-links` or their immediate parent `.nav-items`.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-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 active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span>
      </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" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
  </div>
</nav>
```

Copy

And because we use classes for our navs, you can avoid the list-based approach entirely if you like.

Copy

```

<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-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-item nav-link active" href="#">Home <span class="sr-only">(current)</span></a>
      <a class="nav-item nav-link" href="#">Features</a>
      <a class="nav-item nav-link" href="#">Pricing</a>
      <a class="nav-item nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
    </div>
  </div>
</nav>

```

You may also utilize dropdowns in your navbar nav. 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.

```

<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-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 active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></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-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Dropdown link
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li>
    </ul>
  </div>
</nav>

```

Forms

Place various form controls and components within a navbar with `.form-inline`.

[Copy](#)

[Copy](#)

```
<nav class="navbar navbar-light bg-light">
  <form class="form-inline">
    <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
  </form>
</nav>
```

Immediate children elements in `.navbar` use flex layout and will default to `justify-content: space-between`. Use additional [flex utilities](#) as needed to adjust this behavior.

```
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand">Navbar</a>
  <form class="form-inline">
    <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
  </form>
</nav>
```

Copy

Input groups work, too:

```
<nav class="navbar navbar-light bg-light">
  <form class="form-inline">
    <div class="input-group">
      <div class="input-group-prepend">
        <span class="input-group-text" id="basic-addon1">@</span>
      </div>
      <input type="text" class="form-control" placeholder="Username" aria-label="Username" aria-describedby="basic-addon1">
    </div>
  </form>
</nav>
```

Copy

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.

```
<nav class="navbar navbar-light bg-light">
  <form class="form-inline">
    <button class="btn btn-outline-success" type="button">Main button</button>
    <button class="btn btn-sm btn-outline-secondary" type="button">Smaller button</button>
  </form>
</nav>
```

Copy

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.

Copy



```
<nav class="navbar navbar-light bg-light">
  <span class="navbar-text">
    Navbar text with an inline element
  </span>
</nav>
```

Mix and match with other components and utilities as needed.



```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar w/ text</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-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 mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span>
      </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>
</nav>
```

Color schemes

Theming the navbar has never been easier thanks to the combination of theming classes and `background-color` utilities. Choose from `.navbar-light` for use with light background colors, or `.navbar-dark` for dark background colors. Then, customize with `.bg-*` utilities.



```
<nav class="navbar navbar-dark bg-dark">
  <!-- Navbar content -->
</nav>

<nav class="navbar navbar-dark bg-primary">
  <!-- Navbar content -->
</nav>

<nav class="navbar navbar-light" 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 or add one within to only center the contents of a [fixed or static top navbar](#).

Copy

```
<div class="container">
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Navbar</a>
  </nav>
</div>
```

When the container is within your navbar, its horizontal padding is removed at breakpoints lower than your specified `.navbar-expand{-sm|-md|-lg|-xl}` class. This ensures we're not doubling up on padding unnecessarily on lower viewports when your navbar is collapsed.

Copy

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container">
    <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, or stickied to the top (scrolls with the page until it reaches the top, 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.

Also note that `.sticky-top` uses `position: sticky`, which [isn't fully supported in every browser](#).

Copy

```
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">Default</a>
</nav>
```

Copy

```
<nav class="navbar fixed-top navbar-light bg-light">
  <a class="navbar-brand" href="#">Fixed top</a>
</nav>
```

Copy

```
<nav class="navbar fixed-bottom navbar-light bg-light">
  <a class="navbar-brand" href="#">Fixed bottom</a>
</nav>
```

Copy

```
<nav class="navbar sticky-top navbar-light bg-light">
  <a class="navbar-brand" href="#">Sticky top</a>
</nav>
```

Responsive behaviors

Navbars can utilize `.navbar-toggler`, `.navbar-collapse`, and `.navbar-expand{-sm|-md|-lg|-xl}` classes to change 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 in lowest breakpoint:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-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 mr-auto mt-2 mt-lg-0">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span>
      </a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>
```

Copy

With a brand name shown on the left and toggler on the right:

Copy

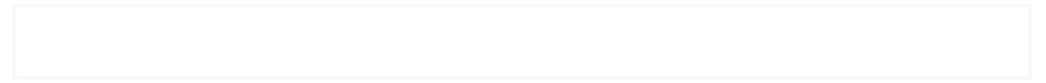
```

<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-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 mr-auto mt-2 mt-lg-0">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span>
      </a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>

```

With a toggler on the left and brand name on the right:



Copy

```

<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-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 mr-auto mt-2 mt-lg-0">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span>
      </a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>

```

External content

Sometimes you want to use the collapse plugin to trigger hidden content elsewhere on the page. Because our plugin works on the `id` and `data-target` matching, that's easily done!

```
<div class="fixed-top">
  <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">
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarToggleExternalContent" aria-
controls="navbarToggleExternalContent" aria-expanded="false" aria-label="Toggle
navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
  </nav>
</div>
```

Copy

Navs

Search...

Getting started

[Layout](#)

[Content](#)

Components

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

Navs

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

Utilities

Extend

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

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.

[Active](#) [Link](#) [Link](#) [Disabled](#)

```
<ul class="nav">
  <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" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
  </li>
</ul>
```

Copy

[Active](#) [Link](#) [Link](#) [Disabled](#)

```
<nav class="nav">
  <a class="nav-link active" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
</nav>
```

Copy

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](#)

Copy

```
<ul class="nav justify-content-center">
  <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" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
  </li>
</ul>
```

Right-aligned with `.justify-content-end`:

[Active](#) [Link](#) [Link](#) [Disabled](#)

Copy

```
<ul class="nav justify-content-end">
  <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" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">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](#)

Copy

```
<ul class="nav flex-column">
  <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" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
  </li>
</ul>
```

As always, vertical navigation is possible without ``s, too.

[Active](#)

[Link](#)

[Link](#)

[Disabled](#)

Copy

```
<nav class="nav flex-column">
  <a class="nav-link active" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">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](#)

Copy

```
<ul class="nav nav-tabs">
  <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" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
  </li>
</ul>
```

Pills

Take that same HTML, but use `.nav-pills` instead:

Active [Link](#) [Link](#) [Disabled](#)

Copy

```
<ul class="nav nav-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" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">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-items`, 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](#)

Copy

```
<ul class="nav nav-pills nav-fill">
  <li class="nav-item">
    <a class="nav-link active" 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" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
  </li>
</ul>
```

When using a `<nav>`-based navigation, be sure to include `.nav-item` on the anchors.

Active	Much longer nav link	Link	Disabled
--------	--------------------------------------	----------------------	--------------------------

```
<nav class="nav nav-pills nav-fill">
  <a class="nav-item nav-link active" href="#">Active</a>
  <a class="nav-item nav-link" href="#">Much longer nav link</a>
  <a class="nav-item nav-link" href="#">Link</a>
  <a class="nav-item nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
</nav>
```

Copy

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
--------	--	----------------------	--------------------------

```
<ul class="nav nav-pills nav-justified">
  <li class="nav-item">
    <a class="nav-link active" 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" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
  </li>
</ul>
```

Copy

Similar to the `.nav-fill` example using a `<nav>`-based navigation, be sure to include `.nav-item` on the anchors.

Active	Much longer nav link	Link	Disabled
--------	--	----------------------	--------------------------

```
<nav class="nav nav-pills nav-justified">
  <a class="nav-item nav-link active" href="#">Active</a>
  <a class="nav-item nav-link" href="#">Much longer nav link</a>
  <a class="nav-item nav-link" href="#">Link</a>
  <a class="nav-item nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
</nav>
```

Copy

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](#)

```
<nav class="nav nav-pills flex-column flex-sm-row">
  <a class="flex-sm-fill text-sm-center nav-link active" 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" href="#">
    tabindex="-1" aria-disabled="true">Disabled</a>
</nav>
```

Copy

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 ``, or wrap a `<nav>` element around the whole navigation. Do not add the role to the `` 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 [WAI \(Web Accessibility Initiative\) ARIA \(Accessible Rich Internet Applications\) Authoring Practices](#). See [JavaScript behavior](#) for dynamic tabbed interfaces in this section for an example.

Using dropdowns

Add dropdown menus with a little extra HTML and the [dropdowns JavaScript plugin](#).

Tabs with dropdowns

[Active](#) [Dropdown ▾](#) [Link](#) [Disabled](#)

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false">Dropdown</a>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else here</a>
      <div class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">Separated link</a>
    </div>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

Copy

Pills with dropdowns

Active [Dropdown ▾](#) [Link](#) [Disabled](#)

Copy

```
<ul class="nav nav-pills">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false">Dropdown</a>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else here</a>
      <div class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">Separated link</a>
    </div>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

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, even via dropdown menus.

If you're building our JavaScript from source, it [requires util.js](#).

Dynamic tabbed interfaces, as described in the [WAI \(Web Accessibility Initiative\) ARIA \(Accessible Rich Internet Applications\) Authoring Practices](#), 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).

Note that dynamic tabbed interfaces should *not* contain dropdown menus, as this causes 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.

[Home](#) [Profile](#) [Contact](#)

Copy

Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica. Reprehenderit butcher retro keffiyeh dreamcatcher synth. Cosby sweater eu banh mi, qui irure terry richardson ex squid. Aliquip placeat salvia cillum iphone. Seitan aliquip quis cardigan american apparel, butcher voluptate nisi qui.

```

<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <a class="nav-link active" id="home-tab" data-toggle="tab" href="#home" role="tab" aria-controls="home" aria-selected="true">Home</a>
  </li>
  <li class="nav-item" role="presentation">
    <a class="nav-link" id="profile-tab" data-toggle="tab" href="#profile" role="tab" aria-controls="profile" aria-selected="false">Profile</a>
  </li>
  <li class="nav-item" role="presentation">
    <a class="nav-link" id="contact-tab" data-toggle="tab" href="#contact" role="tab" aria-controls="contact" aria-selected="false">Contact</a>
  </li>
</ul>
<div class="tab-content" id="myTabContent">
  <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="home-tab">...</div>
  <div class="tab-pane fade" id="profile" role="tabpanel" aria-labelledby="profile-tab">...</div>
  <div class="tab-pane fade" id="contact" role="tabpanel" aria-labelledby="contact-tab">...</div>
</div>

```

To help fit your needs, this works with ``-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.

The screenshot shows a user interface with a navigation bar at the top containing three tabs: "Home", "Profile", and "Contact". The "Home" tab is currently selected, indicated by a thicker border and a blue background. Below the tabs, there is a large amount of placeholder text in Latin: "Et et consectetur ipsum labore excepteur est proident excepteur ad velit occaecat qui minim occaecat veniam. Fugiat veniam incididunt anim aliqua enim pariatur veniam sunt est aute sit dolor anim. Velit non irure adipisicing aliqua ullamco irure incididunt irure non esse consectetur nostrud minim non minim occaecat. Amet duis do nisi duis veniam non est eiusmod tempor incididunt tempor dolor ipsum in qui sit. Exercitation mollit sit culpa nisi culpa non adipisicing reprehenderit do dolore. Duis reprehenderit occaecat anim ullamco ad duis occaecat ex."

```

<nav>
  <div class="nav nav-tabs" id="nav-tab" role="tablist">
    <a class="nav-item nav-link active" id="nav-home-tab" data-toggle="tab" href="#nav-home" role="tab" aria-controls="nav-home" aria-selected="true">Home</a>
    <a class="nav-item nav-link" id="nav-profile-tab" data-toggle="tab" href="#nav-profile" role="tab" aria-controls="nav-profile" aria-selected="false">Profile</a>
    <a class="nav-item nav-link" id="nav-contact-tab" data-toggle="tab" href="#nav-contact" role="tab" aria-controls="nav-contact" aria-selected="false">Contact</a>
  </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">...</div>
  <div class="tab-pane fade" id="nav-profile" role="tabpanel" aria-labelledby="nav-profile-tab">...</div>
  <div class="tab-pane fade" id="nav-contact" role="tabpanel" aria-labelledby="nav-contact-tab">...</div>
</div>

```

The tabs plugin also works with pills.

Consequat occaecat ullamco amet non eiusmod nostrud dolore irure incididunt est duis anim sunt officia. Fugiat velit proident aliquip nisi incididunt nostrud exercitation proident est nisi. Irure magna elit commodo anim ex veniam culpa eiusmod id nostrud sit cupidatat in veniam ad. Eiusmod consequat eu adipisicing minim anim aliquip cupidatat culpa excepteur quis. Occaecat sit eu exercitation irure Lorem incididunt nostrud.

```
<ul class="nav nav-pills mb-3" id="pills-tab" role="tablist">
  <li class="nav-item" role="presentation">
    <a class="nav-link active" id="pills-home-tab" data-toggle="pill"
      href="#pills-home" role="tab" aria-controls="pills-home" aria-
      selected="true">Home</a>
  </li>
  <li class="nav-item" role="presentation">
    <a class="nav-link" id="pills-profile-tab" data-toggle="pill" href="#pills-
      profile" role="tab" aria-controls="pills-profile" aria-
      selected="false">Profile</a>
  </li>
  <li class="nav-item" role="presentation">
    <a class="nav-link" id="pills-contact-tab" data-toggle="pill" href="#pills-
      contact" role="tab" aria-controls="pills-contact" aria-
      selected="false">Contact</a>
  </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">...</div>
  <div class="tab-pane fade" id="pills-profile" role="tabpanel" aria-
    labelledby="pills-profile-tab">...</div>
  <div class="tab-pane fade" id="pills-contact" role="tabpanel" aria-
    labelledby="pills-contact-tab">...</div>
</div>
```

Copy

And with vertical pills.

[Home](#)[Profile](#)[Messages](#)[Settings](#)

Cillum ad ut irure tempor velit nostrud occaecat ullamco aliqua anim
Lorem sint. Veniam sint duis incididunt do esse magna mollit
excepteur laborum qui. Id id reprehenderit sit est eu aliqua occaecat
quis et velit excepteur laborum mollit dolore eiusmod. Ipsum dolor in
occaecat commodo et voluptate minim reprehenderit mollit pariatur.
Deserunt non laborum enim et cillum eu deserunt excepteur ea
incididunt minim occaecat.

Copy

```
<div class="row">
  <div class="col-3">
    <div class="nav flex-column nav-pills" id="v-pills-tab" role="tablist" aria-orientation="vertical">
      <a class="nav-link active" id="v-pills-home-tab" data-toggle="pill" href="#v-pills-home" role="tab" aria-controls="v-pills-home" aria-selected="true">Home</a>
      <a class="nav-link" id="v-pills-profile-tab" data-toggle="pill" href="#v-pills-profile" role="tab" aria-controls="v-pills-profile" aria-selected="false">Profile</a>
      <a class="nav-link" id="v-pills-messages-tab" data-toggle="pill" href="#v-pills-messages" role="tab" aria-controls="v-pills-messages" aria-selected="false">Messages</a>
      <a class="nav-link" id="v-pills-settings-tab" data-toggle="pill" href="#v-pills-settings" role="tab" aria-controls="v-pills-settings" aria-selected="false">Settings</a>
    </div>
  </div>
  <div class="col-9">
    <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">...</div>
      <div class="tab-pane fade" id="v-pills-profile" role="tabpanel" aria-labelledby="v-pills-profile-tab">...</div>
      <div class="tab-pane fade" id="v-pills-messages" role="tabpanel" aria-labelledby="v-pills-messages-tab">...</div>
      <div class="tab-pane fade" id="v-pills-settings" role="tabpanel" aria-labelledby="v-pills-settings-tab">...</div>
    </div>
  </div>
</div>
```

Using data attributes

You can activate a tab or pill navigation without writing any JavaScript by simply specifying `data-toggle="tab"` or `data-toggle="pill"` on an element. Use these data attributes on `.nav-tabs` or `.nav-pills`.

Copy

```

<!-- Nav tabs -->
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <a class="nav-link active" id="home-tab" data-toggle="tab" href="#home" role="tab" aria-controls="home" aria-selected="true">Home</a>
  </li>
  <li class="nav-item" role="presentation">
    <a class="nav-link" id="profile-tab" data-toggle="tab" href="#profile" role="tab" aria-controls="profile" aria-selected="false">Profile</a>
  </li>
  <li class="nav-item" role="presentation">
    <a class="nav-link" id="messages-tab" data-toggle="tab" href="#messages" role="tab" aria-controls="messages" aria-selected="false">Messages</a>
  </li>
  <li class="nav-item" role="presentation">
    <a class="nav-link" id="settings-tab" data-toggle="tab" href="#settings" role="tab" aria-controls="settings" aria-selected="false">Settings</a>
  </li>
</ul>

<!-- Tab panes -->
<div class="tab-content">
  <div class="tab-pane active" id="home" role="tabpanel" aria-labelledby="home-tab">...</div>
  <div class="tab-pane" id="profile" role="tabpanel" aria-labelledby="profile-tab">...</div>
  <div class="tab-pane" id="messages" role="tabpanel" aria-labelledby="messages-tab">...</div>
  <div class="tab-pane" id="settings" role="tabpanel" aria-labelledby="settings-tab">...</div>
</div>

```

Via JavaScript

Enable tabbable tabs via JavaScript (each tab needs to be activated individually):

```

$( '#myTab a').on('click', function (e) {
  e.preventDefault()
  $(this).tab('show')
})

```

Copy

You can activate individual tabs in several ways:

```

$( '#myTab a[href="#profile"]').tab('show') // Select tab by name
$( '#myTab li:first-child a').tab('show') // Select first tab
$( '#myTab li:last-child a').tab('show') // Select last tab
$( '#myTab li:nth-child(3) a').tab('show') // Select third tab

```

Copy

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">...</div>
  <div class="tab-pane fade" id="profile" role="tabpanel" aria-labelledby="profile-tab">...</div>
  <div class="tab-pane fade" id="messages" role="tabpanel" aria-labelledby="messages-tab">...</div>
  <div class="tab-pane fade" id="settings" role="tabpanel" aria-labelledby="settings-tab">...</div>
</div>

```

Copy

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.](#)

`$(()).tab`

Activates a tab element and content container. Tab should have either a `data-target` or an `href` targeting a container node in the DOM.

```
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <a class="nav-link active" id="home-tab" data-toggle="tab" href="#home" role="tab" aria-controls="home" aria-selected="true">Home</a>
  </li>
  <li class="nav-item" role="presentation">
    <a class="nav-link" id="profile-tab" data-toggle="tab" href="#profile" role="tab" aria-controls="profile" aria-selected="false">Profile</a>
  </li>
  <li class="nav-item" role="presentation">
    <a class="nav-link" id="messages-tab" data-toggle="tab" href="#messages" role="tab" aria-controls="messages" aria-selected="false">Messages</a>
  </li>
  <li class="nav-item" role="presentation">
    <a class="nav-link" id="settings-tab" data-toggle="tab" href="#settings" role="tab" aria-controls="settings" aria-selected="false">Settings</a>
  </li>
</ul>

<div class="tab-content">
  <div class="tab-pane active" id="home" role="tabpanel" aria-labelledby="home-tab">...</div>
  <div class="tab-pane" id="profile" role="tabpanel" aria-labelledby="profile-tab">...</div>
  <div class="tab-pane" id="messages" role="tabpanel" aria-labelledby="messages-tab">...</div>
  <div class="tab-pane" id="settings" role="tabpanel" aria-labelledby="settings-tab">...</div>
</div>

<script>
$(function () {
  $('#myTab li:last-child a').tab('show')
})
</script>
```

Copy

`.tab('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).

```
$('#someTab').tab('show')
```

Copy

`.tab('dispose')`

Destroys an element's tab.

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
<code>show.bs.tab</code>	This event fires on tab show, but before the new tab has been shown. Use <code>event.target</code> and <code>event.relatedTarget</code> to target the active tab and the previous active tab (if available) respectively.
<code>shown.bs.tab</code>	This event fires on tab show after a tab has been shown. Use <code>event.target</code> and <code>event.relatedTarget</code> to target the active tab and the previous active tab (if available) respectively.
<code>hide.bs.tab</code>	This event fires when a new tab is to be shown (and thus the previous active tab is to be hidden). Use <code>event.target</code> and <code>event.relatedTarget</code> to target the current active tab and the new soon-to-be-active tab, respectively.
<code>hidden.bs.tab</code>	This event fires after a new tab is shown (and thus the previous active tab is hidden). Use <code>event.target</code> and <code>event.relatedTarget</code> to target the previous active tab and the new active tab, respectively.

```
$('a[data-toggle="tab"]').on('shown.bs.tab', function (e) {  
    e.target // newly activated tab  
    e.relatedTarget // previous active tab  
})
```

Copy

Pagination

Search...

