

BOOTSTRAP 4

“minified documentation”

Introductory Information

Document could be said to be a minified version of the official documentation found on <https://getbootstrap.com/docs/4.3/getting-started/introduction/> , i.e., it contains all the information I found to be the most essential and omits anything I saw as fluff and some JavaScripts aspects that I do not understand yet. The documentation should be used to complement this document especially when it comes to JavaScript and Sass based functionalities.

Headings in chapter 1 and chapter 2 are different than in official documentation. Other than those, headings are equivalent to ones in documentation.

99% of the text is verbatim copy from the documentation, with some small inputs by me for easier understanding.

Attempts were made to make the document as brief as possible, so as little paper and ink would be used in case of printing. Let's be eco friendly ☺ Because of that, some pictures are not presented in their entirety, i.e., they are cut off. Same goes for some code excerpts. In many cases, the document still contains information that is doubled.

Code is embedded in a top and a bottom border among other, for accessibility purposes.

Table of Contents

1. Introduction	4
a. Starter template.....	4
b. Important Globals	5
c. Common Issues	5
d. Tooling.....	9
e. Accessibility.....	9
2. Layout.....	11
a. Containers Introduction.....	11
b. Grid System	12
3. Content	19
a. Reboot.....	19
b. Typography	23
c. Code	28
d. Images.....	29
e. Tables	30
f. Figures.....	34
4. Components.....	35
a. Alerts.....	35
b. Badges.....	37
c. Breadcrumb (# pages – look up later)	38
d. Buttons.....	39
e. Button groups	42
f. Cards	43
g. Carousel (JS).....	53
h. Collapse.....	58
i. Dropdowns (JS)	61
j. Forms (JS).....	66
k. Input Group.....	82
l. Jumbotron.....	86
m. List Group (JS).....	86
n. Media Object.....	91
o. Modal	94

p.	Navs (Navigation).....	98
q.	Navbar.....	104
r.	Pagination	109
s.	Popovers (JS).....	110
t.	Progress Bars.....	111
u.	Scrollspy	114
v.	Spinners.....	115
w.	Toasts (JS).....	117
x.	Tooltips (JS poppers).....	120
5.	Utilities	122
a.	Borders.....	122
b.	Clearfix (sass)	123
c.	Close icon	124
d.	Colors	124
e.	Display Property.....	125
f.	Flex	129
g.	Float (Sass)	135
h.	Overflow.....	136
i.	Position	136
j.	Screen Readers	137
k.	Shadows	138
l.	Sizing	138
m.	Spacing	139
n.	Stretched Link	142
o.	Text.....	144
p.	Vertical Alignment	146
q.	Visibility.....	147
6.	Extend	148
a.	Approach.....	148
b.	Icons	151
7.	Migration.....	151
8.	About.....	151

BOOTSTRAP 4

DOCUMENTATION

Only the things I thought were important

1. Introduction

a. Starter template

The starter template below serves to add Bootstrap through the CDN, if you wish to add your local version, just modify the links.

```
<!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/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
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-3.3.1.slim.min.js"
integrity="sha384-
q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-
U02eT0CpHqdSJQ6hJty5KVphtPhzWj9W01c1HTMGA3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
  </body>
</html>
```

b. 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.

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">
```

For more straightforward sizing in CSS, we switch the global `box-sizing` value from `content-box` to `border-box`. Box-sizing property was introduced in CSS3. Though box-sizing has three possible values (`content-box`, `padding-box`, and `border-box`), the most popular value is `border-box`.

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. (read more about Reboot below in this document).

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.

c. Common Issues

Bootstrap should (in most cases) display and function correctly in all browsers on PC and mobile, except for proxy browsers. For a list of some of the browser bugs that Bootstrap has to grapple with, see our [Wall of browser bugs](#).

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.**

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](#).

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.

Printing

Even in some modern browsers, printing can be quirky.

One potential workaround is the following CSS:

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

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.
<https://output.jsbin.com/OyagoDO/2> (issue & solution)

Copy

```
<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>
```

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](#).

Bootstrap & JavaScript (including Sass)

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

This section was not fully completed because of current lack of knowledge and not understanding this part of the documentation. When your understanding increases, return to it!

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. 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.

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

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. CSS variables offer similar flexibility to Sass's variables, but without the need for compilation before being served to the browser.

d. Tooling

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.

Troubleshooting

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

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.

e. Accessibility

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. It should be possible to create websites and applications that [WCAG 2.0](#) (A/AA/AAA), [Section 508](#) and similar accessibility standards and requirements. 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-ARIA](#) roles and attributes, these components should also be understandable and operable using assistive technologies (such as screen readers).

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>
```

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.

2. Layout

a. Containers Introduction

Containers are the most basic layout element in Bootstrap and are **required when using our default grid system**. Choose from a responsive, fixed-width container (meaning its **max-width** changes at each breakpoint) or fluid-width (meaning it's **100%** wide all the time). While containers *can* be nested, most layouts do not require a nested container.

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

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

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

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.

```
// 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) { ... }
```

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+**. 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.

b. Grid System

Use our powerful mobile-first flexbox grid to build layouts of all shapes and sizes thanks to a twelve column system, five default responsive tiers, Sass variables and mixins, and dozens of predefined classes. 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. **New to or unfamiliar with flexbox?** [Read this CSS Tricks flexbox guide](#) for background, terminology, guidelines, and code snippets.

```
<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>
</div>
```

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](#).

Certain HTML elements, like `<summary>`, `<fieldset>` and `<button>`, do not work as flex containers. The browser's default rendering of those element's UI conflicts with the display: flex declaration. Demo [9.1.a](#) shows how `<button>` elements didn't work in Firefox, and demo [9.2.a](#) shows that `<fieldset>` elements don't work in most browsers. Demo [9.3.a](#) shows that `<summary>` elements don't work in Safari.

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
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	<i>.col-</i>	<i>.col-sm-</i>	<i>.col-md-</i>	<i>.col-lg-</i>	<i>.col-xl-</i>
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

Add any number of unit-less classes for each breakpoint you need and every column will be the same width.

Equal-width columns can be broken into multiple lines, but 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 you're up-to-date.

Column	Column
Column	Column

```
<div class="container">
  <div class="row">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="w-100"></div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Setting one column width

Auto-layout for flexbox grid columns also means you can set the width of one column and have the sibling columns automatically resize around it.

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>
```

Equal-width multi-row

Create equal-width columns that span multiple rows by inserting a `.w-100` where you want the columns to break to a new line. Make the breaks responsive by mixing the `.w-100` with some [responsive display utilities](#).

```
<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>
```

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`.

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.

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`.**

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

```

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

```

No gutters

The gutters between columns in our predefined grid classes can be removed with `.no-gutters`. This removes the negative `margin` 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`.

Copy

```

.no-gutters {
  margin-right: 0;
  margin-left: 0;

  > .col,
  > [class*="col-"] {
    padding-right: 0;
    padding-left: 0;
  }
}

```

```

<div class="row no-gutters">
  <div class="col-12 col-sm-6 col-md-8">.col-12 .col-sm-6 .col-md-8</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>

```

.col-12 .col-sm-6 .col-md-8	.col-6 .col-md-4
-----------------------------	------------------

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.

Column breaks

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

```
<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](#).

```
<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>
```

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, but unordered	Third, but first	Second, but last
----------------------	------------------	------------------

```
<div class="container">
  <div class="row">
    <div class="col">          First, but unordered
    </div>
    <div class="col order-12">      Second, but last
    </div>
    <div class="col order-1">      Third, but first
    </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.

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. **In addition to column clearing at responsive breakpoints, you may need to reset offsets. See this in action in [the grid example](#).**

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.

Nesting

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

Level 1: <code>.col-sm-9</code>	
Level 2: <code>.col-8 .col-sm-6</code>	Level 2: <code>.col-4 .col-sm-6</code>

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.

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.

3. Content

a. Reboot

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

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

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

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

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](#).

```
$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, sans-serif,
  // Emoji fonts
  "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol" !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`.

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](#).

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)`.

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

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>` are for presenting contact information for the nearest ancestor (or an entire body of work). Preserve formatting by ending lines with `
`.

Blockquote

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

Inline elements

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

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.

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 .invisibleclass](#) instead.

b. Typography

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

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>` (`#fff` by default).

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

`.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

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>
```

Lead

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

Inline text elements

Styling for common inline HTML5 elements.

```
<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.

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.

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

Blockquotes

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

```
<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>`.

```
<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>
```

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

— Someone famous in *Source Title*

Alignment

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

```
<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>
```

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.

Inline

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

Inline

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

- Lorem ipsum
- Phasellus iaculis
- Nulla volutpat

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>
```

```
<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	<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>	
Malesuada porta	Etiam porta sem malesuada magna mollis euismod.	
Truncated term i...	Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.	
Nesting	Nested definition list	Aenean posuere, tortor sed cursus feugiat, nunc augue blandit nunc.

```

<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 can be enabled by changing the `$enable-responsive-font-sizes` Sass variable to `true` and recompiling Bootstrap. To support RFS, 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 and its configuration can be found on its [GitHub repository](#).

c. Code

Inline code

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

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.

Variables

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

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 + ,`

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.

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

d. Images

Documentation and examples for opting images into responsive behaviour (so they never become larger than their parent elements) and add lightweight styles to them – all via classes.

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.

```

```

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.

```

```

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](#).

```



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

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.

e. Tables

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

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

```
<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

Table head options

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

Striped rows

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

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

```
<table class="table table-striped">
<table class="table table-striped table-dark">
<table class="table table-bordered">
<table class="table table-bordered table-dark">
<table class="table table-borderless">
<table class="table table-borderless table-dark">
<table class="table table-hover">
<table class="table table-hover table-dark">
<table class="table table-sm">
```

Bordered table

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

Borderless table

Add `.table-borderless` for a table without borders.
`.table-borderless` can also be used on dark tables.

Hoverable rows

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

Small table

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

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>
```

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

```
<!-- 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>
```

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.

List of users

#	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>
```

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.

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

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.

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

f. Figures

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

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.

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

4. Components

a. Alerts

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

Alerts are available for any length of text, as well as an optional dismiss button. For proper styling, use one of the eight **required** contextual classes (e.g., `.alert-success`). For inline dismissal, use the [alerts jQuery plugin](#).

A simple primary alert—check it out!

A simple secondary alert—check it out!

A simple success alert—check it out!

A simple danger alert—check it out!

A simple warning alert—check it out!

A simple info alert—check it out!

A simple light alert—check it out!

A simple dark alert—check it out!

```
<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>
```

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 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.

```
<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>
```

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.

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>
```

JavaScript behavior

Enable dismissal of an alert via JavaScript:

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

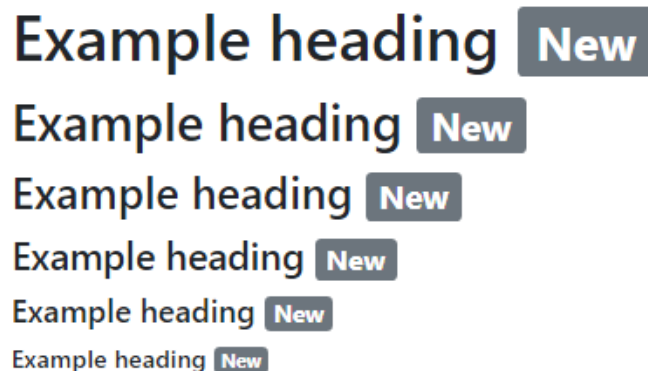
Or with `data` attributes on a button **within the alert**, as demonstrated above:

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

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.

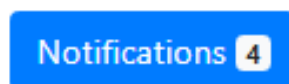
b. Badges

Documentation and examples for badges, our small count and labeling component. Badges scale to match the size of the immediate parent element by using relative font sizing and `em` units.



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

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



```
<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.

Contextual variations

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



```
<span class="badge badge-primary">Primary</span>
<span class="badge badge-secondary">Secondary</span>
```

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.

Links

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

```
<a href="#" class="badge badge-primary">Primary</a>
<a href="#" class="badge badge-secondary">Secondary</a>
```

c. Breadcrumb (# pages – look up later)

Indicate the current page’s location within a navigational hierarchy that automatically adds separators via CSS. 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.

d. Buttons

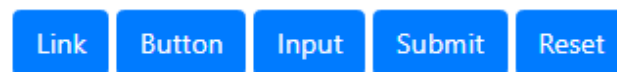
Use Bootstrap's custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more. Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control.



```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</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.



```
<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">
```

Outline buttons

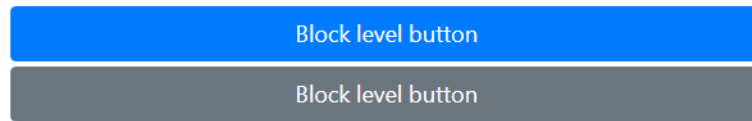
In need of a button, but not the hefty background colors they bring? Replace the default modifier classes with the `.btn-outline-*` ones to remove all background images and colors on any button.



Sizes

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

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



```
<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.

```
<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. 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.

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

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

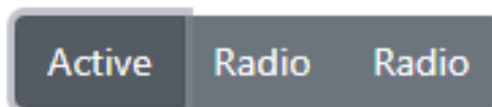
Add `data-toggle="button"` to toggle a button's **active** state. If you're pre-toggling a button, you must manually add the `.active` class **and** `aria-pressed="true"` to the `<button>`.

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

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>`.

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

e. Button groups

Group a series of buttons together on a single line with the button group, and superpower them with JavaScript. Wrap a series of buttons with `.btn` in `.btn-group`. Add on optional JavaScript radio and checkbox style behavior with [our buttons plugin](#).

```
<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>
```

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.

```
<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>
```

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.

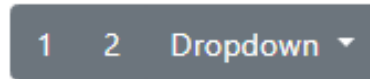
Sizing

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>
```

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>
```

Vertical variation

Make a set of buttons appear vertically stacked rather than horizontally. **Split button dropdowns are not supported here.**

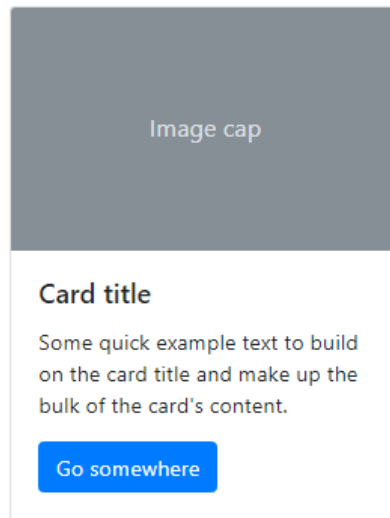
```
<div class="btn-group-vertical">
  ...
</div>
```

f. Cards

Bootstrap's cards provide a flexible and extensible content container with multiple variants and options. 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.

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](#).



```
<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>
```

Card Body

The building block of a card is the `.card-body`. Use it whenever you need a padded section within a card.

```
<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.

```
<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.

```
<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>
```

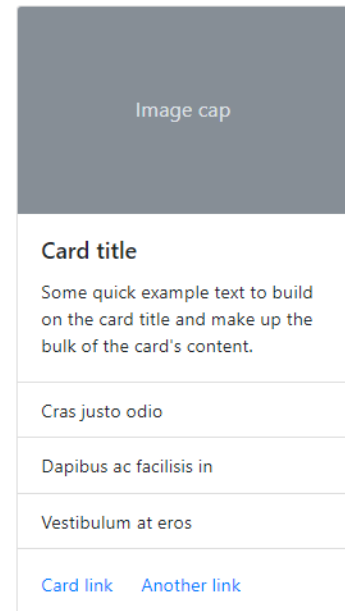
List groups

Create lists of content in a card with a flush list group.

```
<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.



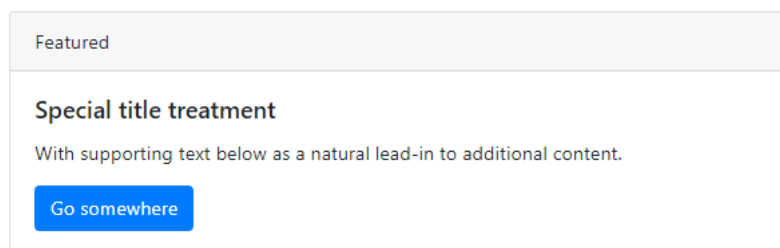
```

<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.



```

<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.

```

<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>

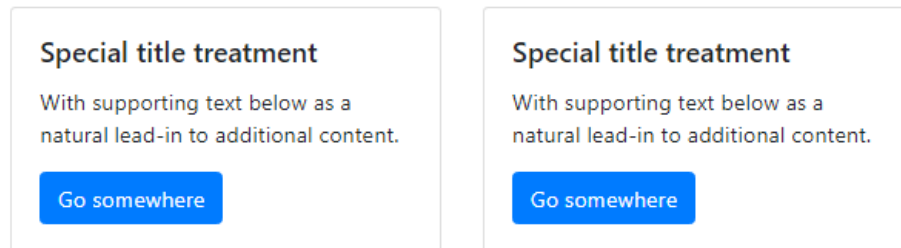
```

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.



```
<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.

```
<div class="card w-75">
<div class="card w-50">
```

Using custom CSS

Use custom CSS in your stylesheets or as inline styles to set a width.

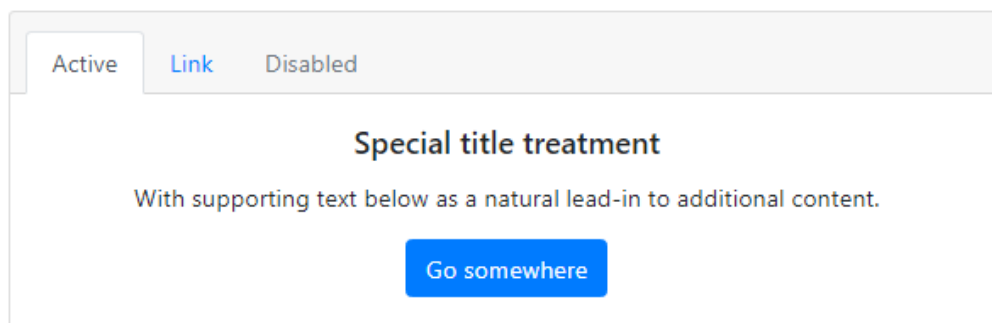
Text alignment

You can quickly change the text alignment of any card—in its entirety or specific parts—with our [text align classes](#).

```
<div class="card" style="width: 18rem;">
<div class="card text-center" style="width: 18rem;">
<div class="card text-right" style="width: 18rem;">
```

Navigation

Add some navigation to a card's header (or block) with Bootstrap's [nav components](#).



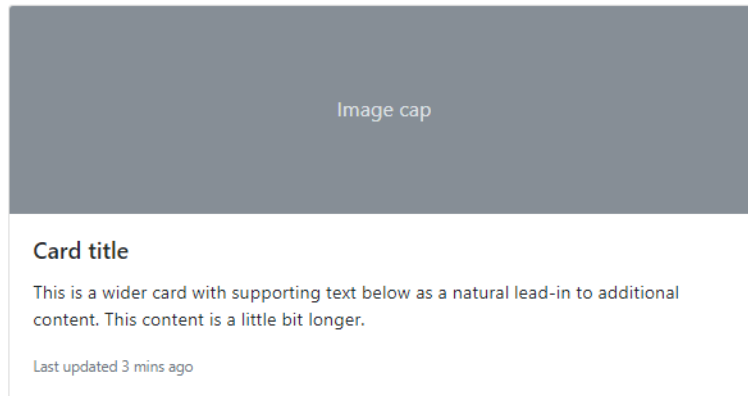
```
<div class="card text-center">
  <div class="card-header">
    <ul class="nav nav-tabs card-header-tabs">
      <li class="nav-item">
        <a class="nav-link active" href="#">Active</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li 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>
```

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

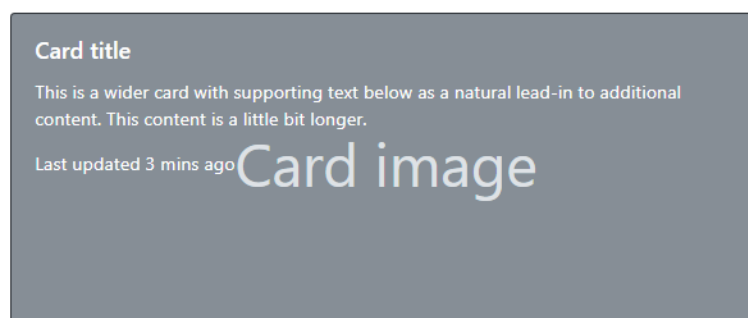
Similar to headers and footers, cards can include top and bottom “image caps”—images at the top or bottom of a card.



```
<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>
```

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.