Getting started

Layout

Content

Components

Alerts

Badge

Breadcrumb

Buttons

Button group

Card

Carousel

Collapse

Dropdowns

Forms

Input group

Jumbotron

List group

Media object

Modal

Navs

Navbar

Pagination

Popovers

Progress

Scrollspy

Spinners

Toasts

Tooltips

Utilities

Extend

Migration

About



Boost your app
monetization with
Facebook's people-based
advertising solutions
ads via Carbon

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](#)

```
<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>
```

Copy

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](#) [»](#)

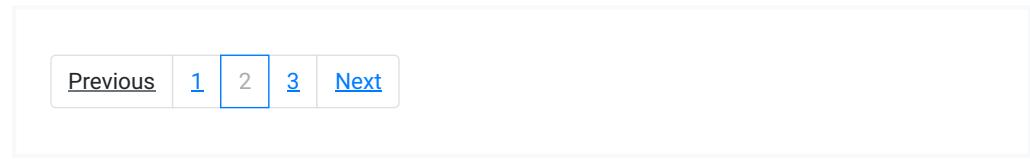
Copy

```
<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.



```
<nav aria-label="...">
  <ul class="pagination">
    <li class="page-item disabled">
      <a class="page-link" href="#" tabindex="-1" aria-
disabled="true">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 <span class="sr-only">(current)</span></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 ``, or omit the anchor in the case of the prev/next arrows, to remove click functionality and prevent keyboard focus while retaining intended styles.



Copy

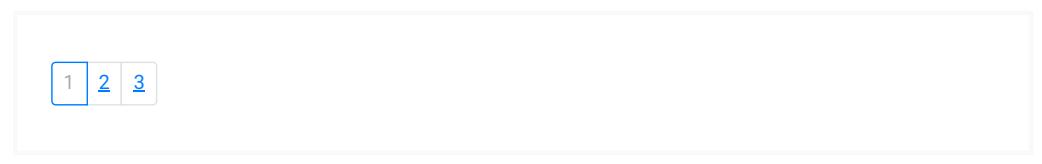
```
<nav aria-label="...">
  <ul class="pagination">
    <li class="page-item disabled">
      <span class="page-link">Previous</span>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item active" aria-current="page">
      <span class="page-link">
        2
        <span class="sr-only">(current)</span>
      </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.



```
<nav aria-label="...">
  <ul class="pagination pagination-lg">
    <li class="page-item active" aria-current="page">
      <span class="page-link">
        1
        <span class="sr-only">(current)</span>
      </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>
```



```
<nav aria-label="...">
  <ul class="pagination pagination-sm">
    <li class="page-item active" aria-current="page">
      <span class="page-link">
        1
        <span class="sr-only">(current)</span>
      </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](#).

[Previous](#) [1](#) [2](#) [3](#) [Next](#)

Copy

```
<nav aria-label="Page navigation example">
  <ul class="pagination justify-content-center">
    <li class="page-item disabled">
      <a class="page-link" href="#" tabindex="-1" aria-
disabled="true">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>
```

[Previous](#) [1](#) [2](#) [3](#) [Next](#)

Copy

```
<nav aria-label="Page navigation example">
  <ul class="pagination justify-content-end">
    <li class="page-item disabled">
      <a class="page-link" href="#" tabindex="-1" aria-
disabled="true">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>
```

Popovers

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Toolips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Overview

Things to know when using the popover plugin:

- Popovers rely on the 3rd party library [Popper.js](#) for positioning. You must include [popper.min.js](#) before bootstrap.js or use [bootstrap.bundle.min.js](#) / [bootstrap.bundle.js](#) which contains Popper.js in order for popovers to work!
- Popovers require the [tooltip plugin](#) as a dependency.
- If you're building our JavaScript from source, it [requires util.js](#).
- 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.

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.

Example: Enable popovers everywhere

One way to initialize all popovers on a page would be to select them by their `data-toggle="popover"` attribute:

```
$(function () {
  $('[data-toggle="popover"]').popover()
})
```

Copy

Example: Using the `container` option

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.

Copy

```
$(function () {
  $('.example-popover').popover({
    container: 'body'
  })
})
```

Example

Click to toggle popover

```
<button type="button" class="btn btn-lg btn-danger" data-toggle="popover" title="Popover title" data-content="And here's some amazing content. It's very engaging. Right?">Click to toggle popover</button>
```

Copy

Four directions

Four options are available: top, right, bottom, and left aligned.

Popover on top Popover on right Popover on bottom Popover on left

```
<button type="button" class="btn btn-secondary" data-container="body" data-toggle="popover" data-placement="top" data-content="Vivamus sagittis lacus vel augue laoreet rutrum faucibus.">
  Popover on top
</button>

<button type="button" class="btn btn-secondary" data-container="body" data-toggle="popover" data-placement="right" data-content="Vivamus sagittis lacus vel augue laoreet rutrum faucibus.">
  Popover on right
</button>

<button type="button" class="btn btn-secondary" data-container="body" data-toggle="popover" data-placement="bottom" data-content="Vivamus sagittis lacus vel augue laoreet rutrum faucibus.">
  Popover on bottom
</button>

<button type="button" class="btn btn-secondary" data-container="body" data-toggle="popover" data-placement="left" data-content="Vivamus sagittis lacus vel augue laoreet rutrum faucibus.">
  Popover on left
</button>
```

Copy

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.

Dismissible popover

```
<a tabindex="0" class="btn btn-lg btn-danger" role="button" data-
  toggle="popover" data-trigger="focus" title="Dismissible popover" data-
  content="And here's some amazing content. It's very engaging.
  Right?">Dismissible popover</a>
```

Copy

```
$('.popover-dismiss').popover({
  trigger: 'focus'
})
```

Copy

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 `` and override the `pointer-events` on the disabled element.

For disabled popover triggers, you may also prefer `data-trigger="hover"` so that the popover appears as immediate visual feedback to your users as they may not expect to *click* on a disabled element.

Disabled button

```
<span class="d-inline-block" data-toggle="popover" data-content="Disabled
  popover">
  <button class="btn btn-primary" style="pointer-events: none;" type="button"
    disabled>Disabled button</button>
</span>
```

Copy

Usage

Enable popovers via JavaScript:

```
$('#example').popover(options)
```

Copy

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 ``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 `whiteList` or 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-`, as in `data-animation=""`.

Note that for security reasons the `sanitize`, `sanitizeFn` and `whiteList` 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: <code>container: 'body'</code> . 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 <code>data-content</code> attribute isn't present. If a function is given, it will be called with its <code>this</code> reference set to the element that the popover is attached to.
delay	number object	0	Delay showing and hiding the popover (ms) - does not apply to manual trigger type If a number is supplied, delay is applied to both hide/show Object structure is: <code>delay: { "show": 500, "hide": 100 }</code>
html	boolean	false	Insert HTML into the popover. If false, jQuery's <code>text</code> method will be used to insert content into the DOM. Use <code>text</code> if you're worried about XSS attacks.

Name	Type	Default	Description
placement	string function	'right'	<p>How to position the popover - auto top bottom left right. When <code>auto</code> is specified, it will dynamically reorient the popover.</p> <p>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 <code>this</code> context is set to the popover instance.</p>
selector	string false	false	If a selector is provided, popover objects will be delegated to the specified targets. In practice, this is used to enable dynamic HTML content to have popovers added. See this and an informative example .
template	string	<pre>'<div class="popover" role="tooltip"> <div class="arrow"> </div><h3 class="popover- header"></h3> <div class="popover- body"></div> </div>'</pre>	<p>Base HTML to use when creating the popover.</p> <p>The popover's <code>title</code> will be injected into the <code>.popover-header</code>.</p> <p>The popover's <code>content</code> will be injected into the <code>.popover-body</code>.</p> <p><code>.arrow</code> will become the popover's arrow.</p> <p>The outermost wrapper element should have the <code>.popover</code> class.</p>
title	string element function	"	<p>Default title value if <code>title</code> attribute isn't present.</p> <p>If a function is given, it will be called with its <code>this</code> reference set to the element that the popover is attached to.</p>
trigger	string	'click'	How popover is triggered - click hover focus manual. You may pass multiple triggers; separate them with a space. <code>manual</code> cannot be combined with any other trigger.
offset	number string	0	Offset of the popover relative to its target. For more information refer to Popper.js's offset docs .
fallbackPlacement	string array	'flip'	Allow to specify which position Popper will use on fallback. For more information refer to Popper.js's behavior docs

Name	Type	Default	Description
boundary	string element	'scrollParent'	Overflow constraint boundary of the popover. Accepts the values of ' <code>'viewport'</code> ', ' <code>'window'</code> ', ' <code>'scrollParent'</code> ', or an HTMLElement reference (JavaScript only). For more information refer to Popper.js's preventOverflow docs .
sanitize	boolean	true	Enable or disable the sanitization. If activated ' <code>'template'</code> ', ' <code>'content'</code> and ' <code>'title'</code> ' options will be sanitized.
whiteList	object	Default value	Object which contains allowed 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	null	To change Bootstrap's default Popper.js config, see Popper.js's configuration

Data attributes for individual popovers

Options for individual popovers can alternatively be specified through the use of data attributes, as explained above.

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.](#)

`$(...).popover(options)`

Initializes popovers for an element collection.

`.popover('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.

```
$('#element').popover('show')
```

Copy

`.popover('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 "manual" triggering of the popover.

```
$('#element').popover('hide')
```

Copy

.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.

```
$( '#element' ).popover('toggle')
```

Copy

.popover('dispose')

Hides and destroys an element's popover. Popovers that use delegation (which are created using [the selector option](#)) cannot be individually destroyed on descendant trigger elements.

```
$( '#element' ).popover('dispose')
```

Copy

.popover('enable')

Gives an element's popover the ability to be shown. **Popovers are enabled by default.**

```
$( '#element' ).popover('enable')
```

Copy

.popover('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.

```
$( '#element' ).popover('disable')
```

Copy

.popover('toggleEnabled')

Toggles the ability for an element's popover to be shown or hidden.

```
$( '#element' ).popover('toggleEnabled')
```

Copy

.popover('update')

Updates the position of an element's popover.

```
$( '#element' ).popover('update')
```

Copy

Events

Event Type	Description
show.bs.popover	This event fires immediately when the <code>show</code> 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 <code>hide</code> 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 <code>show.bs.popover</code> event when the popover template has been added to the DOM.

```
$('#myPopover').on('hidden.bs.popover', function () {  
    // do something...  
})
```

Copy

Progress

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

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.

```
<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>
```

Copy

Bootstrap provides a handful of [utilities for setting width](#). Depending on your needs, these may help with quickly configuring progress.

[Copy](#)

```
<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`.

A progress bar with a light gray background and a blue progress bar inside. The text "25%" is centered within the blue bar.

25%

[Copy](#)

```
<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

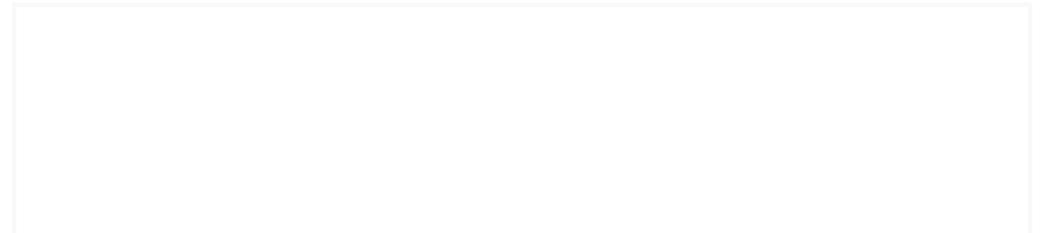
We only set a `height` value on the `.progress`, so if you change that value the inner `.progress-bar` will automatically resize accordingly.

[Copy](#)

```
<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.

[Copy](#)

```
<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.

```
<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.

```
Copy
```

```
<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.

[Toggle animation](#)

[Copy](#)

```
<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>
```

Scrollspy

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

How it works

Scrollspy has a few requirements to function properly:

- If you're building our JavaScript from source, it [requires util.js](#).
- It must be used on a Bootstrap [nav component](#) or [list group](#).
- Scrollspy requires `position: relative;` on the element you're spying on, usually the `<body>`.
- When spying on elements other than the `<body>`, be sure to have a `height` set and `overflow-y: scroll`; applied.
- Anchors (`<a>`) are required and must point to an element with that `id`.

When successfully implemented, your nav or list group will update accordingly, moving the `.active` class from one item to the next based on their associated targets.

Example in navbar

Scroll the area below the navbar and watch the active class change. The dropdown items will be highlighted as well.

@fat

Ad leggings keytar, brunch id art party dolor labore. Pitchfork yr enim lo-fi before they sold out qui. Tumblr farm-to-table bicycle rights whatever. Anim keffiyeh carles cardigan. Velit seitan mcsweeney's photo booth 3 wolf moon irure. Cosby sweater lomo jean shorts, williamsburg hoodie minim qui you probably haven't heard of them et cardigan trust fund culpa biodiesel wes anderson aesthetic. Nihil tattooed accusamus, cred irony biodiesel keffiyeh artisan ullamco consequat.

Copy

```

<nav id="navbar-example2" class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <ul class="nav nav-pills">
    <li class="nav-item">
      <a class="nav-link" href="#fat">@fat</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#mdo">@mdo</a>
    </li>
    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false">Dropdown</a>
      <div class="dropdown-menu">
        <a class="dropdown-item" href="#one">one</a>
        <a class="dropdown-item" href="#two">two</a>
        <div role="separator" class="dropdown-divider"></div>
        <a class="dropdown-item" href="#three">three</a>
      </div>
    </li>
  </ul>
</nav>
<div data-spy="scroll" data-target="#navbar-example2" data-offset="0">
  <h4 id="fat">@fat</h4>
  <p>...</p>
  <h4 id="mdo">@mdo</h4>
  <p>...</p>
  <h4 id="one">one</h4>
  <p>...</p>
  <h4 id="two">two</h4>
  <p>...</p>
  <h4 id="three">three</h4>
  <p>...</p>
</div>

```

Example with nested nav

Scrollspy also works with nested `.nav`s. If a nested `.nav` is `.active`, its parents will also be `.active`. Scroll the area next to the navbar and watch the active class change.

Item 1

Ex consequat commodo adipisicing exercitation aute
excepteur occaecat ullamco duis aliqua id magna ullamco
eu. Do aute ipsum ipsum ullamco cillum consectetur ut et
aute consectetur labore. Fugiat laborum incididunt tempor eu
consequat enim dolore proident. Qui laborum do non
excepteur nulla magna eiusmod consectetur in. Aliqua et
aliqua officia quis et incididunt voluptate non anim
reprehenderit adipisicing dolore ut consequat deserunt mollit
dolore. Aliquip nulla enim veniam non fugiat id cupidatat
nulla elit cupidatat commodo velit ut eiusmod cupidatat elit
dolore.

Item 1-1

Copy

```

<nav id="navbar-example3" class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <nav class="nav nav-pills flex-column">
    <a class="nav-link" href="#item-1">Item 1</a>
    <nav class="nav nav-pills flex-column">
      <a class="nav-link ml-3 my-1" href="#item-1-1">Item 1-1</a>
      <a class="nav-link ml-3 my-1" href="#item-1-2">Item 1-2</a>
    </nav>
    <a class="nav-link" href="#item-2">Item 2</a>
    <a class="nav-link" href="#item-3">Item 3</a>
    <nav class="nav nav-pills flex-column">
      <a class="nav-link ml-3 my-1" href="#item-3-1">Item 3-1</a>
      <a class="nav-link ml-3 my-1" href="#item-3-2">Item 3-2</a>
    </nav>
  </nav>
</nav>

<div data-spy="scroll" data-target="#navbar-example3" data-offset="0">
  <h4 id="item-1">Item 1</h4>
  <p>...</p>
  <h5 id="item-1-1">Item 1-1</h5>
  <p>...</p>
  <h5 id="item-1-2">Item 1-2</h5>
  <p>...</p>
  <h4 id="item-2">Item 2</h4>
  <p>...</p>
  <h4 id="item-3">Item 3</h4>
  <p>...</p>
  <h5 id="item-3-1">Item 3-1</h5>
  <p>...</p>
  <h5 id="item-3-2">Item 3-2</h5>
  <p>...</p>
</div>

```

Example with list-group

Scrollspy also works with `.list-groups`. Scroll the area next to the list group and watch the active class change.

Item 1
Item 2
Item 3
Item 4

Item 1

Ex consequat commodo adipisicing exercitation aute
excepteur occaecat ullamco duis aliqua id magna ullamco
eu. Do aute ipsum ipsum ullamco cillum consectetur ut et
aute consectetur labore. Fugiat laborum incididunt tempor eu
consequat enim dolore proident. Qui laborum do non
excepteur nulla magna eiusmod consectetur in. Aliqua et
aliqua officia quis et incididunt voluptate non anim

Copy

```

<div id="list-example" class="list-group">
  <a class="list-group-item list-group-item-action" href="#list-item-1">Item
  1</a>
  <a class="list-group-item list-group-item-action" href="#list-item-2">Item
  2</a>
  <a class="list-group-item list-group-item-action" href="#list-item-3">Item
  3</a>
  <a class="list-group-item list-group-item-action" href="#list-item-4">Item
  4</a>
</div>
<div data-spy="scroll" data-target="#list-example" data-offset="0"
class="scrollspy-example">
  <h4 id="list-item-1">Item 1</h4>
  <p>...</p>
  <h4 id="list-item-2">Item 2</h4>
  <p>...</p>
  <h4 id="list-item-3">Item 3</h4>
  <p>...</p>
  <h4 id="list-item-4">Item 4</h4>
  <p>...</p>
</div>

```

Usage

Via data attributes

To easily add scrollspy behavior to your topbar navigation, add `data-spy="scroll"` to the element you want to spy on (most typically this would be the `<body>`). Then add the `data-target` attribute with the ID or class of the parent element of any Bootstrap `.nav` component.

```

body {
  position: relative;
}

```

[Copy](#)

```

<body data-spy="scroll" data-target="#navbar-example">
  ...
  <div id="navbar-example">
    <ul class="nav nav-tabs" role="tablist">
      ...
    </ul>
  </div>
  ...
</body>

```

[Copy](#)

Via JavaScript

After adding `position: relative;` in your CSS, call the scrollspy via JavaScript:

```

$( 'body' ).scrollspy({ target: '#navbar-example' })

```

[Copy](#)

Resolvable ID targets required

Navbar links must have resolvable id targets. For example, a `home` must correspond to something in the DOM like `<div id="home"></div>`.

Non-`:visible` target elements ignored

Target elements that are not [:visible according to jQuery](#) will be ignored and their corresponding nav items will never be highlighted.

Methods

.scrollspy('refresh')

When using scrollspy in conjunction with adding or removing of elements from the DOM, you'll need to call the refresh method like so:

```
$('[data-spy="scroll"]').each(function () {  
  var $spy = $(this).scrollspy('refresh')  
})
```

Copy

.scrollspy('dispose')

Destroys an element's scrollspy.

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-offset=""`.

Name	Type	Default	Description
offset	number	10	Pixels to offset from top when calculating position of scroll.
method	string	auto	Finds which section the spied element is in. <code>auto</code> will choose the best method to get scroll coordinates. <code>offset</code> will use jQuery offset method to get scroll coordinates. <code>position</code> will use jQuery position method to get scroll coordinates.
target	string jQuery object DOM element		Specifies element to apply Scrollspy plugin.

Events

Event Type	Description
activate.bs.scrollspy	This event fires on the scroll element whenever a new item becomes activated by the scrollspy.

```
$('[data-spy="scroll"]').on('activate.bs.scrollspy', function () {  
  // do something...  
})
```

Copy

Spinners

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Toolips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

About

Bootstrap “spinners” can be used to show the loading state in your projects. 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, alignment, and sizing can be easily customized with our amazing utility classes.

For accessibility purposes, each loader here includes `role="status"` and a nested `Loading...`.

Border spinner

Use the border spinners for a lightweight loading indicator.



Copy

```
<div class="spinner-border" role="status">
  <span class="sr-only">Loading...</span>
</div>
```

Colors

The border spinner uses `currentColor` for its `border-color`, meaning you can customize the color with [text color utilities](#). You can use any of our text color utilities on the standard spinner.



Copy

```
<div class="spinner-border text-primary" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-border text-secondary" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-border text-success" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-border text-danger" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-border text-warning" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-border text-info" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-border text-light" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-border text-dark" role="status">
  <span class="sr-only">Loading...</span>
</div>
```

Why not use border-color utilities? Each border spinner specifies a `transparent` border for at least one side, so `.border-{color}` utilities would override that.

Growing spinner

If you don't fancy a border spinner, switch to the grow spinner. While it doesn't technically spin, it does repeatedly grow!

```
<div class="spinner-grow" role="status">
  <span class="sr-only">Loading...</span>
</div>
```

Copy

Once again, this spinner is built with `currentColor`, so you can easily change its appearance with [text color utilities](#). Here it is in blue, along with the supported variants.

```
<div class="spinner-grow" style="background-color: blue;" role="status">
  <span class="sr-only">Loading...</span>
</div>
```

Copy

```
<div class="spinner-grow text-primary" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-grow text-secondary" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-grow text-success" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-grow text-danger" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-grow text-warning" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-grow text-info" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-grow text-light" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-grow text-dark" role="status">
  <span class="sr-only">Loading...</span>
</div>
```

Alignment

Spinners in Bootstrap are built with `rem`s, `currentColor`, and `display: inline-flex`. This means they can easily be resized, recolored, and quickly aligned.