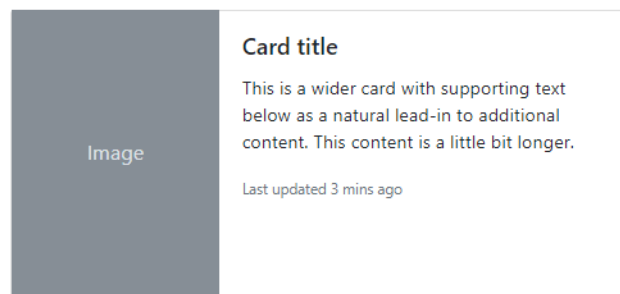


```
<div class="card bg-dark text-white">
  
  <div class="card-img-overlay">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">This is a wider card with supporting text below as a
natural lead-in to additional content. This content is a little bit longer.</p>
    <p class="card-text">Last 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>
```

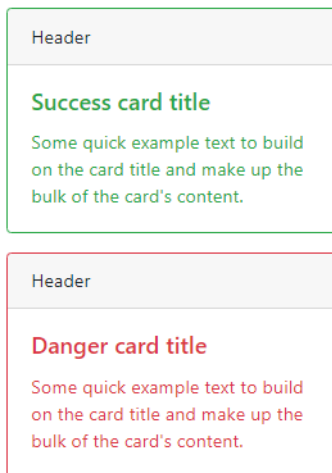
Card styles

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

```
<div class="card text-white bg-primary mb-3" style="max-width: 18rem;">
<div class="card text-white bg-secondary mb-3" style="max-width: 18rem;">
<div class="card text-white bg-success mb-3" style="max-width: 18rem;">
```

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.



```
<div class="card border-primary mb-3" style="max-width: 18rem;">
<div class="card border-secondary mb-3" style="max-width: 18rem;">
<div class="card border-success mb-3" style="max-width: 18rem;">
<div class="card border-danger mb-3" style="max-width: 18rem;">
<div class="card border-warning mb-3" style="max-width: 18rem;">
<div class="card border-info mb-3" style="max-width: 18rem;">
<div class="card border-light mb-3" style="max-width: 18rem;">
<div class="card border-dark mb-3" style="max-width: 18rem;">
```

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`.

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

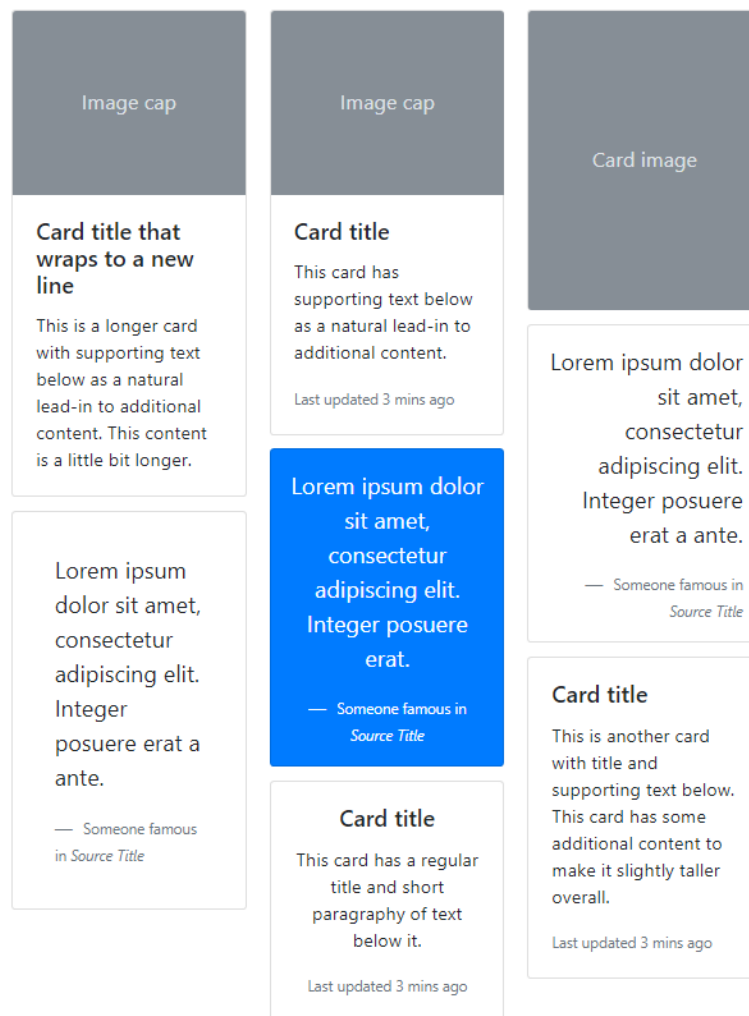
Use card groups to render cards as a single, attached element with equal width and height columns. Card groups use `display: flex` to achieve their uniform sizing.

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

When using card groups with footers, their content will automatically line up. Just like with card groups, card footers in decks will automatically line up.

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.**



```
<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>
```

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;
  }
}
```

g. Carousel (JS)

A slideshow component for cycling through elements—images or slides of text—like a carousel. 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](#).

The `prefers-reduced-motion` media feature is used to detect if the user has requested the system minimize the amount of animation or motion it uses. *no-preference*

Indicates that the user has made no preference known to the system. This keyword value evaluates as false in the [boolean context](#).

reduce

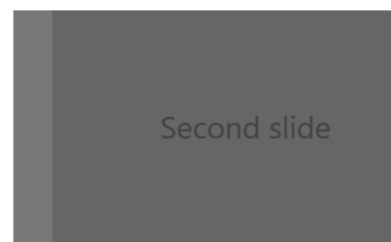
Indicates that user has notified the system that they prefer an interface that minimizes the amount of movement or animation, preferably to the point where all non-essential movement is removed.

Please be aware that nested carousels are not supported, and carousels are generally not compliant with accessibility standards. 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. (code on next page)

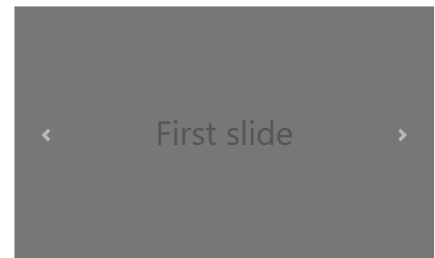


```
<div id="carouselExampleSlidesOnly" class="carousel slide" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      
    </div>
    <div class="carousel-item">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
</div>
```

With controls

Adding in the previous and next controls:

```
<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.



```

<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`.

Crossfade

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

Individual `.carousel-item` interval

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

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.**

Usage Via JavaScript

Call carousel manually with:

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

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
})
```

`.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.

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`).

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.
pause	string boolean	"hover"	<p>If set to <code>"hover"</code>, 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.</p> <p>On touch-enabled devices, when set to <code>"hover"</code>, 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.</p>
ride	string	false	Autoplays the carousel after the user manually cycles the first item. If <code>"carousel"</code> , 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.

h. Collapse

Toggle the visibility of content across your project with a few classes and our JavaScript plugins. 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](#).

Click the buttons below to show and hide another element via class changes:

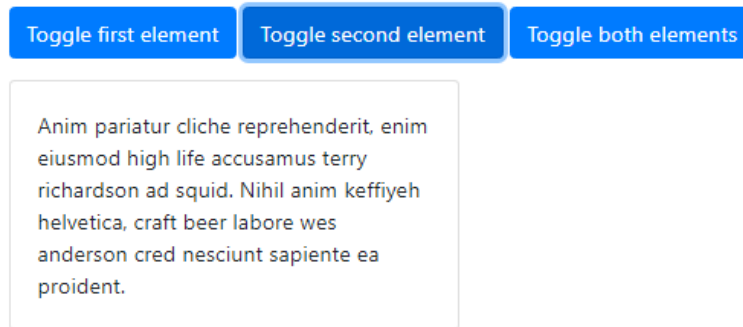
- **.collapse** hides content
- **.collapsing** is applied during transitions
- **.collapse.show** shows content

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.

```
<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



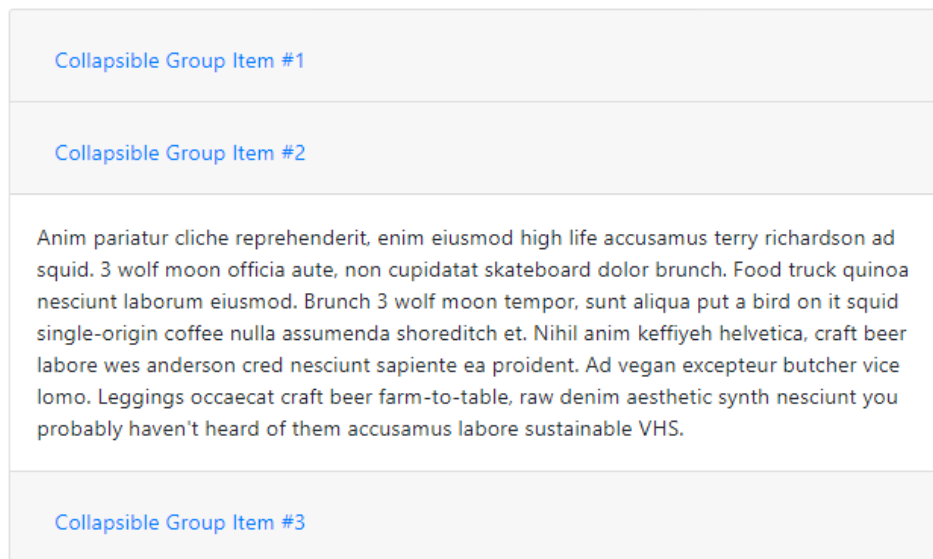
```

<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.



FIRST DIV

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

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`.

Usage 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.

Usage via data JavaScript

Via JavaScript

Enable manually with:

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

Options (more in official documentation)

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

i. Dropdowns (JS)

Toggle contextual overlays for displaying lists of links and more with the Bootstrap dropdown plugin. 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](#).

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.

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 and with `<a>` elements. The best part is you can do this with any button variant.



```
<!-- Example single danger button -->
<div class="btn-group">
  <button type="button" class="btn btn-danger dropdown-toggle" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Action
  </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>
```

Split button

Similarly, create split button dropdowns with virtually the same markup as single button dropdowns, but with the addition of `.dropdown-toggle-split` for proper spacing around the dropdown caret.

We use this extra class to reduce the horizontal `padding` on either side of the caret by 25% and remove the `margin-left` that's added for regular button dropdowns. Those extra changes keep the caret centered in the split button and provide a more appropriately sized hit area next to the main button.

```

<!-- Example split danger button -->
<div class="btn-group">
  <button type="button" class="btn btn-danger">Action</button>
  <button type="button" class="btn btn-danger dropdown-toggle dropdown-toggle-split" data-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.

Drop-up

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

Dropright

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

Dropleft

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

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

```

<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**.

Menu alignment

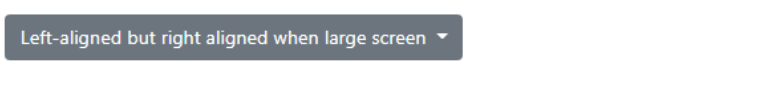
By default, a dropdown menu is automatically positioned 100% from the top and along the left side of its parent. Add `.dropdown-menu-right` to a `.dropdown-menu` to right align the dropdown menu. **Heads up!** Dropdowns are positioned thanks to Popper.js (except when they are contained in a navbar).

```
<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`.



```
<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>
```

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.

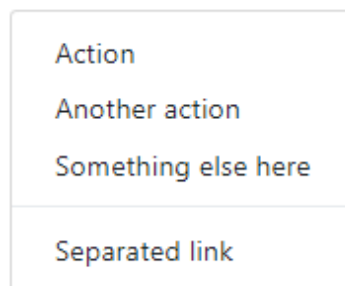
Headers

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

```
<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.



```
<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>
```

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.

```
<div class="dropdown-menu">
.....
  <div class="form-group">
    <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>
    </div>
    <button type="submit" class="btn btn-primary">Sign
in</button>
  </form>
  <div class="dropdown-divider"></div>
  <a class="dropdown-item" href="#">New around here? Sign up</a>
  <a class="dropdown-item" href="#">Forgot password?</a>
</div>
```

Dropdown options

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

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.

Usage Via data attributes

Add `data-toggle="dropdown"` to a link or button to toggle a dropdown.

Usage Via JavaScript

Call the dropdowns via JavaScript:

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

j. Forms (JS)

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

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

```

<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-
describedby="emailHelp" placeholder="Enter email">
    <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"
placeholder="Password">
  </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>`.

Email address

Example select

Example multiple select

Example textarea

```

<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

Choose file No file chosen

```

<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`.

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 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

```
<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"
placeholder="Password">
    </div>
  </div>
</form>
```

Range Inputs

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

Example Range input



```
<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 radios use are built to 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`.

```
<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>
```

Inline

Group checkboxes or radios on the same horizontal row by adding `.form-check-inline` to any `.form-check`.

```
<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>
```

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`).

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

Another label

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

<input type="text" value="First name"/>	<input type="text" value="Last name"/>
---	--

```
<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 (produces equivalent result to the above example)

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.

```
<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.

The form layout consists of the following elements:

- Email** and **Password** fields side-by-side.
- Address** field (1234 Main St).
- Address 2** field (Apartment, studio, or floor).
- City**, **State** (Choose... dropdown), and **Zip** fields side-by-side.
- ☐ **Check me out**
- Sign in** button

```
<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"
placeholder="Email">
    </div>
    <div class="form-group col-md-6">
      <label for="inputPassword4">Password</label>
      <input type="password" class="form-control" id="inputPassword4"
placeholder="Password">
    </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.

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`.

Column sizing

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

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. You can then remix that once again with size-specific column classes. And of course [custom form controls](#) are supported.

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-screen reader visitors with `.sr-only`.

☐ Remember me

Submit

```

<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>

```

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 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>

```

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>
```

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 field set 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).

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 unstyleable via CSS) or our custom feedback styles with additional HTML and CSS.
- You may provide custom validity messages with `setCustomValidity` in JavaScript.

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.

First name	Last name	Username
<input type="text" value="Mark"/>	<input type="text" value="Otto"/>	<input type="text" value="@ Username"/>
Looks good!	Looks good!	Please choose a username.
City	State	Zip
<input type="text" value="City"/>	<input type="text" value="State"/>	<input type="text" value="Zip"/>
Please provide a valid city.	Please provide a valid state.	Please provide a valid zip.
<input type="checkbox"/> Agree to terms and conditions You must agree before submitting.		
<input type="button" value="Submit form"/>		

```
<form class="needs-validation" novalidate>
  <div class="form-row">
    <div class="col-md-4 mb-3">
      <label for="validationCustom01">First name</label>
      <input type="text" class="form-control" id="validationCustom01"
placeholder="First name" value="Mark" required>
      <div class="valid-feedback">
        Looks good!      </div>    </div>
    <div class="col-md-4 mb-3">
      <label for="validationCustom02">Last name</label>
      <input type="text" class="form-control" id="validationCustom02"
placeholder="Last name" value="Otto" required>
      <div class="valid-feedback">
        Looks good!      </div>    </div>
    <div class="col-md-4 mb-3">
      <label for="validationCustomUsername">Username</label>
      <div class="input-group">
```

```

        <div class="input-group-prepend">
            <span class="input-group-text" id="inputGroupPrepend">@</span>
        </div>
        <input type="text" class="form-control" id="validationCustomUsername"
placeholder="Username" aria-describedby="inputGroupPrepend" required>
        <div class="invalid-feedback">
            Please choose a username. ,
        </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"
placeholder="City" required>
            <div class="invalid-feedback">
                Please provide a valid city.
            </div>
        </div>
        <div class="col-md-3 mb-3">
            <label for="validationCustom04">State</label>
            <input type="text" class="form-control" id="validationCustom04"
placeholder="State" required>
            <div class="invalid-feedback">
                Please provide 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"
placeholder="Zip" 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>
        <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);
        }, 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.

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.

Supported elements

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

- `<input>`s and `<textarea>`s with `.form-control` (including up to one `.form-control` in input groups)
- `<select>`s with `.form-select` or `.custom-select`
- `.form-checks`
- `.custom-checkbox`s and `.custom-radio`s
- `.custom-file`

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	Username	
<input type="text" value="Mark"/>	<input type="text" value="Otto"/>	<input type="text" value="@"/>	<input type="text" value="Username"/>
Looks good!	Looks good!	Please choose a unique and valid username.	
City	State	Zip	
<input type="text" value="City"/>	<input type="text" value="State"/>	<input type="text" value="Zip"/>	
Please provide a valid city.	Please provide a valid state.	Please provide a valid zip.	
<input type="button" value="Submit form"/>			

Customizing

Validation states can be customized via Sass with the `$form-validation-states` map. Located in our `_variables.scss` file.

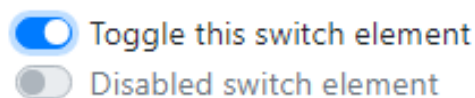
Please note that we do not recommend customizing these values without also modifying the `form-validation-state` mixin.

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.

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.



```
<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. You may also choose from small and large custom selects to match our similarly sized text inputs. The `multiple` attribute is also supported. As is the `size` attribute.

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

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.

Example range



```
<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.

```
<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"`.

```
<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.

Choose file

Browse

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

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. Here’s how one might add a Spanish translation:

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

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

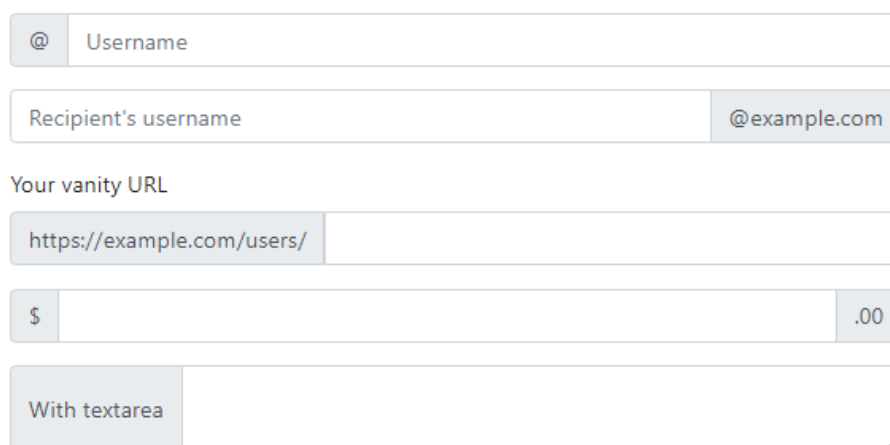
Bootstrap also provides a way to translate the "Browse" text in HTML with the `data-browse` attribute which can be added to the custom input label (example in Dutch):



```
<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>
```

k. Input Group

Easily extend form controls by adding text, buttons, or button groups on either side of textual inputs, custom selects, and custom file inputs. You can for 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>` outside the input group.



```
<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">.<span>00</span></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 `.flex-nowrap`.

```

<div class="input-group flex-nowrap">
  <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>

```

Sizing

Add the relative form sizing classes to the `.input-group` itself and contents within will automatically resize—no need for repeating the form control size classes on each element.

Sizing on the individual input group elements isn't supported.

Small

Default

Large

```
<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>
```

Checkboxes and radios

Place any checkbox or radio option within an input group's addon instead of text.