Margin

Use [margin utilities](#) like `.m-5` for easy spacing.



```
<div class="spinner-border m-5" role="status">
  <span class="sr-only">Loading...</span>
</div>
```

Copy

Placement

Use [flexbox utilities](#), [float utilities](#), or [text alignment](#) utilities to place spinners exactly where you need them in any situation.

Flex



```
<div class="d-flex justify-content-center">
  <div class="spinner-border" role="status">
    <span class="sr-only">Loading...</span>
  </div>
</div>
```

Copy

Loading...



```
<div class="d-flex align-items-center">
  <strong>Loading...</strong>
  <div class="spinner-border ml-auto" role="status" aria-hidden="true"></div>
</div>
```

Copy

Floats



```
<div class="clearfix">
  <div class="spinner-border float-right" role="status">
    <span class="sr-only">Loading...</span>
  </div>
</div>
```

Copy

Text align



```
<div class="text-center">
  <div class="spinner-border" role="status">
    <span class="sr-only">Loading...</span>
  </div>
</div>
```

Copy

Size

Add `.spinner-border-sm` and `.spinner-grow-sm` to make a smaller spinner that can quickly be used within other components.



Copy

```
<div class="spinner-border spinner-border-sm" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-grow spinner-grow-sm" role="status">
  <span class="sr-only">Loading...</span>
</div>
```

Or, use custom CSS or inline styles to change the dimensions as needed.

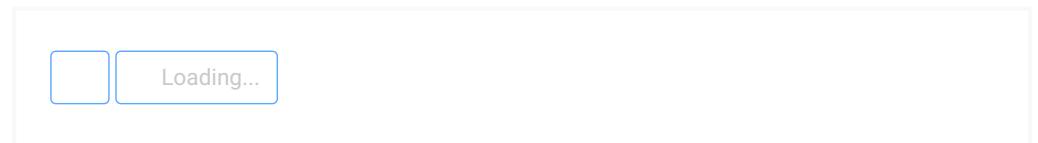


Copy

```
<div class="spinner-border" style="width: 3rem; height: 3rem;" role="status">
  <span class="sr-only">Loading...</span>
</div>
<div class="spinner-grow" style="width: 3rem; height: 3rem;" role="status">
  <span class="sr-only">Loading...</span>
</div>
```

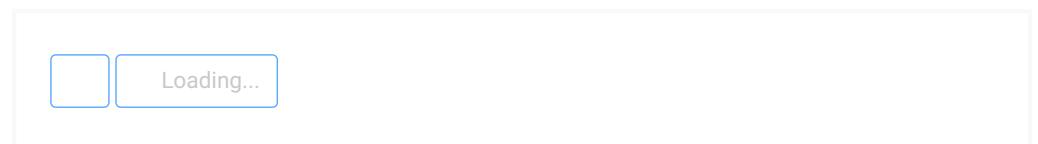
Buttons

Use spinners within buttons to indicate an action is currently processing or taking place. You may also swap the text out of the spinner element and utilize button text as needed.



Copy

```
<button class="btn btn-primary" type="button" disabled>
  <span class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span>
  <span class="sr-only">Loading...</span>
</button>
<button class="btn btn-primary" type="button" disabled>
  <span class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span>
  Loading...
</button>
```



Copy

```
<button class="btn btn-primary" type="button" disabled>
  <span class="spinner-grow spinner-grow-sm" role="status" aria-hidden="true"></span>
  <span class="sr-only">Loading...</span>
</button>
<button class="btn btn-primary" type="button" disabled>
  <span class="spinner-grow spinner-grow-sm" role="status" aria-hidden="true"></span>
  Loading...
</button>
```

Toasts

Search...

Getting started

Layout

Content

Components

Alerts

Badge

Breadcrumb

Buttons

Button group

Card

Carousel

Collapse

Dropdowns

Forms

Input group

Jumbotron

List group

Media object

Modal

Navs

Navbar

Pagination

Popovers

Progress

Scrollspy

Spinners

Toasts

Tooltips

Utilities

Extend

Migration

About



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Toasts are lightweight notifications designed to mimic the push notifications that have been popularized by mobile and desktop operating systems. They're built with flexbox, so they're easy to align and position.

Overview

Things to know when using the toast plugin:

- If you're building our JavaScript from source, it [requires util.js](#).
- Toasts are opt-in for performance reasons, so **you must initialize them yourself**.
- **Please note that you are responsible for positioning toasts.**
- Toasts will automatically hide if you do not specify `autohide: false`.

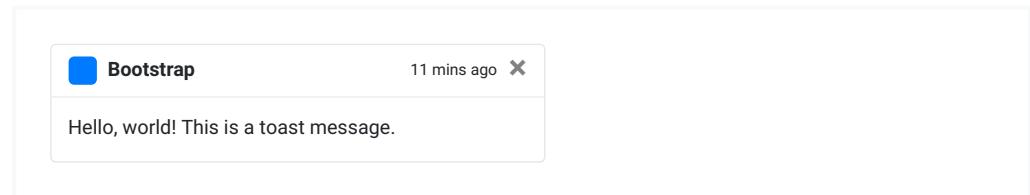
The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the [reduced motion section of our accessibility documentation](#).

Examples

Basic

To encourage extensible and predictable toasts, we recommend a header and body. Toast headers use `display: flex`, allowing easy alignment of content thanks to our margin and flexbox utilities.

Toasts are as flexible as you need and have very little required markup. At a minimum, we require a single element to contain your "toasted" content and strongly encourage a dismiss button.

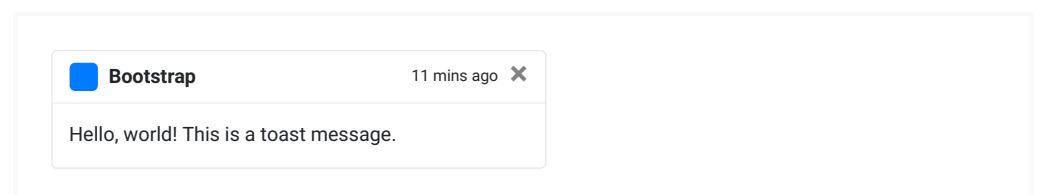


Copy

```
<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    
    <strong class="mr-auto">Bootstrap</strong>
    <small>11 mins ago</small>
    <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="toast-body">
    Hello, world! This is a toast message.
  </div>
</div>
```

Translucent

Toasts are slightly translucent, too, so they blend over whatever they might appear over. For browsers that support the `backdrop-filter` CSS property, we'll also attempt to blur the elements under a toast.

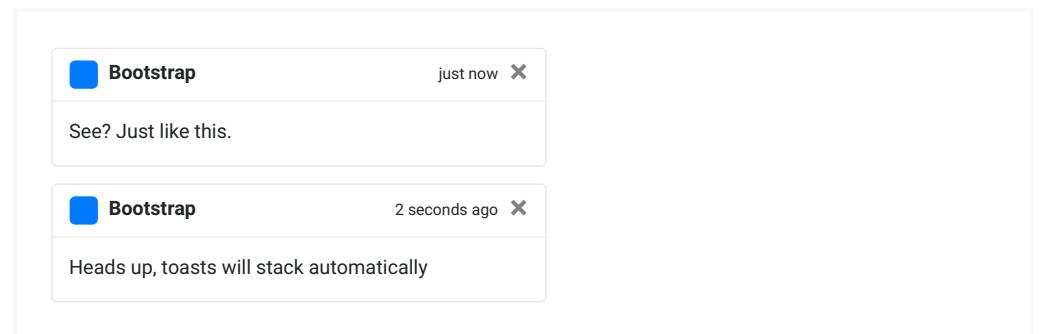


```
<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    
    <strong class="mr-auto">Bootstrap</strong>
    <small class="text-muted">11 mins ago</small>
    <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="toast-body">
    Hello, world! This is a toast message.
  </div>
</div>
```

Copy

Stacking

When you have multiple toasts, we default to vertically stacking them in a readable manner.



Copy

```

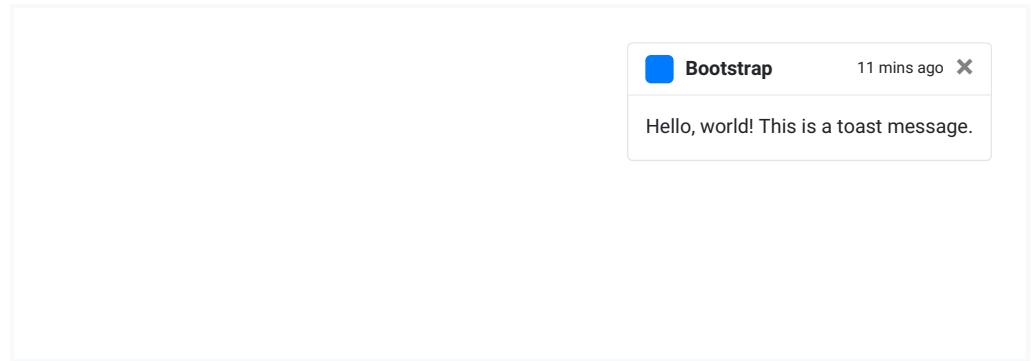
<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    
    <strong class="mr-auto">Bootstrap</strong>
    <small class="text-muted">just now</small>
    <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="toast-body">
    See? Just like this.
  </div>
</div>

<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    
    <strong class="mr-auto">Bootstrap</strong>
    <small class="text-muted">2 seconds ago</small>
    <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="toast-body">
    Heads up, toasts will stack automatically
  </div>
</div>

```

Placement

Place toasts with custom CSS as you need them. The top right is often used for notifications, as is the top middle. If you're only ever going to show one toast at a time, put the positioning styles right on the `.toast`.

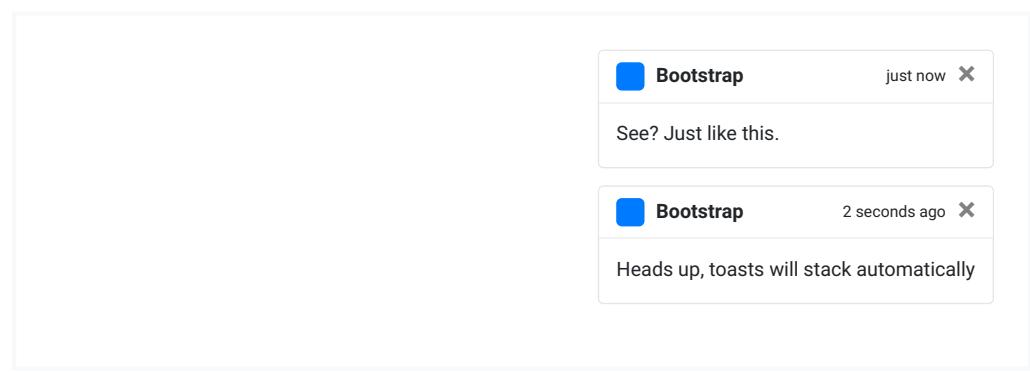


```

<div aria-live="polite" aria-atomic="true" style="position: relative; min-height: 200px;">
  <div class="toast" style="position: absolute; top: 0; right: 0;">
    <div class="toast-header">
      
      <strong class="mr-auto">Bootstrap</strong>
      <small>11 mins ago</small>
      <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
  </div>
</div>

```

For systems that generate more notifications, consider using a wrapping element so they can easily stack.



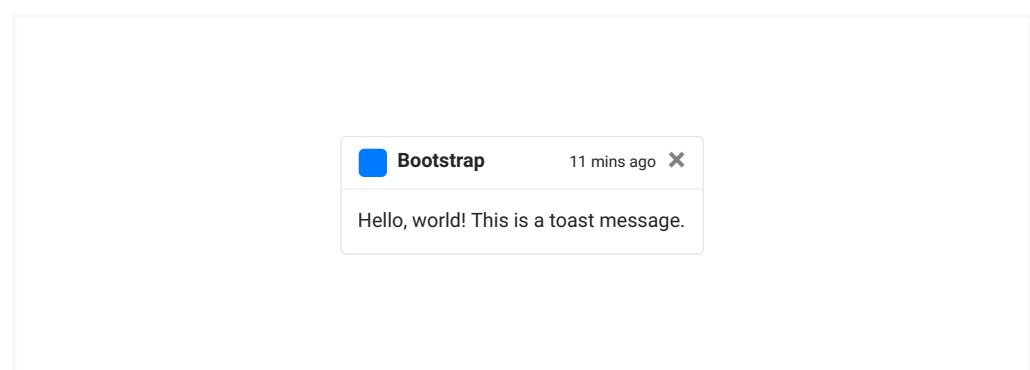
```
<div aria-live="polite" aria-atomic="true" style="position: relative; min-height: 200px;">
  <!-- Position it -->
  <div style="position: absolute; top: 0; right: 0;">

    <!-- Then put toasts within -->
    <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
      <div class="toast-header">
        
        <strong class="mr-auto">Bootstrap</strong>
        <small class="text-muted">just now</small>
        <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="toast-body">
        See? Just like this.
      </div>
    </div>

    <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
      <div class="toast-header">
        
        <strong class="mr-auto">Bootstrap</strong>
        <small class="text-muted">2 seconds ago</small>
        <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="toast-body">
        Heads up, toasts will stack automatically
      </div>
    </div>
  </div>
</div>
```

Copy

You can also get fancy with flexbox utilities to align toasts horizontally and/or vertically.



Copy

```

<!-- Flexbox container for aligning the toasts -->
<div aria-live="polite" aria-atomic="true" class="d-flex justify-content-center align-items-center" style="min-height: 200px;">

    <!-- Then put toasts within -->
    <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
        <div class="toast-header">
            
            <strong class="mr-auto">Bootstrap</strong>
            <small>11 mins ago</small>
            <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
        </div>
        <div class="toast-body">
            Hello, world! This is a toast message.
        </div>
    </div>
</div>

```

Accessibility

Toasts are intended to be small interruptions to your visitors or users, so to help those with screen readers and similar assistive technologies, you should wrap your toasts in an [aria-live region](#). Changes to live regions (such as injecting/updating a toast component) are automatically announced by screen readers without needing to move the user's focus or otherwise interrupt the user. Additionally, include `aria-atomic="true"` to ensure that the entire toast is always announced as a single (atomic) unit, rather than announcing what was changed (which could lead to problems if you only update part of the toast's content, or if displaying the same toast content at a later point in time). If the information needed is important for the process, e.g. for a list of errors in a form, then use the [alert component](#) instead of toast.

Note that the live region needs to be present in the markup *before* the toast is generated or updated. If you dynamically generate both at the same time and inject them into the page, they will generally not be announced by assistive technologies.

You also need to adapt the `role` and `aria-live` level depending on the content. If it's an important message like an error, use `role="alert" aria-live="assertive"`, otherwise use `role="status" aria-live="polite"` attributes.

As the content you're displaying changes, be sure to update the [delay timeout](#) to ensure people have enough time to read the toast.

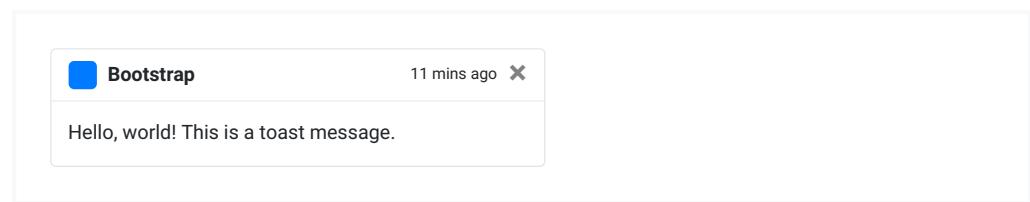
```

<div class="toast" role="alert" aria-live="polite" aria-atomic="true" data-delay="10000">
    <div role="alert" aria-live="assertive" aria-atomic="true">...</div>
</div>

```

Copy

When using `autohide: false`, you must add a close button to allow users to dismiss the toast.


Copy

```

<div role="alert" aria-live="assertive" aria-atomic="true" class="toast" data-autohide="false">
  <div class="toast-header">
    
    <strong class="mr-auto">Bootstrap</strong>
    <small>11 mins ago</small>
    <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="toast-body">
    Hello, world! This is a toast message.
  </div>
</div>

```

JavaScript behavior

Usage

Initialize toasts via JavaScript:

```
$('.toast').toast(option)
```

Copy

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-animation=""`.

Name	Type	Default	Description
animation	boolean	true	Apply a CSS fade transition to the toast
autohide	boolean	true	Auto hide the toast
delay	number	500	Delay hiding the toast (ms)

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.](#)

`$().toast(options)`

Attaches a toast handler to an element collection.

`.toast('show')`

Reveals an element's toast. **Returns to the caller before the toast has actually been shown** (i.e. before the `shown.bs.toast` event occurs). You have to manually call this method, instead your toast won't show.

```
$('#element').toast('show')
```

Copy

.toast('hide')

Hides an element's toast. **Returns to the caller before the toast has actually been hidden** (i.e. before the `hidden.bs.toast` event occurs). You have to manually call this method if you made `autohide` to `false`.

```
$( '#element' ).toast('hide')
```

Copy

.toast('dispose')

Hides an element's toast. Your toast will remain on the DOM but won't show anymore.

```
$( '#element' ).toast('dispose')
```

Copy

Events

Event Type	Description
show.bs.toast	This event fires immediately when the <code>show</code> instance method is called.
shown.bs.toast	This event is fired when the toast has been made visible to the user.
hide.bs.toast	This event is fired immediately when the <code>hide</code> instance method has been called.
hidden.bs.toast	This event is fired when the toast has finished being hidden from the user.

```
$( '#myToast' ).on('hidden.bs.toast', function () {  
    // do something...  
})
```

Copy

Tooltips

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Alerts](#)

[Badge](#)

[Breadcrumb](#)

[Buttons](#)

[Button group](#)

[Card](#)

[Carousel](#)

[Collapse](#)

[Dropdowns](#)

[Forms](#)

[Input group](#)

[Jumbotron](#)

[List group](#)

[Media object](#)

[Modal](#)

[Navs](#)

[Navbar](#)

[Pagination](#)

[Popovers](#)

[Progress](#)

[Scrollspy](#)

[Spinners](#)

[Toasts](#)

[Tooltips](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Grow your app revenue
with quality demand from
Facebook's global
advertisers
ads via Carbon

Overview

Things to know when using the tooltip plugin:

- Tooltips rely on the 3rd party library [Popper.js](#) for positioning. You must include [popper.min.js](#) before bootstrap.js or use [bootstrap.bundle.min.js](#) / [bootstrap.bundle.js](#) which contains Popper.js in order for tooltips to work!
- If you're building our JavaScript from source, it [requires util.js](#).
- Tooltips are opt-in for performance reasons, so **you must initialize them yourself**.
- Tooltips with zero-length titles are never displayed.
- Specify `container: 'body'` to avoid rendering problems in more complex components (like our input groups, button groups, etc).
- Triggering tooltips on hidden elements will not work.
- Tooltips for `.disabled` or `disabled` elements must be triggered on a wrapper element.
- When triggered from hyperlinks that span multiple lines, tooltips will be centered. Use `white-space: nowrap;` on your `<a>`s to avoid this behavior.
- Tooltips must be hidden before their corresponding elements have been removed from the DOM.
- Tooltips can be triggered thanks to an element inside a shadow DOM.

The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the [reduced motion section of our accessibility documentation](#).

Got all that? Great, let's see how they work with some examples.

Example: Enable tooltips everywhere

One way to initialize all tooltips on a page would be to select them by their `data-toggle="tooltip"` attribute:

```
$(function () {  
  $('[data-toggle="tooltip"]').tooltip()  
})
```

Copy

Examples

Hover over the links below to see tooltips:

Tight pants next level keffiyeh [you probably](#) haven't heard of them. Photo booth beard raw denim letterpress vegan messenger bag stumptown. Farm-to-table seitan, mcsweeney's fixie sustainable quinoa 8-bit american apparel [have a](#) terry richardson vinyl chambray. Beard stumptown, cardigans banh mi lomo thundercats. Tofu biodiesel williamsburg marfa, four loko mcsweeney's cleanse vegan chambray. A really ironic artisan [whatever keytar](#), scenester farm-to-table banksy Austin [twitter handle](#) freegan cred raw denim single-origin coffee viral.

Hover over the buttons below to see the four tooltips directions: top, right, bottom, and left.

Tooltip on top Tooltip on right Tooltip on bottom Tooltip on left
Tooltip with HTML

```
<button type="button" class="btn btn-secondary" data-toggle="tooltip" data-placement="top" title="Tooltip on top">  
    Tooltip on top  
</button>  
<button type="button" class="btn btn-secondary" data-toggle="tooltip" data-placement="right" title="Tooltip on right">  
    Tooltip on right  
</button>  
<button type="button" class="btn btn-secondary" data-toggle="tooltip" data-placement="bottom" title="Tooltip on bottom">  
    Tooltip on bottom  
</button>  
<button type="button" class="btn btn-secondary" data-toggle="tooltip" data-placement="left" title="Tooltip on left">  
    Tooltip on left  
</button>
```

Copy

And with custom HTML added:

```
<button type="button" class="btn btn-secondary" data-toggle="tooltip" data-html="true" title="Tooltip <u>with</u> <b>HTML</b>">  
    Tooltip with HTML  
</button>
```

Copy

Usage

The tooltip plugin generates content and markup on demand, and by default places tooltips after their trigger element.