```
<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>
```

Multiple inputs

While multiple `<input>`s are supported visually, validation styles are only available for input groups with a single `<input>`.

Multiple addons

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

Button addons

Custom forms

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

```
<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>
```

Custom file input

The image displays four different configurations of a custom file input group. Each configuration consists of a text input field with the placeholder text 'Choose file' and a button to its right. The buttons are styled as light gray rectangles with rounded corners. The four examples are: 1) 'Upload' button on the left, 'Choose file' in the middle, and 'Browse' button on the right. 2) 'Choose file' on the left, 'Browse' button in the middle, and 'Upload' button on the right. 3) 'Button' button on the left, 'Choose file' in the middle, and 'Browse' button on the right. 4) 'Choose file' on the left, 'Browse' button in the middle, and 'Button' button on the right.

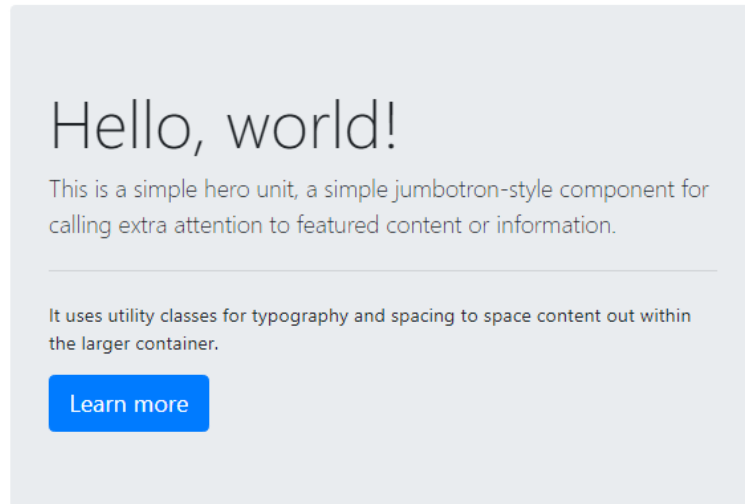
```
<div class="input-group mb-3">
  <div class="input-group-prepend">
    <span class="input-group-text" id="inputGroupFileAddon01">Upload</span>
  </div>
  <div class="custom-file">
    <input type="file" class="custom-file-input" id="inputGroupFile01" aria-
describedby="inputGroupFileAddon01">
  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

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.

I. Jumbotron

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



```
<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.

```
<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h1 class="display-4">Fluid jumbotron</h1>
```

m. List Group (JS)

List groups are a flexible and powerful component for displaying a series of content. Modify and extend them to support just about any content within. 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.

```
<ul class="list-group">
  <li class="list-group-item">Cras justo odio</li>
```

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

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

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).

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**.

```
<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>
```

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.

```
<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"
disabled>Vestibulum at eros</button>
```

Cras justo odio
Dapibus ac facilisis in
Morbi leo risus
Porta ac consectetur ac
Vestibulum at eros

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

```
<ul class="list-group list-group-flush">
  <li class="list-group-item">Cras justo odio</li>
```

Horizontal

Add `.list-group-horizontal` to change the layout of list group items from vertical to horizontal across all breakpoints. Alternatively, choose a responsive variant `.list-group-horizontal-{sm|md|lg|xl}` 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.

```
<ul class="list-group list-group-horizontal">
  <li class="list-group-item">Cras justo odio</li>
<ul class="list-group list-group-horizontal-sm">
  <li class="list-group-item">Cras justo odio</li>
```

Contextual classes

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

```
<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>
```

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

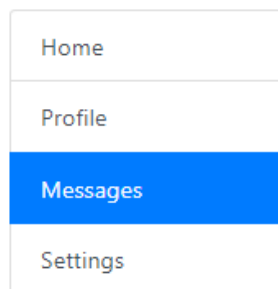
```
<li class="list-group-item d-flex justify-content-between align-items-center">
  Cras justo odio
  <span class="badge badge-primary badge-pill">14</span>
```

Custom content

Add nearly any HTML within, even for linked list groups like the one below, with the help of [flexbox utilities](#).

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.



Ut ut do pariatur aliquip aliqua aliquip exercitation do nostrud commodo reprehenderit aute ipsum voluptate. Irure Lorem et laboris nostrud amet cupidatat cupidatat anim do ut velit mollit consequat enim tempor. Consectetur est minim nostrud nostrud consectetur irure labore voluptate irure. Ipsum id Lorem sit sint voluptate est pariatur eu ad cupidatat et deserunt culpa sit eiusmod deserunt. Consectetur et fugiat anim do eiusmod aliquip nulla laborum elit adipisicing pariatur cillum.

```
<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>
```

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`.

```
COOL FEATURE WITHOUT JAVASCRIPT
<!-- 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>
```

Using Via JavaScript (more on JavaScript in documentation)

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

```
$('#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
```

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>
```

n. Media Object

Documentation and examples for Bootstrap's media object to construct highly repetitive components like blog comments, tweets, and the like. 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>
```

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.

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.

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.

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 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.

```

<div class="media">
   or
   or
  
  <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>

```

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 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.

```
<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
    </div>
  </li>
  <li class="media my-4">
    
    <div class="media-body">
      <h5 class="mt-0 mb-1">List-based media object</h5>
      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
    </div>
  </li>
</ul>
```

o. Modal

Use Bootstrap's JavaScript modal plugin to add dialogs to your site for lightboxes, user notifications, or completely custom content.

How it works

Before getting started with Bootstrap's modal component, be sure to read the following as our menu options have recently changed.

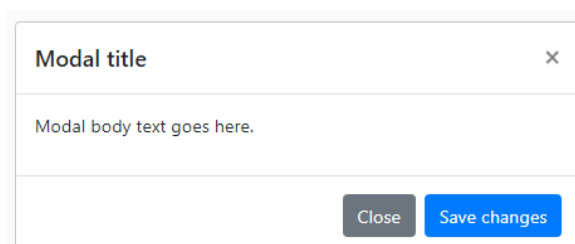
- Modals are built with HTML, CSS, and JavaScript. They're positioned over everything else in the document and remove scroll from the `<body>` so that modal content scrolls instead.
- Clicking on the modal "backdrop" will automatically close the modal.
- Bootstrap only supports one modal window at a 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')
})
```

The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the [reduced motion section of our accessibility documentation](#).

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.



```

<div class="modal" tabindex="-1" role="dialog">
  <div class="modal-dialog" role="document">
    <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>

```

<https://getbootstrap.com/docs/4.3/components/modal/> - check the example

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.

<https://getbootstrap.com/docs/4.3/components/modal/> - check the example

You can also create a scrollable modal that allows scroll the modal body by adding `.modal-dialog-scrollable` to `.modal-dialog`.

```

<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModalCenter">
  Launch demo modal
</button>

<!-- Modal -->
<div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalCenterTitle">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>
  </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>

```

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.

<https://getbootstrap.com/docs/4.3/components/modal/> - check the example

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.

<https://getbootstrap.com/docs/4.3/components/modal/> - check the example

Varying modal content

<https://getbootstrap.com/docs/4.3/components/modal/> - check the example

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`.

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.

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`, and `role="document"` to the `.modal-dialog` itself. 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	None	500px
Large	<code>.modal-lg</code>	800px
Extra large	<code>.modal-xl</code>	1140px

```
<!-- Extra large modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-
target=".bd-example-modal-xl">Extra large modal</button>

<div class="modal fade bd-example-modal-xl" tabindex="-1" role="dialog" aria-
labelledby="myExtraLargeModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-xl">
    <div class="modal-content">
      ...
    </div>
  </div>
</div>
```

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.

Usage 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
myModal</button>
```

Via JavaScript

Call a modal with id `myModal` with a single line of JavaScript:

```
$('#myModal').modal(options)
```

Options

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

p. Navs (Navigation)

Documentation and examples for how to use Bootstrap's included navigation components. 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. 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 disabled" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
  </li>
</ul>
```

Classes are used throughout, so your markup can be super flexible. Use ``s like above, `` if the order of your items is important, or roll your own with a `<nav>` element. Because the `.nav` uses `display: flex`, the nav links behave the same as nav items would, but without the extra markup.

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.

```
<ul class="nav justify-content-center">
```

Centered with `.justify-content-center`. Right-aligned with `.justify-content-end`.

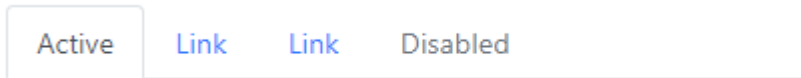
Vertical

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`). As always, vertical navigation is possible without ``s, too.

```
<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](#).



```
<ul class="nav nav-tabs">
```

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. When using a `<nav>`-based navigation, be sure to include `.nav-item` on the anchors. 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.

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.

```
<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 disabled" href="#" tabindex="-1"
  aria-disabled="true">Disabled</a>
</nav>
```

Regarding accessibility

If you're using navs to provide a navigation bar, be sure to add a `role="navigation"` to the most logical parent container of the ``, 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.

Using dropdowns

Add dropdown menus with a little extra HTML and the [dropdowns JavaScript plugin](#).

```
<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>
</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](#).

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. The tabs plugin also works with pills.

Vertical Pills

<div style="margin-bottom: 5px;">Home</div> <div style="background-color: #007bff; color: white; padding: 5px 10px; margin-bottom: 5px;">Profile</div> <div style="margin-bottom: 5px;">Messages</div> <div style="margin-bottom: 5px;">Settings</div>	<p>Culpa dolor voluptate do laboris laboris irure reprehenderit id incididunt dui pariatu mollit aute magna pariatu consectetur. Eu veniam dui non ut dolor deserunt commodo et minim in quis laboris ipsum velit id veniam. Quis ut consectetur adipisicing officia excepteur non sit. Ut et elit aliquip labore Lorem enim eu. Ullamco mollit occaecat dolore ipsum id officia mollit qui esse anim eiusmod do sint minim consectetur qui.</p>
--	--

```

<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`.

```

<!-- Nav tabs -->
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item">
    <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">
    <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">
    <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">
    <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')
})
```

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
```

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.

More on JS integration in documentation.

q. Navbar

Documentation and examples for Bootstrap's powerful, responsive navigation header, the navbar. Includes support for branding, navigation, and more, including support for our collapse plugin.

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.

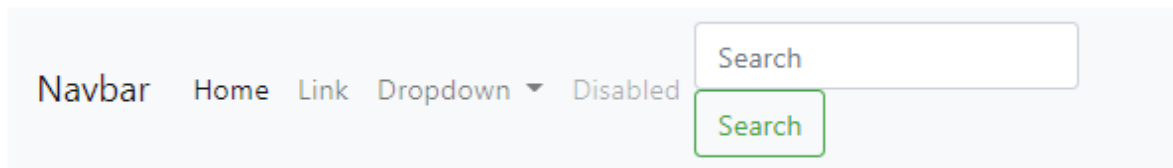
The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the [reduced motion section of our accessibility documentation](#).

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.



```
<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. Adding images to the `.navbar-brand` will likely always require custom styles or utilities to properly size.

```
<!-- Just an image -->
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">
    
  </a>
</nav>
```

Nav

Navbar navigation links build on our `.nav` options with their own modifier class and require the use of [toggler classes](#) for proper responsive styling. **Navigation in navbars will also grow to occupy as much horizontal space as possible** to keep your navbar contents securely aligned. Active states—with `.active`—to indicate the current page can be applied directly to `.nav-links` or their immediate parent `.nav-item`. 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.

Forms

Place various form controls and components within a navbar with `.form-inline`.

```
<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: between`. Use additional [flex utilities](#) as needed to adjust this behavior. Input groups work, too. 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.

Text

Navbars may contain bits of text with the help of `.navbar-text`. This class adjusts vertical alignment and horizontal spacing for strings of text.

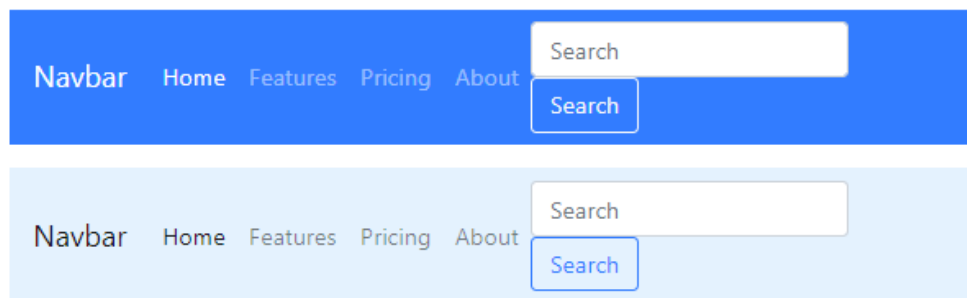
Navbar text with an inline element

```
<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.

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-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](#).

Navbar

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](#).

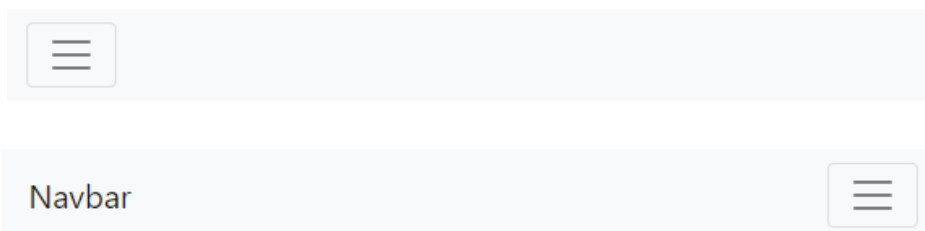
```
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">Default</a>
</nav>
<nav class="navbar fixed-top navbar-light bg-light">
  <a class="navbar-brand" href="#">Fixed top</a>
</nav>
<nav class="navbar fixed-bottom navbar-light bg-light">
  <a class="navbar-brand" href="#">Fixed bottom</a>
</nav>
<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>

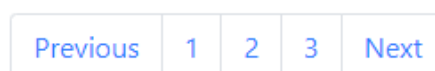
```

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!

r. Pagination

Documentation and examples for showing pagination to indicate a series of related content exists across multiple pages. 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"`.



```
<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>
```

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.

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. 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.

Sizing

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

Alignment

Change the alignment of pagination components with [flexbox utilities](#).

s. Popovers (JS)

Documentation and examples for adding Bootstrap popovers, like those found in iOS, to any element on your site.

Things to know when using the popover plugin:

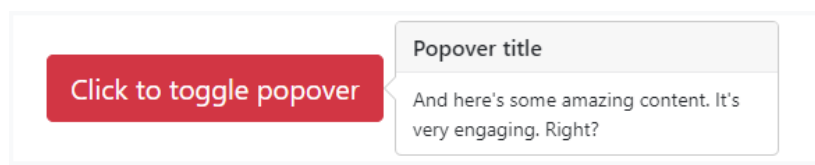
- Popovers rely on the 3rd party library [Popper.js](#) for positioning. You must include [popper.min.js](#) before bootstrap.js or use `bootstrap.bundle.min.js` / `bootstrap.bundle.js` which contains Popper.js in order for popovers to work!
- Popovers require the [tooltip plugin](#) as a dependency.
- 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.

Example: Enable popovers everywhere

One way to initialize all popovers on a page would be to select them by their `data-toggle` attribute:

```
$(function () {
  $('[data-toggle="popover"]').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>
```

More on popovers in the documentation, as they are JS reliant and my knowledge of JS is insufficient to select the appropriate content to include in this section. Might be updated in the future.

t. Progress Bars

Documentation and examples for using Bootstrap custom progress bars featuring support for stacked bars, animated backgrounds, and text labels.

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>
```

Bootstrap provides a handful of [utilities for setting width](#). Depending on your needs, these may help with quickly configuring progress.

```
<div class="progress">
  <div class="progress-bar w-75" role="progressbar" aria-valuenow="75" aria-
valuemin="0" aria-valuemax="100"></div>
</div>
```

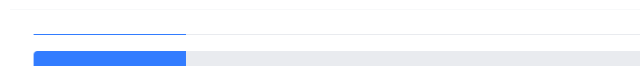
Labels

Add labels to your progress bars by placing text within the `.progress-bar`.

```
<div class="progress-bar" role="progressbar" style="width: 25%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="100">25</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.



```
<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.

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.



```
<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>
```

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.

```
<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>
```

u. Scrollspy

Automatically update Bootstrap navigation or list group components based on scroll position to indicate which link is currently active in the viewport.

Scrollspy has a few requirements to function properly:

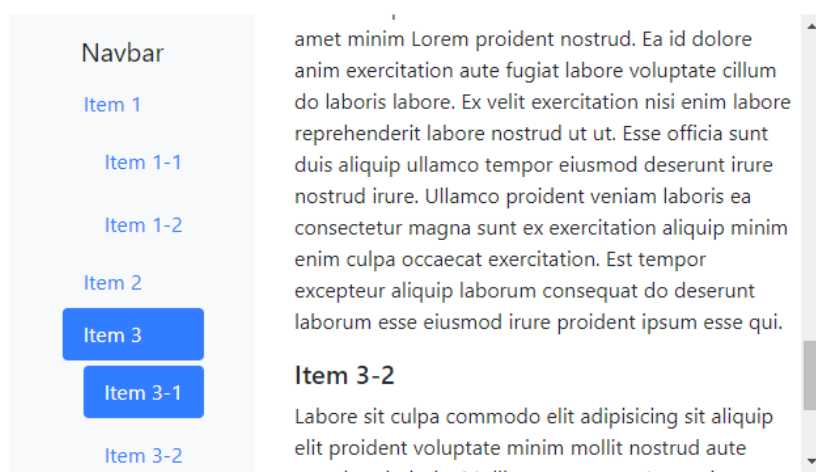
- If you're building our JavaScript from source, it [requires util.js](#).
- It must be used on a Bootstrap [nav component](#) or [list group](#).
- Scrollspy requires `position: relative;` on the element you're spying on, usually the `<body>`.
- When spying on elements other than the `<body>`, be sure 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:

<https://getbootstrap.com/docs/4.3/components/scrollspy/#example-in-navbar>

Scrollspy also works with nested `.navs`. 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. Scrollspy also works with `.list-group`. Scroll the area next to the list group and watch the active class change.



1 Nested . nav example

```

<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>

```

Usage Via JavaScript

After adding `position: relative;` in your CSS, call the scrollspy via JavaScript:

```

$('body').scrollspy({ target: '#navbar-example' })

```

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

v. Spinners

Indicate the loading state of a component or page with Bootstrap spinners, built entirely with HTML, CSS, and no JavaScript. 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

<https://getbootstrap.com/docs/4.3/components/spinners/#border-spinner>

Use the border spinners for a lightweight loading indicator. 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.



```
<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>
```

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! 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.



Alignment

Spinners in Bootstrap are built with `rems`, `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>
```

Placement

Use [flexbox utilities](#), [float utilities](#), or [text alignment](#) utilities to place spinners exactly where you need them in any situation.

```
<div class="d-flex justify-content-center">
<div class="d-flex align-items-center">
<div class="clearfix">
  <div class="spinner-border float-right" role="status">
```

Size

Add `.spinner-border-sm` and `.spinner-grow-sm` to make a smaller spinner that can quickly be used within other components. Or, use custom CSS or inline styles to change the dimensions as needed.

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.



```
<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>
```

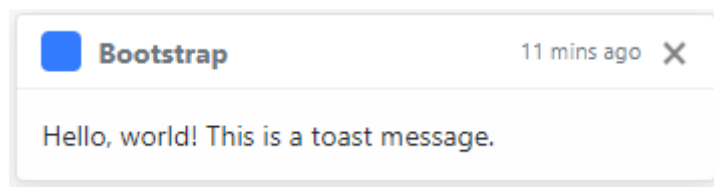
w. Toasts (JS)

Push notifications to your visitors with a toast, a lightweight and easily customizable alert message. 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.

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`.

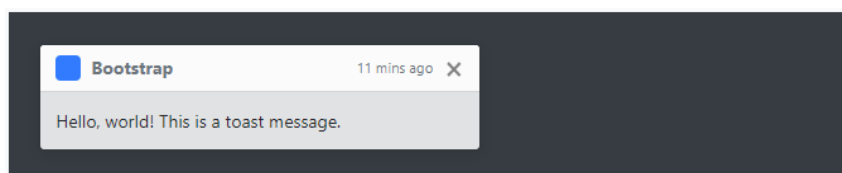
To encourage extensible and predictable toasts, we recommend a header and body. Toast headers use `display: flex`, allowing easy alignment of content thanks to our margin and flexbox utilities. Toasts are as flexible as you need and have very little required markup. At a minimum, we require a single element to contain your "toasted" content and strongly encourage a dismiss button.



```
<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>

```

Stacking

When you have multiple toasts, we default to vertically stacking them in a readable manner.

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`. For systems that generate more notifications, consider using a wrapping element so they can easily stack. You can also get fancy with flexbox utilities to align toasts horizontally and/or vertically.

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 [delaytimeout](#) to ensure people have enough time to read the toast.

When using `autohide: false`, you must add a close button to allow users to dismiss the toast.

Initialize toasts via JavaScript

```
$('.toast').toast(option)
```

More about the JS part in documentation.

x. Tooltips (JS poppers)

Documentation and examples for adding custom Bootstrap tooltips with CSS and JavaScript using CSS3 for animations and data-attributes for local title storage.

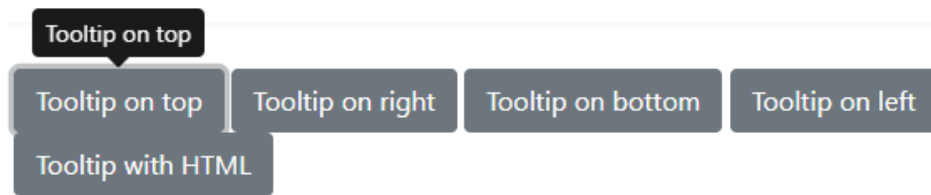
Things to know when using the tooltip plugin:

- 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.

One way to initialize all tooltips on a page would be to select them by their `data-toggle` attribute:

```
$(function () {
  $('[data-toggle="tooltip"]').tooltip()
})
```

More about the JS part in documentation.