Trigger the tooltip via JavaScript:

```
$( '#example' ).tooltip( options )
```

Copy

Overflow `auto` and `scroll`

Tooltip position attempts to automatically change when a parent container has `overflow: auto` or `overflow: scroll` like our `.table-responsive`, but still keeps the original placement's positioning. To resolve, set the `boundary` option to anything other than default value, '`scrollParent`', such as '`window`'.

```
$( '#example' ).tooltip({ boundary: 'window' })
```

Copy

Markup

The required markup for a tooltip is only a `data` attribute and `title` on the HTML element you wish to have a tooltip. The generated markup of a tooltip is rather simple, though it does require a position (by default, set to `top` by the plugin).

Making tooltips work for keyboard and assistive technology users

You should only add tooltips to HTML elements that are traditionally keyboard-focusable and interactive (such as links or form controls). Although arbitrary HTML elements (such as ``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 tooltip in this situation. Additionally, do not rely solely on `hover` as the trigger for your tooltip, as this will make your tooltips impossible to trigger for keyboard users.

```
<!-- HTML to write -->
Hover over me

<!-- Generated markup by the plugin -->


<div class="arrow"></div>
  <div class="tooltip-inner">
    Some tooltip text!
  </div>


```

Copy

Disabled elements

Elements with the `disabled` attribute aren't interactive, meaning users cannot focus, hover, or click them to trigger a tooltip (or popover). As a workaround, you'll want to trigger the tooltip from a wrapper `<div>` or ``, ideally made keyboard-focusable using `tabindex="0"`, and override the `pointer-events` on the disabled element.

Disabled button

```
<span class="d-inline-block" tabindex="0" data-toggle="tooltip" title="Disabled tooltip">
  <button class="btn btn-primary" style="pointer-events: none;" type="button" disabled>Disabled button</button>
</span>
```

Copy

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-animation=""`.

Note that for security reasons the `sanitize`, `sanitizeFn` and `whiteList` options cannot be supplied using data attributes.

Name	Type	Default	Description
animation	boolean	true	Apply a CSS fade transition to the tooltip

Name	Type	Default	Description
container	string element false	false	Appends the tooltip to a specific element. Example: <code>container: 'body'</code> . This option is particularly useful in that it allows you to position the tooltip in the flow of the document near the triggering element - which will prevent the tooltip from floating away from the triggering element during a window resize.
delay	number object	0	<p>Delay showing and hiding the tooltip (ms) - does not apply to manual trigger type</p> <p>If a number is supplied, delay is applied to both hide/show</p> <p>Object structure is: <code>delay: { "show": 500, "hide": 100 }</code></p>
html	boolean	false	<p>Allow HTML in the tooltip.</p> <p>If true, HTML tags in the tooltip's <code>title</code> will be rendered in the tooltip. If false, jQuery's <code>text</code> method will be used to insert content into the DOM.</p> <p>Use text if you're worried about XSS attacks.</p>
placement	string function	'top'	<p>How to position the tooltip - auto top bottom left right.</p> <p>When <code>auto</code> is specified, it will dynamically reorient the tooltip.</p> <p>When a function is used to determine the placement, it is called with the tooltip DOM node as its first argument and the triggering element DOM node as its second. The <code>this</code> context is set to the tooltip instance.</p>
selector	string false	false	If a selector is provided, tooltip objects will be delegated to the specified targets. In practice, this is used to also apply tooltips to dynamically added DOM elements (<code>jQuery.on</code> support). See this and an informative example .
template	string	<pre>'<div class="tooltip" role="tooltip"> <div class="arrow"> </div><div class="tooltip- inner"></div> </div>'</pre>	<p>Base HTML to use when creating the tooltip.</p> <p>The tooltip's <code>title</code> will be injected into the <code>.tooltip-inner</code>.</p> <p><code>.arrow</code> will become the tooltip's arrow.</p> <p>The outermost wrapper element should have the <code>.tooltip</code> class and <code>role="tooltip"</code>.</p>

Name	Type	Default	Description
title	string element function	"	<p>Default title value if <code>title</code> attribute isn't present.</p> <p>If a function is given, it will be called with its <code>this</code> reference set to the element that the tooltip is attached to.</p>
trigger	string	'hover focus'	<p>How tooltip is triggered - click hover focus manual. You may pass multiple triggers; separate them with a space.</p> <p><code>'manual'</code> indicates that the tooltip will be triggered programmatically via the <code>.tooltip('show')</code>, <code>.tooltip('hide')</code> and <code>.tooltip('toggle')</code> methods; this value cannot be combined with any other trigger.</p> <p><code>'hover'</code> on its own will result in tooltips 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.</p>
offset	number string function	0	<p>Offset of the tooltip relative to its target.</p> <p>When a function is used to determine the offset, it is called with an object containing the offset data as its first argument. The function must return an object with the same structure. The triggering element DOM node is passed as the second argument.</p> <p>For more information refer to Popper.js's offset docs.</p>
fallbackPlacement	string array	'flip'	Allow to specify which position Popper will use on fallback. For more information refer to Popper.js's behavior docs
boundary	string element	'scrollParent'	Overflow constraint boundary of the tooltip. Accepts the values of <code>'viewport'</code> , <code>'window'</code> , <code>'scrollParent'</code> , or an HTMLElement reference (JavaScript only). For more information refer to Popper.js's preventOverflow docs .
sanitize	boolean	true	Enable or disable the sanitization. If activated <code>'template'</code> and <code>'title'</code> options will be sanitized.
whiteList	object	Default value	Object which contains allowed attributes and tags

Name	Type	Default	Description
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	null	To change Bootstrap's default Popper.js config, see Popper.js's configuration

Data attributes for individual tooltips

Options for individual tooltips can alternatively be specified through the use of data attributes, as explained above.

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.](#)

`$().tooltip(options)`

Attaches a tooltip handler to an element collection.

`.tooltip('show')`

Reveals an element's tooltip. **Returns to the caller before the tooltip has actually been shown** (i.e. before the `shown.bs.tooltip` event occurs). This is considered a “manual” triggering of the tooltip. Tooltips with zero-length titles are never displayed.

`$('#element').tooltip('show')`

Copy

`.tooltip('hide')`

Hides an element's tooltip. **Returns to the caller before the tooltip has actually been hidden** (i.e. before the `hidden.bs.tooltip` event occurs). This is considered a “manual” triggering of the tooltip.

`$('#element').tooltip('hide')`

Copy

`.tooltip('toggle')`

Toggles an element's tooltip. **Returns to the caller before the tooltip has actually been shown or hidden** (i.e. before the `shown.bs.tooltip` or `hidden.bs.tooltip` event occurs). This is considered a “manual” triggering of the tooltip.

`$('#element').tooltip('toggle')`

Copy

`.tooltip('dispose')`

Hides and destroys an element's tooltip. Tooltips that use delegation (which are created using [the selector option](#)) cannot be individually destroyed on descendant trigger elements.

`$('#element').tooltip('dispose')`

Copy

.tooltip('enable')

Gives an element's tooltip the ability to be shown. **Tooltips are enabled by default.**

```
$( '#element' ).tooltip('enable')
```

Copy

.tooltip('disable')

Removes the ability for an element's tooltip to be shown. The tooltip will only be able to be shown if it is re-enabled.

```
$( '#element' ).tooltip('disable')
```

Copy

.tooltip('toggleEnabled')

Toggles the ability for an element's tooltip to be shown or hidden.

```
$( '#element' ).tooltip('toggleEnabled')
```

Copy

.tooltip('update')

Updates the position of an element's tooltip.

```
$( '#element' ).tooltip('update')
```

Copy

Events

Event Type	Description
show.bs.tooltip	This event fires immediately when the <code>show</code> instance method is called.
shown.bs.tooltip	This event is fired when the tooltip has been made visible to the user (will wait for CSS transitions to complete).
hide.bs.tooltip	This event is fired immediately when the <code>hide</code> instance method has been called.
hidden.bs.tooltip	This event is fired when the tooltip has finished being hidden from the user (will wait for CSS transitions to complete).
inserted.bs.tooltip	This event is fired after the <code>show.bs.tooltip</code> event when the tooltip template has been added to the DOM.

```
$( '#myTooltip' ).on('hidden.bs.tooltip', function () {  
  // do something...  
})
```

Copy

Borders

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Borders](#)

[Clearfix](#)

[Close icon](#)

[Colors](#)

[Display](#)

[Embed](#)

[Flex](#)

[Float](#)

[Image replacement](#)

[Interactions](#)

[Overflow](#)

[Position](#)

[Screen readers](#)

[Shadows](#)

[Sizing](#)

[Spacing](#)

[Stretched link](#)

[Text](#)

[Vertical align](#)

[Visibility](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Border

Use border utilities to add or remove an element's borders. Choose from all borders or one at a time.

Additive



```
<span class="border"></span>
<span class="border-top"></span>
<span class="border-right"></span>
<span class="border-bottom"></span>
<span class="border-left"></span>
```

Copy

Subtractive

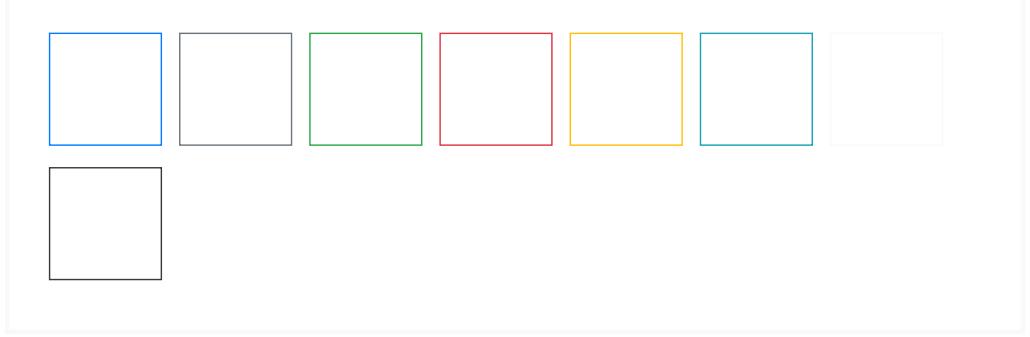


```
<span class="border-0"></span>
<span class="border-top-0"></span>
<span class="border-right-0"></span>
<span class="border-bottom-0"></span>
<span class="border-left-0"></span>
```

Copy

Border color

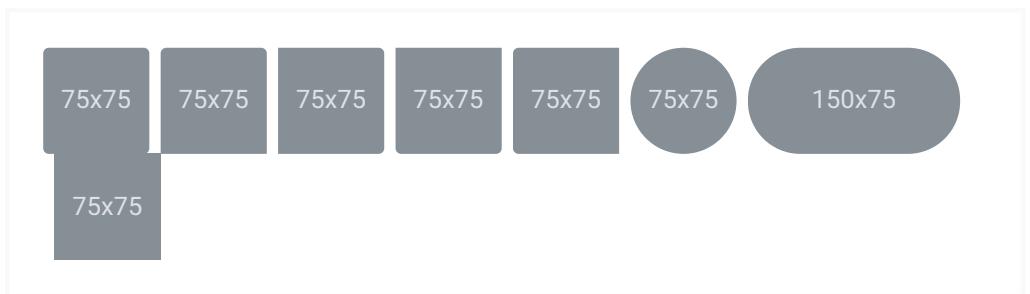
Change the border color using utilities built on our theme colors.

[Copy](#)

```
<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>
```

Border-radius

Add classes to an element to easily round its corners.

[Copy](#)

```



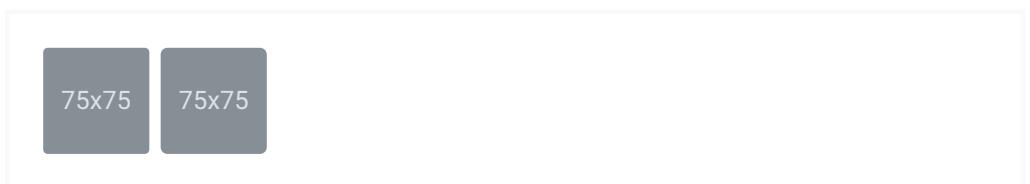





```

Sizes

Use `.rounded-sm` or `.rounded-lg` for larger or smaller border-radius.

[Copy](#)

```


```

Clearfix

[Getting started](#)[Layout](#)[Content](#)[Components](#)[Utilities](#)[Borders](#)[Clearfix](#)[Close icon](#)[Colors](#)[Display](#)[Embed](#)[Flex](#)[Float](#)[Image replacement](#)[Interactions](#)[Overflow](#)[Position](#)[Screen readers](#)[Shadows](#)[Sizing](#)[Spacing](#)[Stretched link](#)[Text](#)[Vertical align](#)[Visibility](#)[Extend](#)[Migration](#)[About](#)

Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Easily clear `floats` by adding `.clearfix` to the parent element. Can also be used as a mixin.

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

Copy

```
// Mixin itself
@mixin clearfix() {
  &::after {
    display: block;
    content: "";
    clear: both;
  }
}

// Usage as a mixin
.element {
  @include clearfix;
}
```

Copy

The following example shows how the clearfix can be used. Without the clearfix the wrapping div would not span around the buttons which would cause a broken layout.

Example Button floated left

Example Button floated right

```
<div class="bg-info clearfix">
  <button type="button" class="btn btn-secondary float-left">Example Button
  floated left</button>
  <button type="button" class="btn btn-secondary float-right">Example Button
  floated right</button>
</div>
```

Copy

[Getting started](#)[Layout](#)[Content](#)[Components](#)[Utilities](#)[Borders](#)[Clearfix](#)[Close icon](#)[Colors](#)[Display](#)[Embed](#)[Flex](#)[Float](#)[Image replacement](#)[Interactions](#)[Overflow](#)[Position](#)[Screen readers](#)[Shadows](#)[Sizing](#)[Spacing](#)[Stretched link](#)[Text](#)[Vertical align](#)[Visibility](#)[Extend](#)[Migration](#)[About](#)

Close icon

Use a generic close icon for dismissing content like modals and alerts.



Grow your app revenue
with quality demand from
Facebook's global
advertisers
ads via Carbon

Be sure to include text for screen readers, as we've done with `aria-label`.



```
<button type="button" class="close" aria-label="Close">  
  <span aria-hidden="true">&times;</span>  
</button>
```

[Copy](#)

Colors

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Borders](#)

[Clearfix](#)

[Close icon](#)

[Colors](#)

[Display](#)

[Embed](#)

[Flex](#)

[Float](#)

[Image replacement](#)

[Interactions](#)

[Overflow](#)

[Position](#)

[Screen readers](#)

[Shadows](#)

[Sizing](#)

[Spacing](#)

[Stretched link](#)

[Text](#)

[Vertical align](#)

[Visibility](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10
free Adobe Stock images.

ads via Carbon

Color

.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

Copy

```
<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">.text-warning</p>
<p class="text-info">.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>
```

Contextual text classes also work well on anchors with the provided hover and focus states. **Note that the .text-white and .text-muted class has no additional link styling beyond underline.**

[Primary link](#)

[Secondary link](#)

[Success link](#)

[Danger link](#)

[Warning link](#)

[Info link](#)

Light link

[Dark link](#)

[Muted link](#)

White link

Copy

```
<p><a href="#" class="text-primary">Primary link</a></p>
<p><a href="#" class="text-secondary">Secondary link</a></p>
<p><a href="#" class="text-success">Success link</a></p>
<p><a href="#" class="text-danger">Danger link</a></p>
<p><a href="#" class="text-warning">Warning link</a></p>
<p><a href="#" class="text-info">Info link</a></p>
<p><a href="#" class="text-light bg-dark">Light link</a></p>
<p><a href="#" class="text-dark">Dark link</a></p>
<p><a href="#" class="text-muted">Muted link</a></p>
<p><a href="#" class="text-white bg-dark">White link</a></p>
```

Background color

Similar to the contextual text color classes, easily set the background of an element to any contextual class. Anchor components will darken on hover, just like the text classes. Background utilities **do not set** `color`, so in some cases you'll want to use `.text-*` utilities.

.bg-primary

.bg-secondary

.bg-success

.bg-danger

.bg-warning

.bg-info

.bg-light

.bg-dark

.bg-white

.bg-transparent

Copy

```
<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-white">.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-white text-dark">.bg-white</div>
<div class="p-3 mb-2 bg-transparent text-dark">.bg-transparent</div>
```

Background gradient

When `$enable-gradients` is set to `true` (default is `false`), you can use `.bg-gradient-` utility classes. [Learn about our Sass options](#) to enable these classes and more.

- `.bg-gradient-primary`
- `.bg-gradient-secondary`
- `.bg-gradient-success`
- `.bg-gradient-danger`
- `.bg-gradient-warning`
- `.bg-gradient-info`
- `.bg-gradient-light`
- `.bg-gradient-dark`

Dealing with 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>` with the class.

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 `.sr-only` class.

Display property

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Borders](#)

[Clearfix](#)

[Close icon](#)

[Colors](#)

[Display](#)

[Embed](#)

[Flex](#)

[Float](#)

[Image replacement](#)

[Interactions](#)

[Overflow](#)

[Position](#)

[Screen readers](#)

[Shadows](#)

[Sizing](#)

[Spacing](#)

[Stretched link](#)

[Text](#)

[Vertical align](#)

[Visibility](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10
free Adobe Stock images.

ads via Carbon

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 `xl`, 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`, and `xl`.

Where `value` is one of:

- `none`
- `inline`
- `inline-block`
- `block`
- `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 effect screen widths with the given breakpoint or larger. For example, `.d-lg-none` sets `display: none;` on both `lg` and `xl` screens.

Examples

d-inline d-inline

```
<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>
```

Copy

```
d-block
```

```
d-block
```

```
<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>
```

Copy

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}-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` will hide the element for all screen sizes except on medium and large devices.

Screen Size	Class
Hidden on all	<code>.d-none</code>
Hidden only on xs	<code>.d-none .d-sm-block</code>
Hidden only on sm	<code>.d-sm-none .d-md-block</code>
Hidden only on md	<code>.d-md-none .d-lg-block</code>
Hidden only on lg	<code>.d-lg-none .d-xl-block</code>
Hidden only on xl	<code>.d-xl-none</code>
Visible on all	<code>.d-block</code>
Visible only on xs	<code>.d-block .d-sm-none</code>
Visible only on sm	<code>.d-none .d-sm-block .d-md-none</code>
Visible only on md	<code>.d-none .d-md-block .d-lg-none</code>
Visible only on lg	<code>.d-none .d-lg-block .d-xl-none</code>
Visible only on xl	<code>.d-none .d-xl-block</code>

```
hide on lg and wider screens
```

```
<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>
```

Copy

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-table`
- `.d-print-table-row`
- `.d-print-table-cell`
- `.d-print-flex`
- `.d-print-inline-flex`

The print and display classes can be combined.

Print Only (Hide on screen only)
Hide up to large on screen, but always show on print

```
<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>
```

Copy

Embeds

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Borders](#)

[Clearfix](#)

[Close icon](#)

[Colors](#)

[Display](#)

[Embed](#)

[Flex](#)

[Float](#)

[Image replacement](#)

[Interactions](#)

[Overflow](#)

[Position](#)

[Screen readers](#)

[Shadows](#)

[Sizing](#)

[Spacing](#)

[Stretched link](#)

[Text](#)

[Vertical align](#)

[Visibility](#)

[Extend](#)

[Migration](#)

[About](#)



Cut your cloud bills in half.
Build and deploy your app
with \$100 in credit.

ads via Carbon

About

Rules are directly applied to `<iframe>`, `<embed>`, `<video>`, and `<object>` elements; optionally use an explicit descendant class `.embed-responsive-item` when you want to match the styling for other attributes.

Pro-Tip! You don't need to include `frameborder="0"` in your `<iframe>`s as we override that for you.

Example

Wrap any embed like an `<iframe>` in a parent element with `.embed-responsive` and an aspect ratio. The `.embed-responsive-item` isn't strictly required, but we encourage it.