```

<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>

```

And with custom HTML added:

```

<button type="button" class="btn btn-secondary" data-toggle="tooltip" data-
html="true" title="<em>Tooltip</em> <u>with</u> <b>HTML</b>">
  Tooltip with HTML
</button>

```

Trigger the tooltip via JavaScript:

```

$('#example').tooltip(options)

```

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' })

```

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).

More on Accessibility in the Documentation

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.

Options (much more in documentation)

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

5. Utilities

a. Borders

Use border utilities to quickly style the border and border-radius of an element. Great for images, buttons, or any other element. Use border utilities to add or remove an element's borders. Choose from all borders or one at a time.

Additive (example <https://getbootstrap.com/docs/4.3/utilities/borders/#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>
```

Subtractive (example <https://getbootstrap.com/docs/4.3/utilities/borders/#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>
```

Border color

Change the border color using utilities built on our theme colors.

```
<span class="border border-primary"></span>
```

Border-radius

```








```

Sizes (example <https://getbootstrap.com/docs/4.3/utilities/borders/#sizes>)

Use `.rounded-lg` or `.rounded-sm` for larger or smaller border-radius.

```


```

b. Clearfix (sass)

Quickly and easily clear floated content within a container by adding a clearfix utility. Easily clear `float` by adding `.clearfix` **to the parent element**. Can also be used as a mixin.

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

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>
```

c. Close icon

Use a generic close icon for dismissing content like modals and alerts.

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>
```

d. Colors

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

`.text-primary`

`.text-light`

`.text-secondary`

`.text-dark`

`.text-success`

`.text-body`

`.text-danger`

`.text-muted`

`.text-warning`

`.text-white`

`.text-info`

`.text-black-50`

`.text-white-50`

```
<p class="text-primary">.text-primary</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

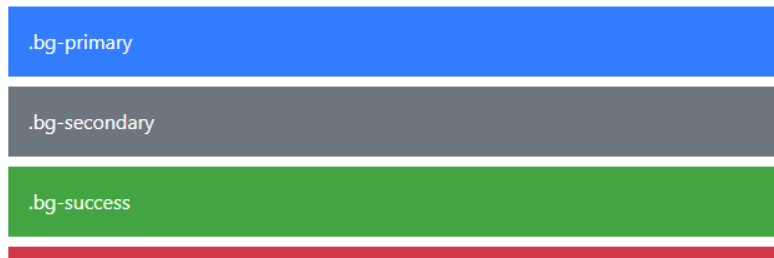
Success link

Danger link

```
<p><a href="#" class="text-primary">Primary 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.



```
<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>
```

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.

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.

e. Display Property

Quickly and responsively toggle the display value of components and more with our display utilities. Includes support for some of the more common values, as well as some extras for controlling display when printing. 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 `x1`, 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.

d-inline

d-inline

d-block

d-block

```
<div class="d-inline p-2 bg-primary text-white">d-inline</div>
<span class="d-block p-2 bg-primary text-white">d-block</span>
```

Hiding elements

For faster mobile-friendly development, use responsive display classes for showing and hiding elements by device. Avoid creating entirely different versions of the same site, instead hide elements responsively for each screen size. To hide elements simply use the `.d-none` class or one of the `.d-{sm,md,lg,xl}-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>

```
<div class="d-lg-none">hide on screens wider than lg</div>
<div class="d-none d-lg-block">hide on screens smaller than lg</div>
```

Display in print

Change the `display` value of elements when printing with our print display utility classes. Includes support for the same `display` values as our responsive `.d-*` utilities.

- `.d-print-none`
- `.d-print-inline`
- `.d-print-inline-block`
- `.d-print-block`
- `.d-print-table`
- `.d-print-table-row`
- `.d-print-table-cell`
- `.d-print-flex`
- `.d-print-inline-flex`

The print and display classes can be combined.

```
<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>
```

Embeds

Create responsive video or slideshow embeds based on the width of the parent by creating an intrinsic ratio that scales on any device. 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.



```
<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>
```

Aspect ratios

Aspect ratios can be customized with modifier classes. By default the following ratio classes are provided:

```
<!-- 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;

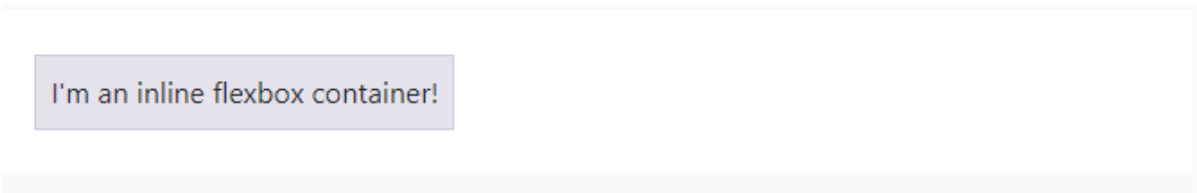
```

f. Flex

Quickly manage the layout, alignment, and sizing of grid columns, navigation, components, and more with a full suite of responsive flexbox utilities. For more complex implementations, custom CSS may be necessary. 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.



```
<div class="d-flex p-2 bd-highlight">I'm a flexbox container!</div>
```



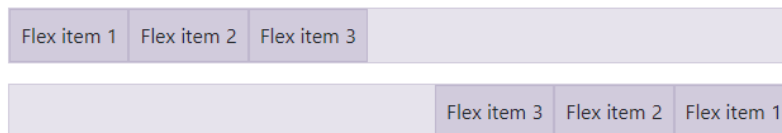
```
<div class="d-inline-flex p-2 bd-highlight">I'm an inline flexbox container!</div>
```

Responsive variations also exist for `.d-flex` and `.d-inline-flex`.

- `.d-flex`
- `.d-inline-flex`
- `.d-sm-flex`
- `.d-sm-inline-flex`
- `.d-md-flex`
- `.d-md-inline-flex`
- `.d-lg-flex`
- `.d-lg-inline-flex`
- `.d-xl-flex`
- `.d-xl-inline-flex`

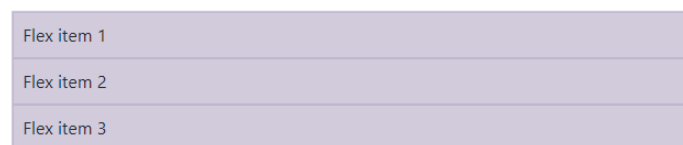
Direction

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="d-flex flex-row-reverse bd-highlight">
```

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>
```

Responsive variations also exist for `flex-direction`, e.g.:

- `.flex-sm-row`
- `.flex-sm-row-reverse`
- `.flex-md-column`
- `.flex-md-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`.



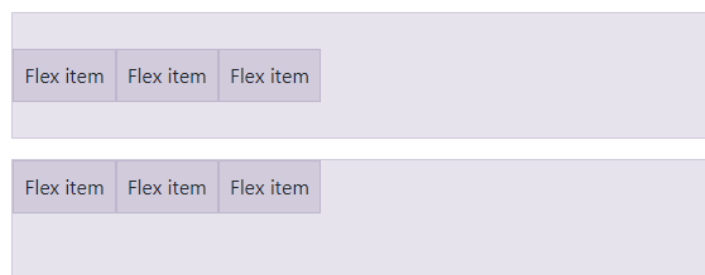
```
<div class="d-flex justify-content-start">...</div>
<div class="d-flex justify-content-end">...</div>
<div class="d-flex justify-content-center">...</div>
<div class="d-flex justify-content-between">...</div>
<div class="d-flex justify-content-around">...</div>
```

Responsive variations also exist for `justify-content`, e.g.:

- `.justify-content-sm-start`
- `.justify-content-sm-end`
- `.justify-content-md-end`
- `.justify-content-md-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-center">...</div>
<div class="d-flex align-items-baseline">...</div>
```

Responsive variations also exist for `align-items`, e.g.:

- `.align-items-baseline`
- `.align-items-stretch`
- `.align-items-lg-end`
- `.align-items-lg-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).



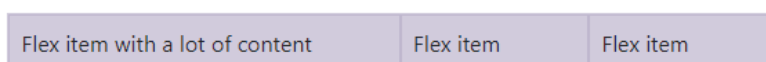
```
<div class="align-self-start">Aligned flex item</div>
<div class="align-self-end">Aligned flex item</div>
<div class="align-self-center">Aligned flex item</div>
```

Responsive variations also exist for `align-self`, e.g.:

- `.align-self-sm-baseline`
- `.align-self-sm-stretch`
- `.align-self-md-start`
- `.align-self-md-end`

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>
```

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.

Flex item	Flex item	Third flex item
-----------	-----------	-----------------

```
<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>
```

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`.

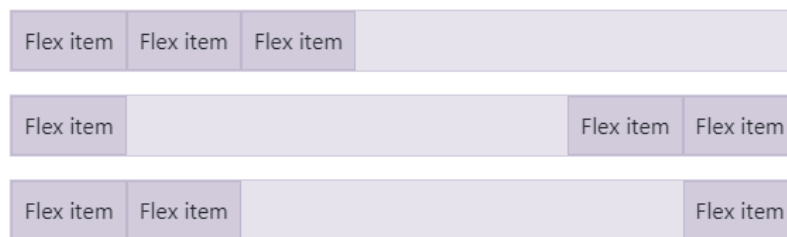
```
<div class="d-flex bd-highlight">
  <div class="p-2 w-100 bd-highlight">Flex item</div>
  <div class="p-2 flex-shrink-1 bd-highlight">Flex item</div>
</div>
```

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`

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 [StackOverflow answer](https://stackoverflow.com/a/37535548) (<https://stackoverflow.com/a/37535548>) for more details.



```
<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>
```

g. Float (Sass)

Toggle floats on any element, across any breakpoint, using our responsive float utilities. 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 affect on flex items.