REO Speedwagon - Can't Fight This Feeling (Official Music Video)

```
<div class="embed-responsive embed-responsive-16by9">
  <iframe class="embed-responsive-item"
    src="https://www.youtube.com/embed/zp0ULjyy-n8?rel=0" allowfullscreen></iframe>
</div>
```

Copy

Aspect ratios

Aspect ratios can be customized with modifier classes. By default the following ratio classes are provided:

Copy

```
<!-- 21:9 aspect ratio -->
<div class="embed-responsive embed-responsive-21by9">
  <iframe class="embed-responsive-item" src="..."></iframe>
</div>

<!-- 16:9 aspect ratio -->
<div class="embed-responsive embed-responsive-16by9">
  <iframe class="embed-responsive-item" src="..."></iframe>
</div>

<!-- 4:3 aspect ratio -->
<div class="embed-responsive embed-responsive-4by3">
  <iframe class="embed-responsive-item" src="..."></iframe>
</div>

<!-- 1:1 aspect ratio -->
<div class="embed-responsive embed-responsive-1by1">
  <iframe class="embed-responsive-item" src="..."></iframe>
</div>
```

Within `_variables.scss`, you can change the aspect ratios you want to use. Here's an example of the `$embed-responsive-aspect-ratios` list:

```
$embed-responsive-aspect-ratios: (
  (21 9),
  (16 9),
  (4 3),
  (1 1)
) !default;
```

Copy

Flex

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Borders](#)

[Clearfix](#)

[Close icon](#)

[Colors](#)

[Display](#)

[Embed](#)

[Flex](#)

[Float](#)

[Image replacement](#)

[Interactions](#)

[Overflow](#)

[Position](#)

[Screen readers](#)

[Shadows](#)

[Sizing](#)

[Spacing](#)

[Stretched link](#)

[Text](#)

[Vertical align](#)

[Visibility](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

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!

```
<div class="d-flex p-2 bd-highlight">I'm a flexbox container!</div>
```

Copy

I'm an inline flexbox container!

```
<div class="d-inline-flex p-2 bd-highlight">I'm an inline flexbox container!</div>
```

Copy

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`

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.



```
<div class="d-flex flex-row bd-highlight mb-3">
  <div class="p-2 bd-highlight">Flex item 1</div>
  <div class="p-2 bd-highlight">Flex item 2</div>
  <div class="p-2 bd-highlight">Flex item 3</div>
</div>
<div class="d-flex flex-row-reverse bd-highlight">
  <div class="p-2 bd-highlight">Flex item 1</div>
  <div class="p-2 bd-highlight">Flex item 2</div>
  <div class="p-2 bd-highlight">Flex item 3</div>
</div>
```

Copy

Use `.flex-column` to set a vertical direction, or `.flex-column-reverse` to start the vertical direction from the opposite side.



```
<div class="d-flex flex-column bd-highlight mb-3">
  <div class="p-2 bd-highlight">Flex item 1</div>
  <div class="p-2 bd-highlight">Flex item 2</div>
  <div class="p-2 bd-highlight">Flex item 3</div>
</div>
<div class="d-flex flex-column-reverse bd-highlight">
  <div class="p-2 bd-highlight">Flex item 1</div>
  <div class="p-2 bd-highlight">Flex item 2</div>
  <div class="p-2 bd-highlight">Flex item 3</div>
</div>
```

Copy

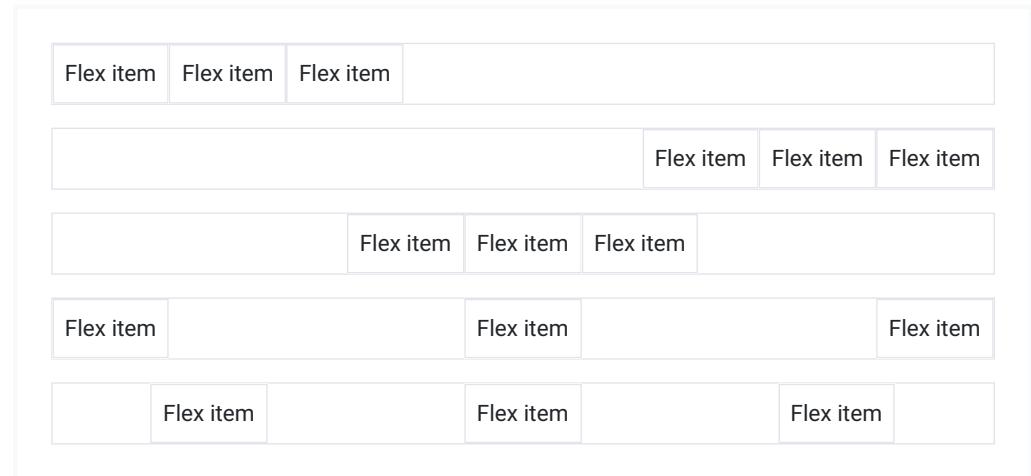
Responsive variations also exist for `flex-direction`.

- `.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

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`, or `around`.



```
<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>
```

Copy

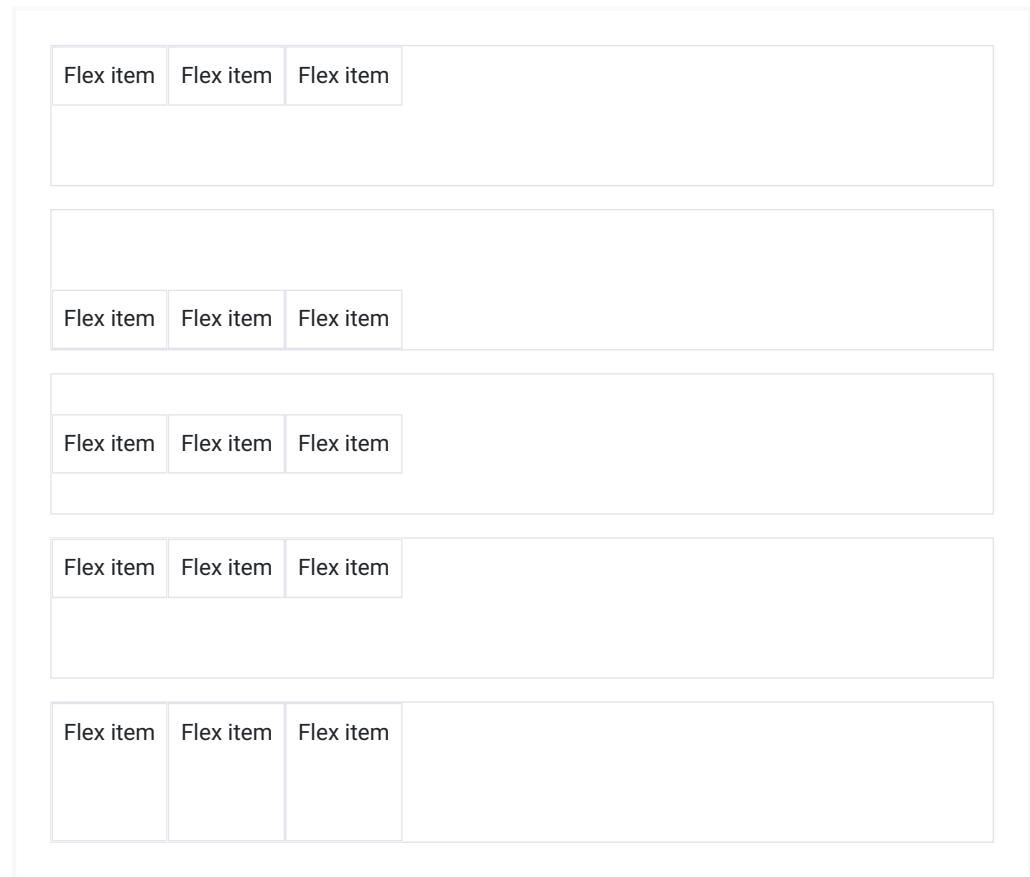
Responsive variations also exist for `justify-content`.

- .justify-content-start
- .justify-content-end
- .justify-content-center
- .justify-content-between
- .justify-content-around
- .justify-content-sm-start
- .justify-content-sm-end
- .justify-content-sm-center
- .justify-content-sm-between
- .justify-content-sm-around
- .justify-content-md-start
- .justify-content-md-end
- .justify-content-md-center
- .justify-content-md-between
- .justify-content-md-around
- .justify-content-lg-start
- .justify-content-lg-end
- .justify-content-lg-center
- .justify-content-lg-between
- .justify-content-lg-around
- .justify-content-xl-start
- .justify-content-xl-end

- `.justify-content-xl-center`
- `.justify-content-xl-between`
- `.justify-content-xl-around`

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).



```
<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>
```

Copy

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 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		Flex item	
	Aligned flex item		
Flex item		Flex item	
	Aligned 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>
```

Copy

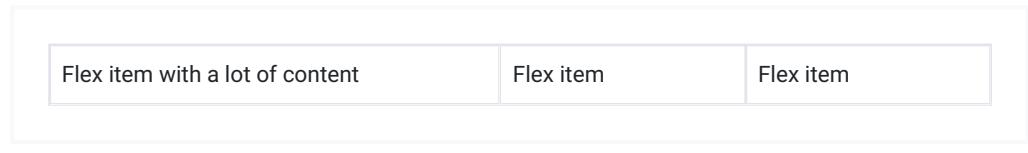
Responsive variations also exist for `align-self`.

- .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

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.



```
<div class="d-flex bd-highlight">
  <div class="p-2 flex-fill bd-highlight">Flex item with a lot of content</div>
  <div class="p-2 flex-fill bd-highlight">Flex item</div>
  <div class="p-2 flex-fill bd-highlight">Flex item</div>
</div>
```

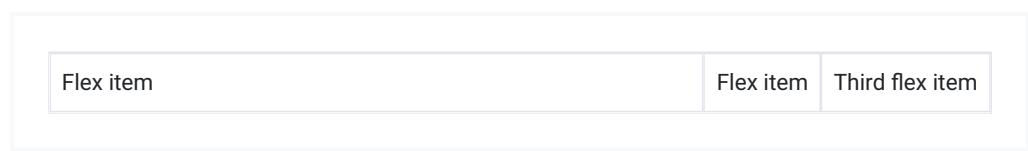
Copy

Responsive variations also exist for `flex-fill`.

- .flex-fill
- .flex-sm-fill
- .flex-md-fill
- .flex-lg-fill
- .flex-xl-fill

Grow and shrink

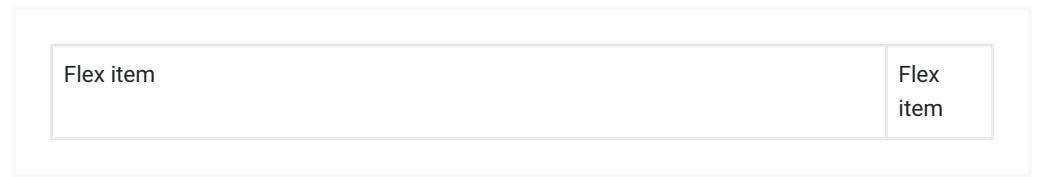
Use `.flex-grow-*` utilities to toggle a flex item's ability to grow to fill available space. In the example below, the `.flex-grow-1` elements uses all available space it can, while allowing the remaining two flex items their necessary space.



```
<div class="d-flex bd-highlight">
  <div class="p-2 flex-grow-1 bd-highlight">Flex item</div>
  <div class="p-2 bd-highlight">Flex item</div>
  <div class="p-2 bd-highlight">Third flex item</div>
</div>
```

Copy

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`.



Copy

```
<div class="d-flex bd-highlight">
  <div class="p-2 w-100 bd-highlight">Flex item</div>
  <div class="p-2 flex-shrink-1 bd-highlight">Flex item</div>
</div>
```

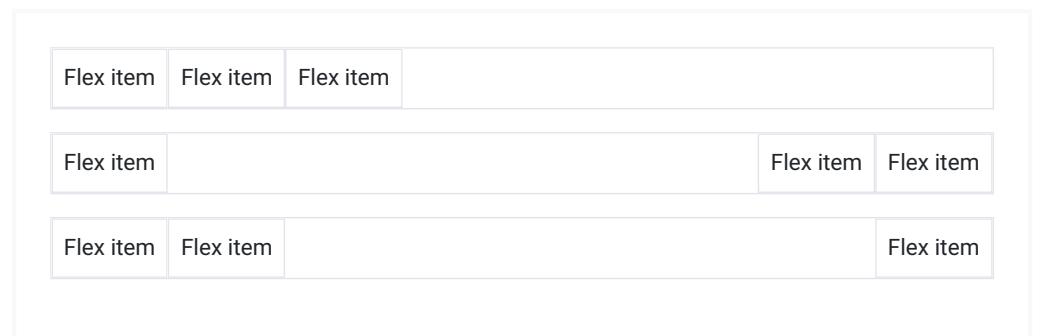
Responsive variations also exist for `flex-grow` and `flex-shrink`.

- `.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`

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 (`.mr-auto`), and pushing two items to the left (`.ml-auto`).

Unfortunately, IE10 and IE11 do not properly support auto margins on flex items whose parent has a non-default `justify-content` value. [See this StackOverflow answer](#) for more details.



Copy

```

<div class="d-flex bd-highlight mb-3">
  <div class="p-2 bd-highlight">Flex item</div>
  <div class="p-2 bd-highlight">Flex item</div>
  <div class="p-2 bd-highlight">Flex item</div>
</div>

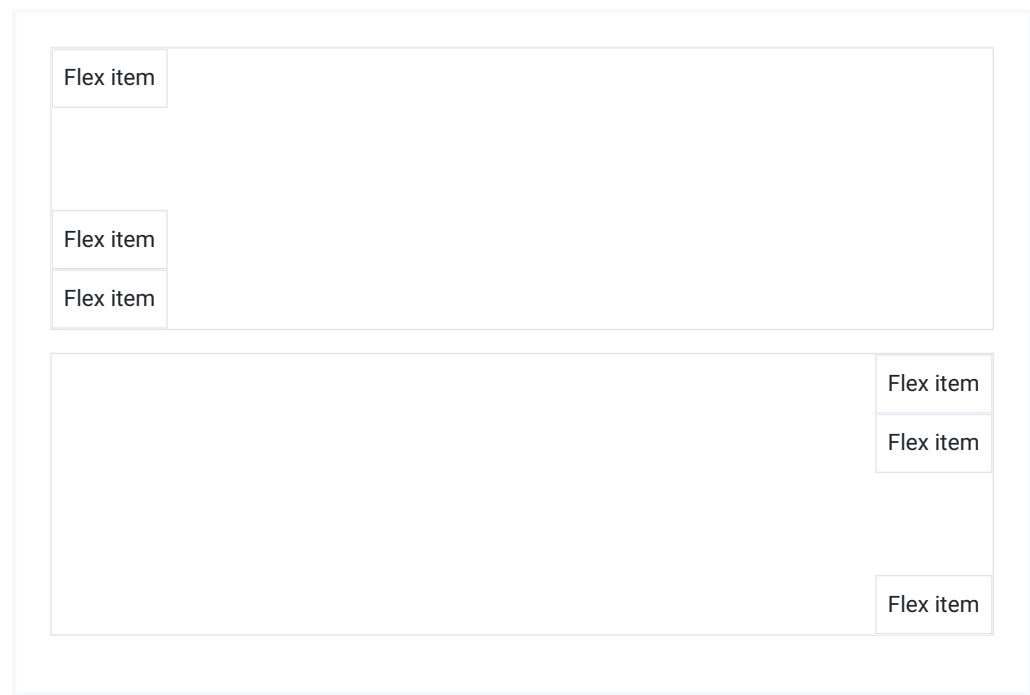
<div class="d-flex bd-highlight mb-3">
  <div class="mr-auto p-2 bd-highlight">Flex item</div>
  <div class="p-2 bd-highlight">Flex item</div>
  <div class="p-2 bd-highlight">Flex item</div>
</div>

<div class="d-flex bd-highlight mb-3">
  <div class="p-2 bd-highlight">Flex item</div>
  <div class="p-2 bd-highlight">Flex item</div>
  <div class="ml-auto p-2 bd-highlight">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`.



```

<div class="d-flex align-items-start flex-column bd-highlight mb-3" style="height: 200px;">
  <div class="mb-auto p-2 bd-highlight">Flex item</div>
  <div class="p-2 bd-highlight">Flex item</div>
  <div class="p-2 bd-highlight">Flex item</div>
</div>

<div class="d-flex align-items-end flex-column bd-highlight mb-3" style="height: 200px;">
  <div class="p-2 bd-highlight">Flex item</div>
  <div class="p-2 bd-highlight">Flex item</div>
  <div class="mt-auto p-2 bd-highlight">Flex item</div>
</div>

```

[Copy](#)

Wrap

Change how flex items wrap in a flex container. Choose from no wrapping at all (the browser default) with `.flexnowrap`, wrapping with `.flex-wrap`, or reverse wrapping with `.flex-wrap-reverse`.

Flex item				
-----------	-----------	-----------	-----------	-----------

```
<div class="d-flex flexnowrap">
  ...
</div>
```

Copy

Flex item							
Flex item							

```
<div class="d-flex flexwrap">
  ...
</div>
```

Copy

Flex item							
Flex item							

```
<div class="d-flex flexwrapreverse">
  ...
</div>
```

Copy

Responsive variations also exist for `flex-wrap`.

- `.flexnowrap`
- `.flexwrap`
- `.flexwrapreverse`
- `.flexsmnowrap`
- `.flexsmwrap`
- `.flexsmwrapreverse`
- `.flexmdnowrap`
- `.flexmdwrap`
- `.flexmdwrapreverse`
- `.flexlgnowrap`
- `.flexlgwrap`
- `.flexlgwrapreverse`
- `.flexxlnowrap`
- `.flexxlwrap`
- `.flexxlwrapreverse`

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 (e.g., `5`), add custom CSS for any additional values needed.

Third flex item	Second flex item	First flex item
-----------------	------------------	-----------------

```
<div class="d-flex flex-wrap bd-highlight">
  <div class="order-3 p-2 bd-highlight">First flex item</div>
  <div class="order-2 p-2 bd-highlight">Second flex item</div>
  <div class="order-1 p-2 bd-highlight">Third flex item</div>
</div>
```

Responsive variations also exist for `order`.

- `.order-0`
- `.order-1`
- `.order-2`
- `.order-3`
- `.order-4`
- `.order-5`
- `.order-6`
- `.order-7`
- `.order-8`
- `.order-9`
- `.order-10`
- `.order-11`
- `.order-12`
- `.order-sm-0`
- `.order-sm-1`
- `.order-sm-2`
- `.order-sm-3`
- `.order-sm-4`
- `.order-sm-5`
- `.order-sm-6`
- `.order-sm-7`
- `.order-sm-8`
- `.order-sm-9`
- `.order-sm-10`
- `.order-sm-11`
- `.order-sm-12`
- `.order-md-0`
- `.order-md-1`
- `.order-md-2`
- `.order-md-3`
- `.order-md-4`
- `.order-md-5`
- `.order-md-6`
- `.order-md-7`
- `.order-md-8`
- `.order-md-9`
- `.order-md-10`
- `.order-md-11`
- `.order-md-12`
- `.order-lg-0`
- `.order-lg-1`
- `.order-lg-2`
- `.order-lg-3`
- `.order-lg-4`
- `.order-lg-5`
- `.order-lg-6`

- .order-lg-7
- .order-lg-8
- .order-lg-9
- .order-lg-10
- .order-lg-11
- .order-lg-12
- .order-xl-0
- .order-xl-1
- .order-xl-2
- .order-xl-3
- .order-xl-4
- .order-xl-5
- .order-xl-6
- .order-xl-7
- .order-xl-8
- .order-xl-9
- .order-xl-10
- .order-xl-11
- .order-xl-12

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							

```
<div class="d-flex align-content-start flex-wrap">
  ...
</div>
```

Copy

Flex item							
Flex item							

```
<div class="d-flex align-content-end flex-wrap">...</div>
```

Copy

Flex item							
Flex item							

```
<div class="d-flex align-content-center flex-wrap">...</div>
```

Copy

Flex item							
Flex item							

```
<div class="d-flex align-content-between flex-wrap">...</div>
```

Copy

Flex item							
Flex item							

```
<div class="d-flex align-content-around flex-wrap">...</div>
```

Copy

Flex item							
Flex item							

```
<div class="d-flex align-content-stretch flex-wrap">...</div>
```

Copy

Responsive variations also exist for `align-content`.

- `.align-content-start`
- `.align-content-end`

- .align-content-center
- .align-content-around
- .align-content-stretch
- .align-content-sm-start
- .align-content-sm-end
- .align-content-sm-center
- .align-content-sm-around
- .align-content-sm-stretch
- .align-content-md-start
- .align-content-md-end
- .align-content-md-center
- .align-content-md-around
- .align-content-md-stretch
- .align-content-lg-start
- .align-content-lg-end
- .align-content-lg-center
- .align-content-lg-around
- .align-content-lg-stretch
- .align-content-xl-start
- .align-content-xl-end
- .align-content-xl-center
- .align-content-xl-around
- .align-content-xl-stretch

Float

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Borders](#)

[Clearfix](#)

[Close icon](#)

[Colors](#)

[Display](#)

[Embed](#)

[Flex](#)

[Float](#)

[Image replacement](#)

[Interactions](#)

[Overflow](#)

[Position](#)

[Screen readers](#)

[Shadows](#)

[Sizing](#)

[Spacing](#)

[Stretched link](#)

[Text](#)

[Vertical align](#)

[Visibility](#)

[Extend](#)

[Migration](#)

[About](#)



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.

Classes

Toggle a float with a class:

Float left on all viewport sizes

Don't float on all viewport sizes

Float right on all viewport sizes

```
<div class="float-left">Float left on all viewport sizes</div><br>
<div class="float-right">Float right on all viewport sizes</div><br>
<div class="float-none">Don't float on all viewport sizes</div>
```

Copy

Mixins

Or by Sass mixin:

```
.element {
  @include float-left;
}
.another-element {
  @include float-right;
}
.one-more {
  @include float-none;
}
```

Copy

Responsive

Responsive variations also exist for each `float` value.

```
Float left on viewports sized SM (small) or wider
Float left on viewports sized MD (medium) or wider
Float left on viewports sized LG (large) or wider
```

```
<div class="float-sm-left">Float left on viewports sized SM (small) or  
wider</div><br>  
<div class="float-md-left">Float left on viewports sized MD (medium) or  
wider</div><br>  
<div class="float-lg-left">Float left on viewports sized LG (large) or  
wider</div><br>  
<div class="float-xl-left">Float left on viewports sized XL (extra-large) or  
wider</div><br>
```

Here are all the support classes;

- `.float-left`
- `.float-right`
- `.float-none`
- `.float-sm-left`
- `.float-sm-right`
- `.float-sm-none`
- `.float-md-left`
- `.float-md-right`
- `.float-md-none`
- `.float-lg-left`
- `.float-lg-right`
- `.float-lg-none`
- `.float-xl-left`
- `.float-xl-right`
- `.float-xl-none`

[Getting started](#)[Layout](#)[Content](#)[Components](#)[Utilities](#)[Borders](#)[Clearfix](#)[Close icon](#)[Colors](#)[Display](#)[Embed](#)[Flex](#)[Float](#)[Image replacement](#)[Interactions](#)[Overflow](#)[Position](#)[Screen readers](#)[Shadows](#)[Sizing](#)[Spacing](#)[Stretched link](#)[Text](#)[Vertical align](#)[Visibility](#)[Extend](#)[Migration](#)[About](#)

Image replacement

Swap text for background images with the image replacement class.



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Warning

The `text-hide()` class and mixin has been deprecated as of v4.1. It will be removed entirely in v5.

Utilize the `.text-hide` class or mixin to help replace an element's text content with a background image.

```
<h1 class="text-hide">Custom heading</h1>
```

Copy

```
// Usage as a mixin
.heading {
  @include text-hide;
}
```

Copy

Use the `.text-hide` class to maintain the accessibility and SEO benefits of heading tags, but want to utilize a `background-image` instead of text.

```
<h1 class="text-hide" style="background-image: url('...');">Bootstrap</h1>
```

Copy

[Getting started](#)[Layout](#)[Content](#)[Components](#)[Utilities](#)[Borders](#)[Clearfix](#)[Close icon](#)[Colors](#)[Display](#)[Embed](#)[Flex](#)[Float](#)[Image replacement](#)[Interactions](#)[Overflow](#)[Position](#)[Screen readers](#)[Shadows](#)[Sizing](#)[Spacing](#)[Stretched link](#)[Text](#)[Vertical align](#)[Visibility](#)[Extend](#)[Migration](#)[About](#)

Interactions

Utility classes that change how users interact with the contents of a website.



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Text selection

Change how the content is selected when the user interacts with it.

This paragraph will be entirely selected when clicked by the user.

This paragraph has the default select behavior.

This paragraph will not be selectable when clicked by the user.

Copy

```
<p class="user-select-all">This paragraph will be entirely selected when clicked by the user.</p>
<p class="user-select-auto">This paragraph has the default select behavior.</p>
<p class="user-select-none">This paragraph will not be selectable when clicked by the user.</p>
```

Customize the available classes by changing the `$user-selects` Sass list in `_variables.scss`.

[Getting started](#)[Layout](#)[Content](#)[Components](#)[Utilities](#)[Borders](#)[clearfix](#)[Close icon](#)[Colors](#)[Display](#)[Embed](#)[Flex](#)[Float](#)[Image replacement](#)[Interactions](#)[Overflow](#)[Position](#)[Screen readers](#)[Shadows](#)[Sizing](#)[Spacing](#)[Stretched link](#)[Text](#)[Vertical align](#)[Visibility](#)[Extend](#)[Migration](#)[About](#)

Overflow

Use these shorthand utilities for quickly configuring how content overflows an element.



Adobe Creative Cloud for
Teams starting at \$33.99
per month.

ads via Carbon

Barebones `overflow` functionality is provided for two values by default, and they are not responsive.

This is an example of using
`.overflow-auto` on an element
with set width and height
dimensions. By design this

This is an example of using
`.overflow-hidden` on an
element with set width and
height dimensions

```
<div class="overflow-auto">...</div>
<div class="overflow-hidden">...</div>
```

Copy

Using Sass variables, you may customize the overflow utilities by changing the `$overflows` variable in `_variables.scss`.

Position

[Getting started](#)[Layout](#)[Content](#)[Components](#)[Utilities](#)[Borders](#)[Clearfix](#)[Close icon](#)[Colors](#)[Display](#)[Embed](#)[Flex](#)[Float](#)[Image replacement](#)[Interactions](#)[Overflow](#)[Position](#)[Screen readers](#)[Shadows](#)[Sizing](#)[Spacing](#)[Stretched link](#)[Text](#)[Vertical align](#)[Visibility](#)[Extend](#)[Migration](#)[About](#)

Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Common 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>
```

Copy

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>
```

Copy

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>
```

Copy

Sticky top

Position an element at the top of the viewport, from edge to edge, but only after you scroll past it. The `.sticky-top` utility uses CSS's `position: sticky`, which isn't fully supported in all browsers.

IE11 and IE10 will render `position: sticky` as `position: relative`. As such, we wrap the styles in a `@supports` query, limiting the stickiness to only browsers that can render it properly.

```
<div class="sticky-top">...</div>
```

Copy

[Getting started](#)[Layout](#)[Content](#)[Components](#)[Utilities](#)[Borders](#)[Clearfix](#)[Close icon](#)[Colors](#)[Display](#)[Embed](#)[Flex](#)[Float](#)[Image replacement](#)[Interactions](#)[Overflow](#)[Position](#)[Screen readers](#)[Shadows](#)[Sizing](#)[Spacing](#)[Stretched link](#)[Text](#)[Vertical align](#)[Visibility](#)[Extend](#)[Migration](#)[About](#)

Screen readers

Use screen reader utilities to hide elements on all devices except screen readers.



Cut your cloud bills in half.
Build and deploy your app
with \$100 in credit.

ads via Carbon

Hide an element to all devices **except screen readers** with `.sr-only`. Combine `.sr-only` with `.sr-only-focusable` to show the element again when it's focused (e.g. by a keyboard-only user). Can also be used as mixins.

```
<a class="sr-only sr-only-focusable" href="#content">Skip to main content</a>
```

Copy

```
// Usage as a mixin
.skip-navigation {
  @include sr-only;
  @include sr-only-focusable;
}
```

Copy

[Getting started](#)[Layout](#)[Content](#)[Components](#)[Utilities](#)[Borders](#)[Clearfix](#)[Close icon](#)[Colors](#)[Display](#)[Embed](#)[Flex](#)[Float](#)[Image replacement](#)[Interactions](#)[Overflow](#)[Position](#)[Screen readers](#)[Shadows](#)[Sizing](#)[Spacing](#)[Stretched link](#)[Text](#)[Vertical align](#)[Visibility](#)[Extend](#)[Migration](#)[About](#)

Shadows

Add or remove shadows to elements with box-shadow utilities.



Deploy more with our Linux virtual machines, global infrastructure, and simple pricing.

ads via Carbon

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

Copy

```
<div class="shadow-none p-3 mb-5 bg-light rounded">No shadow</div>
<div class="shadow-sm p-3 mb-5 bg-white rounded">Small shadow</div>
<div class="shadow p-3 mb-5 bg-white rounded">Regular shadow</div>
<div class="shadow-lg p-3 mb-5 bg-white rounded">Larger shadow</div>
```


Sizing

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Borders](#)

[clearfix](#)

[Close icon](#)

[Colors](#)

[Display](#)

[Embed](#)

[Flex](#)

[Float](#)

[Image replacement](#)

[Interactions](#)

[Overflow](#)

[Position](#)

[Screen readers](#)

[Shadows](#)

[Sizing](#)

[Spacing](#)

[Stretched link](#)

[Text](#)

[Vertical align](#)

[Visibility](#)

[Extend](#)

[Migration](#)

[About](#)



Adobe Creative Cloud for
Teams starting at \$33.99
per month.

ads via Carbon

Relative to the parent

Width and height utilities are generated from the `$sizes` Sass map in `_variables.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

Copy

```
<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

Copy

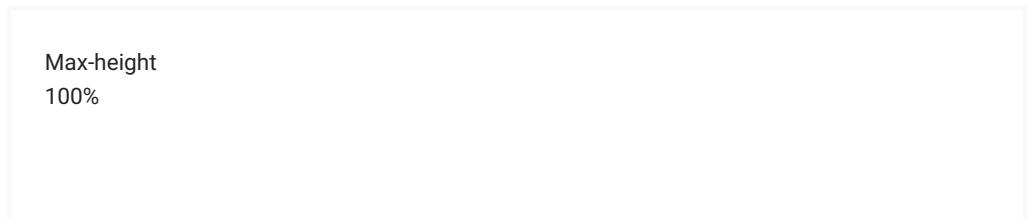
```
<div style="height: 100px; background-color: rgba(255,0,0,0.1);">
  <div class="h-25 d-inline-block" style="width: 120px; background-color: rgba(0,0,255,.1)">Height 25%</div>
  <div class="h-50 d-inline-block" style="width: 120px; background-color: rgba(0,0,255,.1)">Height 50%</div>
  <div class="h-75 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.



Max-width 100%

```
</img>
```

[Copy](#)

Max-height
100%

```
<div style="height: 100px; background-color: rgba(255,0,0,0.1);">  
  <div class="mh-100" style="width: 100px; height: 200px; background-color:  
    rgba(0,0,255,0.1);">Max-height 100%</div>  
</div>
```

[Copy](#)

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>
```

[Copy](#)

Spacing

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Borders](#)

[clearfix](#)

[Close icon](#)

[Colors](#)

[Display](#)

[Embed](#)

[Flex](#)

[Float](#)

[Image replacement](#)

[Interactions](#)

[Overflow](#)

[Position](#)

[Screen readers](#)

[Shadows](#)

[Sizing](#)

[Spacing](#)

[Stretched link](#)

[Text](#)

[Vertical align](#)

[Visibility](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

How it works

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`.

Notation

Spacing utilities that apply to all breakpoints, from `xs` to `xl`, 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`, and `xl`.

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`
- `l` - for classes that set `margin-left` or `padding-left`
- `r` - for classes that set `margin-right` or `padding-right`
- `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;
}

.ml-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>
```

Copy

Negative margin

In CSS, `margin` properties can utilize negative values (`padding` cannot). As of 4.2, we've added negative margin utilities for every non-zero integer size listed above (e.g., `1`, `2`, `3`, `4`, `5`). These utilities are ideal for customizing grid column gutters across breakpoints.

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;
}
```

Copy

Here's an example of customizing the Bootstrap grid at the medium (`md`) breakpoint and above. We've increased the `.col` padding with `.px-md-5` and then counteracted that with `.mx-md-n5` on the parent `.row`.

Custom column padding

Custom column padding

```
<div class="row mx-md-n5">
  <div class="col px-md-5"><div class="p-3 border bg-light">Custom column
padding</div></div>
  <div class="col px-md-5"><div class="p-3 border bg-light">Custom column
padding</div></div>
</div>
```

Copy

Stretched link

Search...

Getting started

Layout

Content

Components

Utilities

Borders

Clearfix

Close icon

Colors

Display

Embed

Flex

Float

Image replacement

Interactions

Overflow

Position

Screen readers

Shadows

Sizing

Spacing

Stretched link

Text

Vertical align

Visibility

Extend

Migration

About



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Add `.stretched-link` to a link to make its [containing block](#) clickable via a `::after` pseudo element. In most cases, this means that an element with `position: relative;` that contains a link with the `.stretched-link` class is clickable.

Cards have `position: relative` by default in Bootstrap, so in this case you can safely add the `.stretched-link` class to a link in the card without any other HTML changes.

Multiple links and tap targets are not recommended with stretched links. However, some `position` and `z-index` styles can help should this be required.



Go somewhere

```
<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">Card with stretched link</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 stretched-link">Go somewhere</a>
  </div>
</div>
```

Copy

Media objects do not have `position: relative` by default, so we need to add the `.position-relative` here to prevent the link from stretching outside the media object.



Media with stretched link

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

[Go somewhere](#)

Copy

```
<div class="media position-relative">
  
  <div class="media-body">
    <h5 class="mt-0">Media with stretched link</h5>
    <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.</p>
    <a href="#" class="stretched-link">Go somewhere</a>
  </div>
</div>
```

Columns are `position: relative` by default, so clickable columns only require the `.stretched-link` class on a link. However, stretching a link over an entire `.row` requires `.position-static` on the column and `.position-relative` on the row.

Columns with stretched link

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

[Go somewhere](#)

Copy

```
<div class="row no-gutters bg-light position-relative">
  <div class="col-md-6 mb-md-0 p-md-4">
    
  </div>
  <div class="col-md-6 position-static p-4 pl-md-0">
    <h5 class="mt-0">Columns with stretched link</h5>
    <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.</p>
    <a href="#" class="stretched-link">Go somewhere</a>
  </div>
</div>
```

Identifying the containing block

If the stretched link doesn't seem to work, the `containing block` will probably be the cause. The following CSS properties will make an element the containing block:

- A `position` value other than `static`
- A `transform` or `perspective` value other than `none`
- A `will-change` value of `transform` or `perspective`
- A `filter` value other than `none` or a `will-change` value of `filter` (only works on Firefox)



Card with stretched links

Some quick example text to build on the card title and make up the bulk of the card's content.

[Stretched link will not work here, because position: relative is added to the link](#)

This [stretched link](#) will only be spread over the `p`-tag, because a transform is applied to it.

Copy

```
<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">Card with stretched links</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>
    <p class="card-text">
      <a href="#" class="stretched-link text-danger" style="position: relative;">Stretched link will not work here, because <code>position: relative</code> is added to the link</a>
    </p>
    <p class="card-text bg-light" style="transform: rotate(0);">
      This <a href="#" class="text-warning stretched-link">stretched link</a>
      will only be spread over the <code>p</code>-tag, because a transform is applied to it.
    </p>
  </div>
</div>
```

Text

Search...

Getting started

Layout

Content

Components

Utilities

Borders

Clearfix

Close icon

Colors

Display

Embed

Flex

Float

Image replacement

Interactions

Overflow

Position

Screen readers

Shadows

Sizing

Spacing

Stretched link

Text

Vertical align

Visibility

Extend

Migration

About



Boost your app
monetization with
Facebook's people-based
advertising solutions
ads via Carbon

Text alignment

Easily realign text to components with text alignment classes.

Ambitioni dedisse scripsisse iudicaretur. Cras mattis iudicium purus sit amet fermentum. Donec sed odio operae, eu vulputate felis rhoncus. Praeterea iter est quasdam res quas ex communi. At nos hinc posthac, sitientis piros Afros. Petierunt uti sibi concilium totius Galliae in diem certam indicere. Cras mattis iudicium purus sit amet fermentum.

<pre><p class="text-justify">Ambitioni dedisse scripsisse iudicaretur. Cras mattis iudicium purus sit amet fermentum. Donec sed odio operae, eu vulputate felis rhoncus. Praeterea iter est quasdam res quas ex communi. At nos hinc posthac, sitientis piros Afros. Petierunt uti sibi concilium totius Galliae in diem certam indicere. Cras mattis iudicium purus sit amet fermentum.</p></pre>

For left, right, and center alignment, responsive classes are available that use the same viewport width breakpoints as the grid system.

Left aligned text on all viewport sizes.

Center aligned text on all viewport sizes.

Right aligned text on all viewport sizes.

Left aligned text on viewports sized SM (small) or wider.

Left aligned text on viewports sized MD (medium) or wider.

Left aligned text on viewports sized LG (large) or wider.

Left aligned text on viewports sized XL (extra-large) or wider.

<pre><p class="text-left">Left aligned text on all viewport sizes.</p><p class="text-center">Center aligned text on all viewport sizes.</p><p class="text-right">Right aligned text on all viewport sizes.</p></pre>

<pre><p class="text-sm-left">Left aligned text on viewports sized SM (small) or wider.</p></pre>

<pre><p class="text-md-left">Left aligned text on viewports sized MD (medium) or wider.</p></pre>

<pre><p class="text-lg-left">Left aligned text on viewports sized LG (large) or wider.</p></pre>

<pre><p class="text-xl-left">Left aligned text on viewports sized XL (extra-large) or wider.</p></pre>

Copy

Copy

Copy

Text wrapping and overflow

Wrap text with a `.text-wrap` class.

This text
should wrap.

```
<div class="badge badge-primary text-wrap" style="width: 6rem;">  
    This text should wrap.  
</div>
```

Copy

Prevent text from wrapping with a `.text nowrap` class.

This text should overflow the parent.

```
<div class="text-nowrap bd-highlight" style="width: 8rem;">  
  This text should overflow the parent.  
</div>
```

Copy

For longer content, you can add a `.text-truncate` class to truncate the text with an ellipsis. **Requires `display: inline-block` or `display: block`.**

Praeterea ...
Praeterea iter est q...

```
<!-- Block level -->
<div class="row">
  <div class="col-2 text-truncate">
    Praeterea iter est quasdam res quas ex communi.
  </div>
</div>

<!-- Inline level -->
<span class="d-inline-block text-truncate" style="m
  Praeterea iter est quasdam res quas ex communi.
</span>
```

Copy

Word break

Prevent long strings of text from breaking your components' layout by using `.text-break` to set `word-wrap: break-word`.

Copy

Text transform

Transform text in components with text capitalization classes.

lowercased text.

UPPERCASED TEXT.

CapiTaliZed Text.

Copy

```
<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 weight and italics

Quickly change the weight (boldness) of text or italicize text.

Bold text.

Bolder weight text (relative to the parent element).

Normal weight text.

Light weight text.

Lighter weight text (relative to the parent element).

Italic text.

Copy

```
<p class="font-weight-bold">Bold text.</p>
<p class="font-weight-bolder">Bolder weight text (relative to the parent
element).</p>
<p class="font-weight-normal">Normal weight text.</p>
<p class="font-weight-light">Light weight text.</p>
<p class="font-weight-lighter">Lighter weight text (relative to the parent
element).</p>
<p class="font-italic">Italic text.</p>
```

Monospace

Change a selection to our monospace font stack with `.text-monospace`.

This is in monospace

Copy

```
<p class="text-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](#).

Copy

```
<p class="text-muted">  
  Muted text with a <a href="#" class="text-reset">reset link</a>.  
</p>
```

Text decoration

Remove a text decoration with a [.text-decoration-none](#) class.

[Non-underlined link](#)

Copy

```
<a href="#" class="text-decoration-none">Non-underlined link</a>
```

[Getting started](#)[Layout](#)[Content](#)[Components](#)[Utilities](#)[Borders](#)[Clearfix](#)[Close icon](#)[Colors](#)[Display](#)[Embed](#)[Flex](#)[Float](#)[Image replacement](#)[Interactions](#)[Overflow](#)[Position](#)[Screen readers](#)[Shadows](#)[Sizing](#)[Spacing](#)[Stretched link](#)[Text](#)[Vertical align](#)[Visibility](#)[Extend](#)[Migration](#)[About](#)

Vertical alignment

Easily change the vertical alignment of inline, inline-block, inline-table, and table cell elements.



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

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.