Toggle a Float with a Class

Float left on all viewport sizes

Float right on all viewport sizes

Don't float 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>
```

Responsive

Responsive variations also exist for each **float** value.

```
<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>
```

- **.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**

Image replacement

Swap text for background images with the image replacement class. 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.

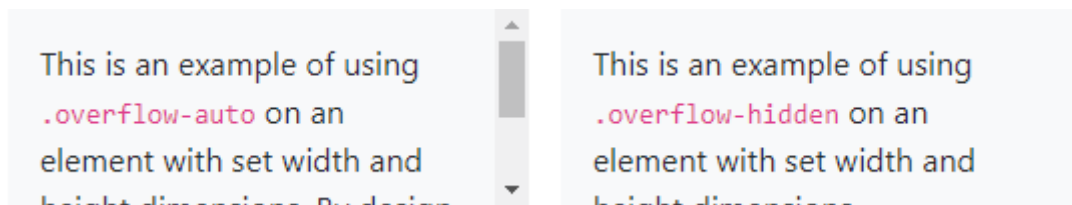
```
<h1 class="text-hide">Custom heading</h1>
```

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>
```

h. Overflow

Use these shorthand utilities for quickly configuring how content overflows an element. Barebones `overflow` functionality is provided for two values by default, and they are not responsive.



```
<div class="overflow-auto">...</div>
<div class="overflow-hidden">...</div>
```

Using Sass variables, you may customize the overflow utilities by changing the `$overflows` variable in `_variables.scss`.

i. Position

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>
```

Fixed top

Position an element at the top of the viewport, from edge to edge. Be sure you understand the ramifications of fixed position in your project; you may need to add additional CSS.

```
<div class="fixed-top">...</div>
```

Fixed bottom

Position an element at the bottom of the viewport, from edge to edge. Be sure you understand the ramifications of fixed position in your project; you may need to add additional CSS.

```
<div class="fixed-bottom">...</div>
```

Sticky top

Position an element at the top of the viewport, from edge to edge, but only after you scroll past it. 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>
```

j. Screen Readers

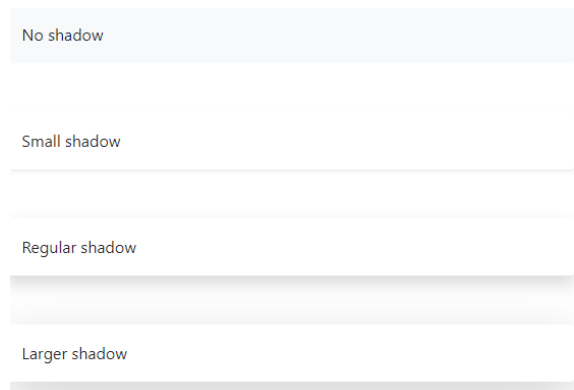
Use screen reader utilities to hide elements on all devices except screen readers.

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>
// Usage as a mixin
.skip-navigation {
  @include sr-only;
  @include sr-only-focusable;
}
```

k. Shadows

Add or remove shadows to elements with box-shadow utilities. 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).



```
<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>
```

l. Sizing

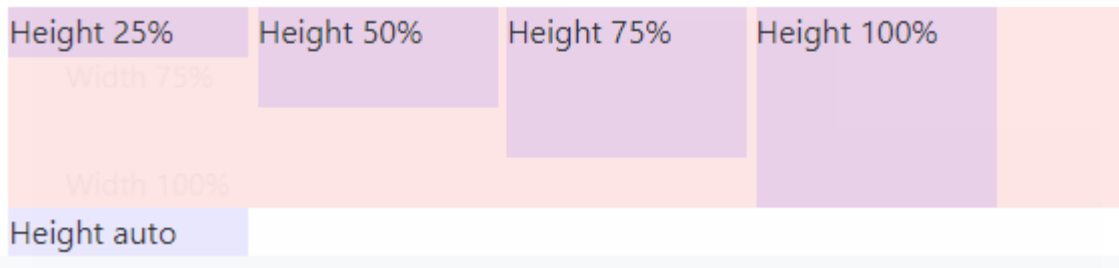
Easily make an element as wide or as tall with our width and height utilities.

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.



```
<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-auto p-3" style="background-color: #eee;">Width auto</div>
```



```
<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-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.

```

```

```
<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>
```

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>
```

m. Spacing

Bootstrap includes a wide range of shorthand responsive margin and padding utility classes to modify an element's appearance. 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`

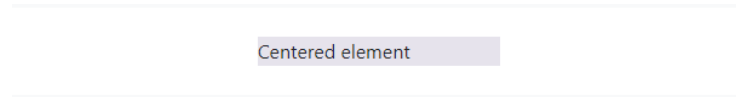
(You can add more sizes by adding entries to the `$spacers` Sass map variable.)

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;
}
```

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`.



```
<div class="mx-auto" style="width: 200px;">
  Centered element
</div>
```

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;
}
```

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`.

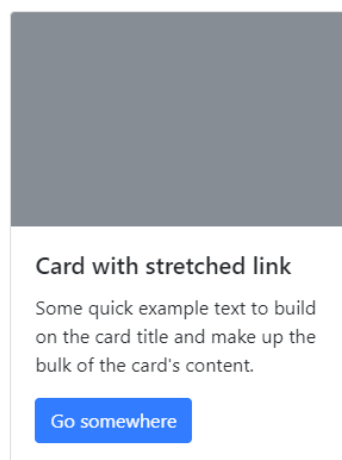


```
<div class="row mx-md-n5">
  <div class="col px-md-5"><div class="p-3 border bg-light">Custom column
padding</div></div>
  <div class="col px-md-5"><div class="p-3 border bg-light">Custom column
padding</div></div>
</div>
```

n. Stretched Link

Make any HTML element or Bootstrap component clickable by “stretching” a nested link via CSS.

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.



2 Basically means that wherever on a card you hover, the link will be activated.

```
<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>
```

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.



```
<div class="media position-relative">
  
```

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.

```
<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.

```
<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>
```

o. Text

Documentation and examples for common text utilities to control alignment, wrapping, weight, and more. Easily realign text to components with text alignment classes.

```
<p class="text-justify">...</p>
```

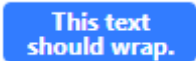
For left, right, and center alignment, responsive classes are available that use the same viewport width breakpoints as the grid system.

```
<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>

<p class="text-sm-left">Left aligned text on viewports sized SM (small) or wider.</p>
<p class="text-md-left">Left aligned text on viewports sized MD (medium) or wider.</p>
<p class="text-lg-left">Left aligned text on viewports sized LG (large) or wider.</p>
<p class="text-xl-left">Left aligned text on viewports sized XL (extra-large) or wider.</p>
```

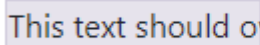
Text wrapping and overflow

Wrap text with a `.text-wrap` class.



```
<div class="badge badge-primary text-wrap" style="width: 6rem;">
  This text should wrap.
</div>
```

Prevent text from wrapping with a `.text-nowrap` class.



```
<div class="text-nowrap bd-highlight" style="width: 8rem;">
  This text should overflow the parent.
</div>
```

Monospace

Change a selection to our monospace font stack with `.text-monospace`.

```
<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.

```
<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.

```
<a href="#" class="text-decoration-none">Non-underlined link</a>
```

p. Vertical Alignment

Easily change the vertical alignment of inline, inline-block, inline-table, and table cell elements. Change the alignment of elements with the `vertical-alignment` utilities. Please note that vertical-align only affects inline, inline-block, inline-table, and table cell elements. Choose from `.align-baseline`, `.align-top`, `.align-middle`, `.align-bottom`, `.align-text-bottom`, and `.align-text-top` as needed.

With inline elements:

```
<span class="align-baseline">baseline</span>
<span class="align-top">top</span>
<span class="align-middle">middle</span>
<span class="align-bottom">bottom</span>
<span class="align-text-top">text-top</span>
<span class="align-text-bottom">text-bottom</span>
```

With table cells:

```
<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>
```

q. Visibility

Control the visibility, without modifying the display, of elements with visibility utilities.

Set the **visibility** of elements with our visibility utilities. These utility classes do not modify the **display** value at all and do not affect layout – **.invisible** elements still take up space in the page. 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>
```

```
// 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);
}
```

6. Extend

a. Approach

Learn about the guiding principles, strategies, and techniques used to build and maintain Bootstrap so you can more easily customize and extend it yourself.

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](https://github.com/twbs/bootstrap/issues/new)—we'd love to discuss it with you.

If you're reading this in printed form, here's the link:

<https://github.com/twbs/bootstrap/issues/new>

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 is random 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](https://getbootstrap.com/docs/4.3/layout/overview/#z-index) (<https://getbootstrap.com/docs/4.3/layout/overview/#z-index>).

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 is `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>`. 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.

b. Icons

Guidance and suggestions for using external icon libraries with Bootstrap. 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](https://fontawesome.com/) (<https://fontawesome.com/>)
- [Iconic](https://github.com/iconic/open-iconic) (<https://github.com/iconic/open-iconic>)
- [Octicons](https://octicons.github.com/) (<https://octicons.github.com/>)

More options

While we haven't tried these out, they do look promising and provide multiple formats—including SVG.

- [Bytesize](https://github.com/danklammer/bytesize-icons) (<https://github.com/danklammer/bytesize-icons>)
- [Google Material icons](https://material.io/tools/icons/) (<https://material.io/tools/icons/>)
- [Ionicons](https://ionicons.com/) (<https://ionicons.com/>)
- [Feather](https://feathericons.com/) (<https://feathericons.com/>)
- [Dripicons](http://demo.amitjakhu.com/dripicons/) (<http://demo.amitjakhu.com/dripicons/>)
- [Ikons](http://ikons.piotrkwiatkowski.co.uk/) (<http://ikons.piotrkwiatkowski.co.uk/>)
- [Glyph](https://glyph.smarticons.co/) (<https://glyph.smarticons.co/>)
- [Icons8](https://icons8.com/) (<https://icons8.com/>)

7. Migration

Bootstrap 4 is a major rewrite of the entire project. The changes can be found on the Bootstrap Documentation webpage:

<https://getbootstrap.com/docs/4.3/migration/>

8. About

Learn more about the team maintaining Bootstrap, how and why the project started, and how to get involved. You can also find information related to Bootstrap licence here.

<https://getbootstrap.com/docs/4.3/about/overview/>