With inline elements:

```
baseline top middle bottom text-top text-bottom
```

```
<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>
```

Copy

With table cells:

baseline	top	text-top	text-bottom
middle			
bottom			

```
<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>
```

Copy

Visibility

[Getting started](#)[Layout](#)[Content](#)[Components](#)[Utilities](#)[Borders](#)[Clearfix](#)[Close icon](#)[Colors](#)[Display](#)[Embed](#)[Flex](#)[Float](#)[Image replacement](#)[Interactions](#)[Overflow](#)[Position](#)[Screen readers](#)[Shadows](#)[Sizing](#)[Spacing](#)[Stretched link](#)[Text](#)[Vertical align](#)[Visibility](#)[Extend](#)[Migration](#)[About](#)

Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

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. Content 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>
```

Copy

```
// Class
.visible {
  visibility: visible !important;
}
.invisible {
  visibility: hidden !important;
}

// Usage as a mixin
// Warning: The `invisible()` mixin has been deprecated as of v4.3.0. It will be
// removed entirely in v5.
.element {
  @include invisible(visible);
}
.element {
  @include invisible(hidden);
}
```

Copy

Grid system

Search...

[Getting started](#)

[Layout](#)

[Overview](#)

[Grid](#)

[Utilities for layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

How it works

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 look at how the grid comes together.

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

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

```
<div class="container">
  <div class="row">
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
  </div>
```

Copy

The above example creates three equal-width columns on small, medium, large, and extra large devices using our predefined grid classes. Those columns are centered in the page with the parent `.container`.

Breaking it down, here's how it works:

- Containers provide a means to center and horizontally pad your site's contents. Use `.container` for a responsive pixel width or `.container-fluid` for `width: 100%` across all viewport and device sizes.
- 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. This way, all the content in your columns is visually aligned down the left side.
- In a grid layout, content must be placed within columns and only columns may be immediate children of rows.
- Thanks to flexbox, grid columns without a specified `width` will automatically layout as equal width columns. For example, four instances of `.col-sm` will each automatically be 25% wide from the small breakpoint and up. See the [auto-layout columns](#) section for more examples.
- Column classes indicate the number of columns you'd like to use out of the possible 12 per row. So, if you want three equal-width columns across, you can use `.col-4`.
- Column `widths` are set in percentages, so they're always fluid and sized relative to their parent element.
- Columns have horizontal `padding` to create the gutters between individual columns, however, you can remove the `margin` from rows and `padding` from columns with `.no-gutters` on the `.row`.
- To make the grid responsive, there are five grid breakpoints, one for each [responsive breakpoint](#): all breakpoints (extra small), small, medium, large, and extra large.
- Grid breakpoints are based on minimum width media queries, meaning **they apply to that one breakpoint and all those above it** (e.g., `.col-sm-4` applies to small, medium, large, and extra large devices, but not the first `xs` breakpoint).
- You can use predefined grid classes (like `.col-4`) or [Sass mixins](#) for more semantic markup.

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

Grid options

While Bootstrap uses `ems` or `rems` for defining most sizes, `pxs` are used for grid breakpoints and container widths. This is because the viewport width is in pixels and does not change with the [font size](#).

See how aspects of the Bootstrap grid system work across multiple devices with a handy table.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
--	-----------------------	-----------------	------------------	-----------------	------------------------

	Extra small ≤576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				
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 `xl`. 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

```
<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>
```

Copy

Equal-width multi-line

Create equal-width columns that span multiple lines by inserting a `.w-100` where you want the columns to break to a new line. Make the breaks responsive by mixing `.w-100` with some [responsive display utilities](#).

There was a [Safari flexbox bug](#) that prevented this from working without an explicit `flex-basis` or `border`. There are workarounds for older browser versions, but they shouldn't be necessary if your target browsers don't fall into the buggy versions.

col	col
col	col

```
<div class="container">
  <div class="row">
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="w-100"></div>
    <div class="col">col</div>
    <div class="col">col</div>
  </div>
</div>
```

Copy

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

```
<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>
```

Copy

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

```
<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>
```

Copy

Responsive classes

Bootstrap's grid includes five 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

[Copy](#)

```
<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

[Copy](#)

```
<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	

[Copy](#)

```
<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>
```

Gutters

Gutters can be responsively adjusted by breakpoint-specific padding and negative margin utility classes. To change the gutters in a given row, pair a negative margin utility on the `.row` and matching padding utilities on the `.cols`. The `.container` or `.container-fluid` parent may need to be adjusted too to avoid unwanted overflow, using again matching padding utility.

Here's an example of customizing the Bootstrap grid at the large (`lg`) breakpoint and above. We've increased the `.col` padding with `.px-lg-5`, counteracted that with `.mx-lg-n5` on the parent `.row` and then adjusted the `.container` wrapper with `.px-lg-5`.

Custom column padding	Custom column padding
-----------------------	-----------------------

```
<div class="container px-lg-5">
  <div class="row mx-lg-n5">
    <div class="col py-3 px-lg-5 border bg-light">Custom column padding</div>
    <div class="col py-3 px-lg-5 border bg-light">Custom column padding</div>
  </div>
</div>
```

Copy

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.

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

Column	Column
Column	Column

```
<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>
```

Copy

Column	Column	Column
Column		

```
<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>
```

Copy

Column	Column	Column	Column
Column			

```
<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>
```

Copy

Column	Column	Column
Column		

Copy

```
<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
--------	--------	--------	--------

Copy

```
<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()`:

Copy

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

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

Alignment

Use flexbox alignment utilities to vertically and horizontally align columns. Internet Explorer 10-11 do not support vertical alignment of flex items when the flex container has a `min-height` as shown below. [See Flexbugs #3 for more details.](#)

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
----------------------	----------------------	----------------------

Copy

```

<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>

```

One of three columns

One of three columns

One of three columns

Copy

```

<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>

```

Horizontal alignment

One of two columns

Copy

```

<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>

```

No gutters

The gutters between columns in our predefined grid classes can be removed with `.no-gutters`. This removes the negative `margins` from `.row` and the horizontal `padding` from all immediate children columns.

Here's the source code for creating these styles. Note that column overrides are scoped to only the first children columns and are targeted via [attribute selector](#). While this generates a more specific selector, column padding can still be further customized with [spacing utilities](#).

Need an edge-to-edge design? Drop the parent `.container` or `.container-fluid`.

```

.no-gutters {
  margin-right: 0;
  margin-left: 0;

  > .col,
  > [class*="col-"] {
    padding-right: 0;
    padding-left: 0;
  }
}

```

Copy

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

Copy

```

<div class="row no-gutters">
  <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>

```

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.

Copy

```
<div class="container">
  <div class="row">
    <div class="col-9".col-9</div>
    <div class="col-4".col-4<br>Since 9 + 4 = 13 > 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 `.rows`, 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

Copy

```
<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

Copy

```
<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 12 across all five grid tiers.

First in DOM, no order applied

Third in DOM, with an order of 1

Second in DOM, with a larger order

Copy

```

<div class="container">
  <div class="row">
    <div class="col">
      First in DOM, no order applied
    </div>
    <div class="col order-12">
      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: 13 (order: $columns + 1)`, respectively. These classes can also be intermixed with the numbered `.order-*` classes as needed.

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

[Copy](#)

```

<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
-----------	------------------------

[Copy](#)

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

[Copy](#)

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

[Copy](#)

```

<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
--------------------	---

[Copy](#)

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

[Copy](#)

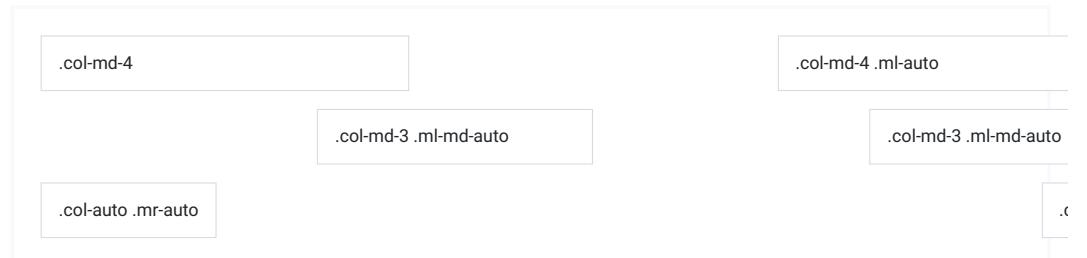
```

<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 `.mr-auto` to force sibling columns away from one another.



[Copy](#)

```

<div class="container">
  <div class="row">
    <div class="col-md-4".col-md-4</div>
    <div class="col-md-4 ml-auto".col-md-4 .ml-auto</div>
  </div>
  <div class="row">
    <div class="col-md-3 ml-md-auto".col-md-3 .ml-md-auto</div>
    <div class="col-md-3 ml-md-auto".col-md-3 .ml-md-auto</div>
  </div>
  <div class="row">
    <div class="col-auto mr-auto".col-auto .mr-auto</div>
    <div class="col-auto".col-auto</div>
  </div>
</div>

```

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).



[Copy](#)

```

<div class="container">
  <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>

```

Sass mixins

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: 30px;

$grid-breakpoints: (
  // Extra small screen / phone
  xs: 0,
  // Small screen / phone
  sm: 576px,
  // Medium screen / tablet
  md: 768px,
  // Large screen / desktop
  lg: 992px,
  // Extra large screen / wide desktop
  xl: 1200px
);

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

Copy

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();
@include make-col($size, $columns: $grid-columns);

// Get fancy by offsetting, or changing the sort order
@include make-col-offset($size, $columns: $grid-columns);
```

Copy

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);
  }
}
```

Copy

Main content

Secondary content

Copy

```
<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: 30px !default;
```

Copy

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
);
```

Copy

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 `%`).

Overview

Search...

[Getting started](#)

[Layout](#)

[Overview](#)

[Grid](#)

[Utilities for layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10
free Adobe Stock images.

ads via Carbon

Containers

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

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	Extra large ≥1200px
<code>.container</code>	100%	540px	720px	960px	1140px
<code>.container-sm</code>	100%	540px	720px	960px	1140px
<code>.container-md</code>	100%	100%	720px	960px	1140px
<code>.container-lg</code>	100%	100%	100%	960px	1140px
<code>.container-xl</code>	100%	100%	100%	100%	1140px
<code>.container-fluid</code>	100%	100%	100%	100%	100%

All-in-one

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

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

Copy

Fluid

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

...

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

Copy

Responsive

Responsive containers are new in Bootstrap v4.4. They allow you to specify a class that is 100% wide until the specified breakpoint is reached, after which we apply `max-widths` 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`, and `.`

```
<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>
```

Copy

Responsive breakpoints

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.

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

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

Copy

Since we write our source CSS in Sass, all our media queries are available via Sass 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) { ... }  
  
// 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;  
  }  
}
```

Copy

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

Copy

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

// Small devices (landscape phones, less than 768px)
@media (max-width: 767.98px) { ... }

// Medium devices (tablets, less than 992px)
@media (max-width: 991.98px) { ... }

// Large devices (desktops, less than 1200px)
@media (max-width: 1199.98px) { ... }

// Extra large devices (large desktops)
// No media query since the extra-large breakpoint has no upper bound on its
width
```

Note that since browsers do not currently support [range context queries](#), 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 for these comparisons.

Once again, these media queries are also available via Sass mixins:

Copy

```
@include media-breakpoint-down(xs) { ... }
@include media-breakpoint-down(sm) { ... }
@include media-breakpoint-down(md) { ... }
@include media-breakpoint-down(lg) { ... }
// No media query necessary for xl breakpoint as it has no upper bound on its
width

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

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

Copy

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

// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) and (max-width: 767.98px) { ... }

// Medium devices (tablets, 768px and up)
@media (min-width: 768px) and (max-width: 991.98px) { ... }

// Large devices (desktops, 992px and up)
@media (min-width: 992px) and (max-width: 1199.98px) { ... }

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

These media queries are also available via Sass mixins:

Copy

```
@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) { ... }
```

Similarly, media queries may span multiple breakpoint widths:

Copy

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

The Sass mixin for targeting the same screen size range would be:

Copy

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

Z-index

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.

Copy

```
$zindex-dropdown:      1000 !default;
$zindex-sticky:       1020 !default;
$zindex-fixed:        1030 !default;
$zindex-modal-backdrop: 1040 !default;
$zindex-modal:         1050 !default;
$zindex-popover:      1060 !default;
$zindex-tooltip:       1070 !default;
```

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.

Utilities for layout

[Getting started](#)

[Layout](#)

[Overview](#)

[Grid](#)

[Utilities for layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Students and Teachers,
save up to 60% on Adobe
Creative Cloud.

ads via Carbon

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 4 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 4 includes a five-level scale for spacing utilities, based on a `1rem` value default `$spacer` variable. Choose values for all viewports (e.g., `.mr-3` for `margin-right: 1rem`), or pick responsive variants to target specific viewports (e.g., `.mr-md-3` for `margin-right: 1rem` 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.

Approach

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Approach](#)

[Icons](#)

[Migration](#)

[About](#)



Students and Teachers,
save up to 60% on Adobe
Creative Cloud.

ads via Carbon

While the getting started pages provide an introductory tour of the project and what it offers, this document focuses on *why* we do the things we do in Bootstrap. It explains our philosophy to building on the web so that others can learn from us, contribute with us, and help us improve.

See something that doesn't sound right, or perhaps could be done better? [Open an issue](#)—we'd love to discuss it with you.

Summary

We'll dive into each of these more throughout, but at a high level, here's what guides our approach.

- Components should be responsive and mobile-first
- Components should be built with a base class and extended via modifier classes
- Component states should obey a common z-index scale
- Whenever possible, prefer a HTML and CSS implementation over JavaScript
- Whenever possible, use utilities over custom styles
- Whenever possible, avoid enforcing strict HTML requirements (children selectors)

Responsive

Bootstrap's responsive styles are built to be responsive, an approach that's often referred to as *mobile-first*. We use this term in our docs and largely agree with it, but at times it can be too broad. While not every component *must* be entirely responsive in Bootstrap, this responsive approach is about reducing CSS overrides by pushing you to add styles as the viewport becomes larger.

Across Bootstrap, you'll see this most clearly in our media queries. In most cases, we use `min-width` queries that begin to apply at a specific breakpoint and carry up through the higher breakpoints. For example, a `.d-none` applies from `min-width: 0` to infinity. On the other hand, a `.d-md-none` applies from the medium breakpoint and up.

At times we'll use `max-width` when a component's inherent complexity requires it. At times, these overrides are functionally and mentally clearer to implement and support than rewriting core functionality from our components. We strive to limit this approach, but will use it from time to time.

Classes

Aside from our Reboot, a cross-browser normalization stylesheet, all our styles aim to use classes as selectors. This means steering clear of type selectors (e.g., `input[type="text"]`) and extraneous parent classes (e.g., `.parent .child`) that make styles too specific to easily override.

As such, components should be built with a base class that houses common, not-to-be overridden property-value pairs. For example, `.btn` and `.btn-primary`. We use `.btn` for all the common styles like `display`, `padding`, and `border-width`. We then use modifiers like `.btn-primary` to add the color, background-color, border-color, etc.

Modifier classes should only be used when there are multiple properties or values to be changed across multiple variants. Modifiers are not always necessary, so be sure you're actually saving lines of code and preventing unnecessary overrides when creating them. Good examples of modifiers are our theme color classes and size variants.

z-index scales

There are two `z-index` scales in Bootstrap—elements within a component and overlay components.

Component elements

- Some components in Bootstrap are built with overlapping elements to prevent double borders without modifying the `border` property. For example, button groups, input groups, and pagination.
- These components share a standard `z-index` scale of `0` through `3`.
- `0` is default (initial), `1` is `:hover`, `2` is `:active/.active`, and `3` is `:focus`.
- This approach matches our expectations of highest user priority. If an element is focused, it's in view and at the user's attention. Active elements are second highest because they indicate state. Hover is third highest because it indicates user intent, but nearly *anything* can be hovered.

Overlay components

Bootstrap includes several components that function as an overlay of some kind. This includes, in order of highest `z-index`, dropdowns, fixed and sticky navbars, modals, tooltips, and popovers. These components have their own `z-index` scale that begins at `1000`. This starting number was chosen arbitrarily and serves as a small buffer between our styles and your project's custom styles.

Each overlay component increases its `z-index` value slightly in such a way that common UI principles allow user focused or hovered elements to remain in view at all times. For example, a modal is document blocking (e.g., you cannot take any other action save for the modal's action), so we put that above our navbars.

Learn more about this in our [z-index layout page](#).

HTML and CSS over JS

Whenever possible, we prefer to write HTML and CSS over JavaScript. In general, HTML and CSS are more prolific and accessible to more people of all different experience levels. HTML and CSS are also faster in your browser than JavaScript, and your browser generally provides a great deal of functionality for you.

This principle is our first-class JavaScript API using `data` attributes. You don't need to write nearly any JavaScript to use our JavaScript plugins; instead, write HTML. Read more about this in [our JavaScript overview page](#).

Lastly, our styles build on the fundamental behaviors of common web elements. Whenever possible, we prefer to use what the browser provides. For example, you can put a `.btn` class on nearly any element, but most elements don't provide any semantic value or browser functionality. So instead, we use `<button>`s and `<a>`s.

The same goes for more complex components. While we *could* write our own form validation plugin to add classes to a parent element based on an input's state, thereby allowing us to style the text say red, we prefer using the `:valid/:invalid` pseudo-elements every browser provides us.

Utilities

Utility classes—formerly helpers in Bootstrap 3—are a powerful ally in combatting CSS bloat and poor page performance. A utility class is typically a single, immutable property-value pairing expressed as a class (e.g., `.d-block` represents `display: block;`). Their primary appeal is speed of use while writing HTML and limiting the amount of custom CSS you have to write.

Specifically regarding custom CSS, utilities can help combat increasing file size by reducing your most commonly repeated property-value pairs into single classes. This can have a dramatic effect at scale in your projects.

Flexible HTML

While not always possible, we strive to avoid being overly dogmatic in our HTML requirements for components. Thus, we focus on single classes in our CSS selectors and try to avoid immediate children selectors (`>`). This gives you more flexibility in your implementation and helps keep our CSS simpler and less specific.

Icons

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Approach](#)

[Icons](#)

[Migration](#)

[About](#)



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

Bootstrap doesn't include an icon library by default, but we have a handful of recommendations for you to choose from. While most icon sets include multiple file formats, we prefer SVG implementations for their improved accessibility and vector support.

Preferred

We've tested and used these icon sets ourselves.

- [Font Awesome](#)
- [Feather](#)
- [Octicons](#)

More options

While we haven't tried these out, they do look promising and provide multiple formats—including SVG.

- [Bytesize](#)
- [Google Material icons](#)
- [Ionicons](#)
- [Dripicons](#)
- [Ikons](#)
- [Icons8](#)

Migrating to v4

Search...

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)



Limited time offer: Get 10
free Adobe Stock images.

ads via Carbon

Stable changes

Moving from Beta 3 to our stable v4.x release, there are no breaking changes, but there are some notable changes.

Printing

- Fixed broken print utilities. Previously, using a `.d-print-*` class would unexpectedly overrule any other `.d-*` class. Now, they match our other display utilities and only apply to that media (`@media print`).
- Expanded available print display utilities to match other utilities. Beta 3 and older only had `block`, `inline-block`, `inline`, and `none`. Stable v4 added `flex`, `inline-flex`, `table`, `table-row`, and `table-cell`.
- Fixed print preview rendering across browsers with new print styles that specify `@page size`.

Beta 3 changes

While Beta 2 saw the bulk of our breaking changes during the beta phase, but we still have a few that needed to be addressed in the Beta 3 release. These changes apply if you're updating to Beta 3 from Beta 2 or any older version of Bootstrap.

Miscellaneous

- Removed the unused `$thumbnail-transition` variable. We weren't transitioning anything, so it was just extra code.
- The npm package no longer includes any files other than our source and dist files; if you relied on them and were running our scripts via the `node_modules` folder, you should adapt your workflow.

Forms

- Rewrote both custom and default checkboxes and radios. Now, both have matching HTML structure (outer `<div>` with sibling `<input>` and `<label>`) and the same layout styles (stacked default, inline with modifier class). This allows us to style the label based on the input's state, simplifying support for the `disabled` attribute (previously requiring a parent class) and better supporting our form validation.

As part of this, we've changed the CSS for managing multiple `background-images` on custom form checkboxes and radios. Previously, the now removed `.custom-control-indicator` element had the background color, gradient, and SVG icon. Customizing the background gradient meant replacing all of those every time you needed to change just one. Now, we have `.custom-control-label::before` for the fill and gradient and `.custom-control-label::after` handles the icon.

To make a custom check inline, add `.custom-control-inline`.

- Updated selector for input-based button groups. Instead of `[data-toggle="buttons"] { }` for style and behavior, we use the `data` attribute just for JS behaviors and rely on a new `.btn-group-toggle` class for styling.
- Removed `.col-form-legend` in favor of a slightly improved `.col-form-label`. This way `.col-form-label-sm` and `.col-form-label-lg` can be used on `<legend>` elements with ease.
- Custom file inputs received a change to their `$custom-file-text` Sass variable. It's no longer a nested Sass map and now only powers one string—the `Browse` button as that is now the only pseudo-element generated from our Sass. The `Choose file` text now comes from the `.custom-file-label`.

Input groups

- Input group addons are now specific to their placement relative to an input. We've dropped `.input-group-addon` and `.input-group-btn` for two new classes, `.input-group-prepend` and `.input-group-append`. You must explicitly use an append or a prepend now, simplifying much of our CSS. Within an append or prepend, place your buttons as they would exist anywhere else, but wrap text in `.input-group-text`.
- Validation styles are now supported, as are multiple inputs (though you can only validate one input per group).
- Sizing classes must be on the parent `.input-group` and not the individual form elements.

Beta 2 changes

While in beta, we aim to have no breaking changes. However, things don't always go as planned. Below are the breaking changes to bear in mind when moving from Beta 1 to Beta 2.

Breaking

- Removed `$badge-color` variable and its usage on `.badge`. We use a color contrast function to pick a `color` based on the `background-color`, so the variable is unnecessary.
- Renamed `grayscale()` function to `gray()` to avoid breaking conflict with the CSS native `grayscale` filter.
- Renamed `.table-inverse`, `.thead-inverse`, and `.thead-default` to `.*-dark` and `.*-light`, matching our color schemes used elsewhere.
- Responsive tables now generate classes for each grid breakpoint. This breaks from Beta 1 in that the `.table-responsive` you've been using is more like `.table-responsive-md`. You may now use `.table-responsive` or `.table-responsive-{sm,md,lg,xl}` as needed.
- Dropped Bower support as the package manager has been deprecated for alternatives (e.g., Yarn or npm). [See bower/bower#2298](#) for details.
- Bootstrap still requires jQuery 1.9.1 or higher, but you're advised to use version 3.x since v3.x's supported browsers are the ones Bootstrap supports plus v3.x has some security fixes.
- Removed the unused `.form-control-label` class. If you did make use of this class, it was duplicate of the `.col-form-label` class that vertically centered a `<label>` with its associated input in horizontal form layouts.
- Changed the `color-yiq` from a mixin that included the `color` property to a function that returns a value, allowing you to use it for any CSS property. For example, instead of `color-yiq(#000)`, you'd write `color: color-yiq(#000);`.

Highlights

- Introduced new `pointer-events` usage on modals. The outer `.modal-dialog` passes through events with `pointer-events: none` for custom click handling (making it possible to just listen on the `.modal-backdrop` for any clicks), and then counteracts it for the actual `.modal-content` with `pointer-events: auto`.

Summary

Here are the big ticket items you'll want to be aware of when moving from v3 to v4.

Browser support

- Dropped IE8, IE9, and iOS 6 support. v4 is now only IE10+ and iOS 7+. For sites needing either of those, use v3.
- Added official support for Android v5.0 Lollipop's Browser and WebView. Earlier versions of the Android Browser and WebView remain only unofficially supported.

Global changes

- **Flexbox is enabled by default.** In general this means a move away from floats and more across our components.
- Switched from [Less](#) to [Sass](#) for our source CSS files.
- Switched from `px` to `rem` as our primary CSS unit, though pixels are still used for media queries and grid behavior as device viewports are not affected by type size.
- Global font-size increased from `14px` to `16px`.
- Revamped grid tiers to add a fifth option (addressing smaller devices at `576px` and below) and removed the `-xs` infix from those classes. Example: `.col-6.col-sm-4.col-md-3`.
- Replaced the separate optional theme with configurable options via SCSS variables (e.g., `$enable-gradients: true`).
- Build system overhauled to use a series of npm scripts instead of Grunt. See [package.json](#) for all scripts, or our project readme for local development needs.
- Non-responsive usage of Bootstrap is no longer supported.
- Dropped the online Customizer in favor of more extensive setup documentation and customized builds.
- Added dozens of new [utility classes](#) for common CSS property-value pairs and margin/padding spacing shortcuts.

Grid system

- **Moved to flexbox.**
 - Added support for flexbox in the grid mixins and predefined classes.
 - As part of flexbox, included support for vertical and horizontal alignment classes.
- **Updated grid class names and a new grid tier.**
 - Added a new `sm` grid tier below `768px` for more granular control. We now have `xs`, `sm`, `md`, `lg`, and `xl`. This also means every tier has been bumped up one level (so `.col-md-6` in v3 is now `.col-lg-6` in v4).
 - `xs` grid classes have been modified to not require the infix to more accurately represent that they start applying styles at `min-width: 0` and not a set pixel value. Instead of `.col-xs-6`, it's now `.col-6`. All other grid tiers require the infix (e.g., `sm`).
- **Updated grid sizes, mixins, and variables.**
 - Grid gutters now have a Sass map so you can specify specific gutter widths at each breakpoint.
 - Updated grid mixins to utilize a `make-col-ready` prep mixin and a `make-col` to set the `flex` and `max-width` for individual column sizing.
 - Changed grid system media query breakpoints and container widths to account for new grid tier and ensure columns are evenly divisible by `12` at their max width.
 - Grid breakpoints and container widths are now handled via Sass maps (`$grid-breakpoints` and `$container-max-widths`) instead of a handful of separate variables. These replace the `@screen-*` variables entirely and allow you to fully customize the grid tiers.
 - Media queries have also changed. Instead of repeating our media query declarations with the same value each time, we now have `@include media-breakpoint-up/down/only`. Now, instead of writing `@media (min-width: @screen-sm-min) { ... }`, you can write `@include media-breakpoint-up(sm) { ... }`.

Components

- **Dropped panels, thumbnails, and wells** for a new all-encompassing component, [cards](#).
- **Dropped the Glyphicons icon font.** If you need icons, some options are:
 - the upstream version of [Glyphicons](#)

- [Octicons](#)
- [Font Awesome](#)
- See the [Extend page](#) for a list of alternatives. Have additional suggestions? Please open an issue or PR.
- **Dropped the Affix jQuery plugin.**
 - We recommend using `position: sticky` instead. [See the HTML5 Please entry](#) for details and specific polyfill recommendations. One suggestion is to use an `@supports` rule for implementing it (e.g., `@supports (position: sticky) { ... }`)
 - If you were using Affix to apply additional, non-`position` styles, the polyfills might not support your use case. One option for such uses is the third-party [ScrollPos-Style](#) library.
- **Dropped the pager component** as it was essentially slightly customized buttons.
- **Refactored nearly all components** to use more un-nested class selectors instead of over-specific children selectors.

By component

This list highlights key changes by component between v3.x.x and v4.0.0.

Reboot

New to Bootstrap 4 is the [Reboot](#), a new stylesheet that builds on Normalize with our own somewhat opinionated reset styles. Selectors appearing in this file only use elements—there are no classes here. This isolates our reset styles from our component styles for a more modular approach. Some of the most important resets this includes are the `box-sizing: border-box` change, moving from `em` to `rem` units on many elements, link styles, and many form element resets.

Typography

- Moved all `.text-` utilities to the [_utilities.scss](#) file.
- Dropped `.page-header` as its styles can be applied via utilities.
- `.dl-horizontal` has been dropped. Instead, use `.row` on `<dl>` and use grid column classes (or mixins) on its `<dt>` and `<dd>` children.
- Redesigned blockquotes, moving their styles from the `<blockquote>` element to a single class, `.blockquote`. Dropped the `.blockquote-reverse` modifier for text utilities.
- `.list-inline` now requires that its children list items have the new `.list-inline-item` class applied to them.

Images

- Renamed `.img-responsive` to `.img-fluid`.
- Renamed `.img-rounded` to `.rounded`
- Renamed `.img-circle` to `.rounded-circle`

Tables

- Nearly all instances of the `>` selector have been removed, meaning nested tables will now automatically inherit styles from their parents. This greatly simplifies our selectors and potential customizations.
- Renamed `.table-condensed` to `.table-sm` for consistency.
- Added a new `.table-inverse` option.
- Added table header modifiers: `.thead-default` and `.thead-inverse`.
- Renamed contextual classes to have a `.table--prefix`. Hence `.active`, `.success`, `.warning`, `.danger` and `.info` to `.table-active`, `.table-success`, `.table-warning`, `.table-danger` and `.table-info`.

Forms

- Moved element resets to the [_reboot.scss](#) file.
- Renamed `.control-label` to `.col-form-label`.
- Renamed `.input-lg` and `.input-sm` to `.form-control-lg` and `.form-control-sm`, respectively.

- Dropped `.form-group-*` classes for simplicity's sake. Use `.form-control-*` classes instead now.
- Dropped `.help-block` and replaced it with `.form-text` for block-level help text. For inline help text and other flexible options, use utility classes like `.text-muted`.
- Dropped `.radio-inline` and `.checkbox-inline`.
- Consolidated `.checkbox` and `.radio` into `.form-check` and the various `.form-check-*` classes.
- Horizontal forms overhauled:
 - Dropped the `.form-horizontal` class requirement.
 - `.form-group` no longer applies styles from the `.row` via mixin, so `.row` is now required for horizontal grid layouts (e.g., `<div class="form-group row">`).
 - Added new `.col-form-label` class to vertically center labels with `.form-controls`.
 - Added new `.form-row` for compact form layouts with the grid classes (swap your `.row` for a `.form-row` and go).
- Added custom forms support (for checkboxes, radios, selects, and file inputs).
- Replaced `.has-error`, `.has-warning`, and `.has-success` classes with HTML5 form validation via CSS's `:invalid` and `:valid` pseudo-classes.
- Renamed `.form-control-static` to `.form-control-plaintext`.

Buttons

- Renamed `.btn-default` to `.btn-secondary`.
- Dropped the `.btn-xs` class entirely as `.btn-sm` is proportionally much smaller than v3's.
- The [stateful button](#) feature of the `button.js` jQuery plugin has been dropped. This includes the `$(()).button(string)` and `$(()).button('reset')` methods. We advise using a tiny bit of custom JavaScript instead, which will have the benefit of behaving exactly the way you want it to.
 - Note that the other features of the plugin (button checkboxes, button radios, single-toggle buttons) have been retained in v4.
- Change buttons' `[disabled]` to `:disabled` as IE9+ supports `:disabled`. However `fieldset[disabled]` is still necessary because [native disabled fieldsets are still buggy in IE11](#).

Button group

- Rewrote component with flexbox.
- Removed `.btn-group-justified`. As a replacement you can use `<div class="btn-group d-flex" role="group"></div>` as a wrapper around elements with `.w-100`.
- Dropped the `.btn-group-xs` class entirely given removal of `.btn-xs`.
- Removed explicit spacing between button groups in button toolbars; use margin utilities now.
- Improved documentation for use with other components.

Dropdowns

- Switched from parent selectors to singular classes for all components, modifiers, etc.
- Simplified dropdown styles to no longer ship with upward or downward facing arrows attached to the dropdown menu.
- Dropdowns can be built with `<div>s` or `s` now.
- Rebuilt dropdown styles and markup to provide easy, built-in support for `<a>` and `<button>` based dropdown items.
- Renamed `.divider` to `.dropdown-divider`.
- Dropdown items now require `.dropdown-item`.
- Dropdown toggles no longer require an explicit ``; this is now provided automatically via CSS's `::after` on `.dropdown-toggle`.

Grid system

- Added a new `576px` grid breakpoint as `sm`, meaning there are now five total tiers (`xs`, `sm`, `md`, `lg`, and `xl`).
- Renamed the responsive grid modifier classes from `.col-{breakpoint}-{modifier}-{size}` to `{modifier}-{breakpoint}-{size}` for simpler grid classes.
- Dropped push and pull modifier classes for the new flexbox-powered `order` classes. For example, instead of `.col-8.push-4` and `.col-4.pull-8`, you'd use `.col-8.order-2` and `.col-`

4.order-1.

- Added flexbox utility classes for grid system and components.

List groups

- Rewrote component with flexbox.
- Replaced `a.list-group-item` with an explicit class, `.list-group-item-action`, for styling link and button versions of list group items.
- Added `.list-group-flush` class for use with cards.

Modal

- Rewrote component with flexbox.
- Given the move to flexbox, alignment of dismiss icons in the header is likely broken as we're no longer using floats. Floated content comes first, but with flexbox that's no longer the case. Update your dismiss icons to come after modal titles to fix.
- The `remote` option (which could be used to automatically load and inject external content into a modal) and the corresponding `loaded.bs.modal` event were removed. We recommend instead using client-side templating or a data binding framework, or calling [jQuery.load](#) yourself.

Navs

- Rewrote component with flexbox.
- Dropped nearly all `>` selectors for simpler styling via un-nested classes.
- Instead of HTML-specific selectors like `.nav > li > a`, we use separate classes for `.navs`, `.nav-items`, and `.nav-links`. This makes your HTML more flexible while bringing along increased extensibility.

Navbar

The navbar has been entirely rewritten in flexbox with improved support for alignment, responsiveness, and customization.

- Responsive navbar behaviors are now applied to the `.navbar` class via the **required** `.navbar-expand-{breakpoint}` where you choose where to collapse the navbar. Previously this was a Less variable modification and required recompiling.
- `.navbar-default` is now `.navbar-light`, though `.navbar-dark` remains the same. **One of these is required on each navbar.** However, these classes no longer set `background-colors`; instead they essentially only affect `color`.
- Navbars now require a background declaration of some kind. Choose from our background utilities (`.bg-*`) or set your own with the light/inverse classes above [for mad customization](#).
- Given flexbox styles, navbars can now use flexbox utilities for easy alignment options.
- `.navbar-toggle` is now `.navbar-toggler` and has different styles and inner markup (no more three ``s).
- Dropped the `.navbar-form` class entirely. It's no longer necessary; instead, just use `.form-inline` and apply margin utilities as necessary.
- Navbars no longer include `margin-bottom` or `border-radius` by default. Use utilities as necessary.
- All examples featuring navbars have been updated to include new markup.

Pagination

- Rewrote component with flexbox.
- Explicit classes (`.page-item`, `.page-link`) are now required on the descendants of `.pagination`
- Dropped the `.pager` component entirely as it was little more than customized outline buttons.

Breadcrumbs

- An explicit class, `.breadcrumb-item`, is now required on the descendants of `.breadcrumbs`

Labels and badges

- Consolidated `.label` and `.badge` to disambiguate from the `<label>` element and simplify related components.
- Added `.badge-pill` as modifier for rounded “pill” look.
- Badges are no longer floated automatically in list groups and other components. Utility classes are now required for that.
- `.badge-default` has been dropped and `.badge-secondary` added to match component modifier classes used elsewhere.

Panels, thumbnails, and wells

Dropped entirely for the new card component.

Panels

- `.panel` to `.card`, now built with flexbox.
- `.panel-default` removed and no replacement.
- `.panel-group` removed and no replacement. `.card-group` is not a replacement, it is different.
- `.panel-heading` to `.card-header`
- `.panel-title` to `.card-title`. Depending on the desired look, you may also want to use [heading elements or classes](#) (e.g. `<h3>`, `.h3`) or bold elements or classes (e.g. ``, ``, `.font-weight-bold`). Note that `.card-title`, while similarly named, produces a different look than `.panel-title`.
- `.panel-body` to `.card-body`
- `.panel-footer` to `.card-footer`
- `.panel-primary`, `.panel-success`, `.panel-info`, `.panel-warning`, and `.panel-danger` have been dropped for `.bg-`, `.text-`, and `.border` utilities generated from our `$theme-colors` Sass map.

Progress

- Replaced contextual `.progress-bar-*` classes with `.bg-*` utilities. For example, `class="progress-bar progress-bar-danger"` becomes `class="progress-bar bg-danger"`.
- Replaced `.active` for animated progress bars with `.progress-bar-animated`.

Carousel

- Overhauled the entire component to simplify design and styling. We have fewer styles for you to override, new indicators, and new icons.
- All CSS has been un-nested and renamed, ensuring each class is prefixed with `.carousel-`.
 - For carousel items, `.next`, `.prev`, `.left`, and `.right` are now `.carousel-item-next`, `.carousel-item-prev`, `.carousel-item-left`, and `.carousel-item-right`.
 - `.item` is also now `.carousel-item`.
 - For prev/next controls, `.carousel-control.right` and `.carousel-control.left` are now `.carousel-control-next` and `.carousel-control-prev`, meaning they no longer require a specific base class.
- Removed all responsive styling, deferring to utilities (e.g., showing captions on certain viewports) and custom styles as needed.
- Removed image overrides for images in carousel items, deferring to utilities.
- Tweaked the Carousel example to include the new markup and styles.

Tables

- Removed support for styled nested tables. All table styles are now inherited in v4 for simpler selectors.
- Added inverse table variant.

Utilities

- **Display, hidden, and more:**
 - Made display utilities responsive (e.g., `.d-none` and `d-{sm,md,lg,xl}-none`).

- Dropped the bulk of `.hidden-*` utilities for new [display utilities](#). For example, instead of `.hidden-sm-up`, use `.d-sm-none`. Renamed the `.hidden-print` utilities to use the display utility naming scheme. More info under the [Responsive utilities](#) section of this page.
- Added `.float-{sm,md,lg,xl}-{left,right,none}` classes for responsive floats and removed `.pull-left` and `.pull-right` since they're redundant to `.float-left` and `.float-right`.
- **Type:**
 - Added responsive variations to our text alignment classes `.text-{sm,md,lg,xl}-{left,center,right}`.
- **Alignment and spacing:**
 - Added new [responsive margin and padding utilities](#) for all sides, plus vertical and horizontal shorthands.
 - Added boatload of [flexbox utilities](#).
 - Dropped `.center-block` for the new `.mx-auto` class.
- Clearfix updated to drop support for older browser versions.

Vendor prefix mixins

Bootstrap 3's [vendor prefix](#) mixins, which were deprecated in v3.2.0, have been removed in Bootstrap 4. Since we use [Autoprefixer](#), they're no longer necessary.

Removed the following mixins: `animation`, `animation-delay`, `animation-direction`, `animation-duration`, `animation-fill-mode`, `animation-iteration-count`, `animation-name`, `animation-timing-function`, `backface-visibility`, `box-sizing`, `content-columns`, `hyphens`, `opacity`, `perspective`, `perspective-origin`, `rotate`, `rotateX`, `rotateY`, `scale`, `scaleX`, `scaleY`, `skew`, `transform-origin`, `transition-delay`, `transition-duration`, `transition-property`, `transition-timing-function`, `transition-transform`, `translate`, `translate3d`, `user-select`

Documentation

Our documentation received an upgrade across the board as well. Here's the low down:

- We're still using Jekyll, but we have plugins in the mix:
 - `bugify.rb` is used to efficiently list out the entries on our [browser bugs](#) page.
 - `example.rb` is a custom fork of the default `highlight.rb` plugin, allowing for easier example-code handling.
 - `callout.rb` is a similar custom fork of that, but designed for our special docs callouts.
 - [`jekyll-to`](#) is used to generate our table of contents.
- All docs content has been rewritten in Markdown (instead of HTML) for easier editing.
- Pages have been reorganized for simpler content and a more approachable hierarchy.
- We moved from regular CSS to SCSS to take full advantage of Bootstrap's variables, mixins, and more.

Responsive utilities

All `@screen-` variables have been removed in v4.0.0. Use the `media-breakpoint-up()`, `media-breakpoint-down()`, or `media-breakpoint-only()` Sass mixins or the `$grid-breakpoints` Sass map instead.

Our responsive utility classes have largely been removed in favor of explicit [display](#) utilities.

- The `.hidden` and `.show` classes have been removed because they conflicted with jQuery's `$(...).hide()` and `$(...).show()` methods. Instead, try toggling the `[hidden]` attribute or use inline styles like `style="display: none;"` and `style="display: block;"`.
- All `.hidden-` classes have been removed, save for the print utilities which have been renamed.
 - Removed from v3: `.hidden-xs .hidden-sm .hidden-md .hidden-lg .visible-xs-block .visible-xs-inline .visible-xs-inline-block .visible-sm-block .visible-sm-inline .visible-sm-inline-block .visible-md-block .visible-md-inline .visible-md-inline-block .visible-lg-block .visible-lg-inline .visible-lg-inline-block`
 - Removed from v4 alphas: `.hidden-xs-up .hidden-xs-down .hidden-sm-up .hidden-sm-down .hidden-md-up .hidden-md-down .hidden-lg-up .hidden-lg-down`

- Print utilities no longer start with `.hidden-` or `.visible-`, but with `.d-print-`.
 - Old names: `.visible-print-block`, `.visible-print-inline`, `.visible-print-inline-block`, `.hidden-print`
 - New classes: `.d-print-block`, `.d-print-inline`, `.d-print-inline-block`, `.d-print-none`

Rather than using explicit `.visible-*` classes, you make an element visible by simply not hiding it at that screen size. You can combine one `.d-*none` class with one `.d-*block` class to show an element only on a given interval of screen sizes (e.g. `.d-none.d-md-block.d-xl-none` shows the element only on medium and large devices).

Note that the changes to the grid breakpoints in v4 means that you'll need to go one breakpoint larger to achieve the same results. The new responsive utility classes don't attempt to accommodate less common cases where an element's visibility can't be expressed as a single contiguous range of viewport sizes; you will instead need to use custom CSS in such cases.

About

[Getting started](#)

[Layout](#)

[Content](#)

[Components](#)

[Utilities](#)

[Extend](#)

[Migration](#)

[About](#)

[Overview](#)

[Team](#)

[Brand](#)

[License](#)

[Translations](#)



Try Linode today and get a
free \$100 credit.

ads via Carbon

Team

Bootstrap is maintained by a [small team of developers](#) on GitHub. We're actively looking to grow this team and would love to hear from you if you're excited about CSS at scale, writing and maintaining vanilla JavaScript plugins, and improving build tooling processes for frontend code.

History

Originally created by a designer and a developer at Twitter, Bootstrap has become one of the most popular front-end frameworks and open source projects in the world.

Bootstrap was created at Twitter in mid-2010 by [@mdo](#) and [@fat](#). Prior to being an open-sourced framework, Bootstrap was known as *Twitter Blueprint*. A few months into development, Twitter held its [first Hack Week](#) and the project exploded as developers of all skill levels jumped in without any external guidance. It served as the style guide for internal tools development at the company for over a year before its public release, and continues to do so today.

Originally [released](#) on Friday, August 19, 2011, we've since had over [twenty releases](#), including two major rewrites with v2 and v3. With Bootstrap 2, we added responsive functionality to the entire framework as an optional stylesheet. Building on that with Bootstrap 3, we rewrote the library once more to make it responsive by default with a mobile first approach.

With Bootstrap 4, we once again rewrote the project to account for two key architectural changes: a migration to Sass and the move to CSS's flexbox. Our intention is to help in a small way to move the web development community forward by pushing for newer CSS properties, fewer dependencies, and new technologies across more modern browsers.

Get involved

Get involved with Bootstrap development by [opening an issue](#) or submitting a pull request. Read our [contributing guidelines](#) for information on how we develop.

