2023

[BOOTSTRAP 5 CHEAT SHEET

Bootstrap v5
Quickly design and customize responsive mobile-first sites with the world's most popular front-end open source toolkit.

MUKIBUL MINHAZ
MINHAZ SOLUTIONS
8/27/2023

Table of Contents

Table of Contents

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 use the browser's default root font-size (typically 16px) so visitors can customize their browser defaults as needed.
- Use the $font-family-base, $font-size-base, and $line-height-base attributes as our typographic base applied to the <body>.
- Set the global link color via $link-color.
- Use $body-bg to set a background-color on the <body> (#fff by default).

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

Headings

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

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

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

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

h1. Bootstrap heading

h2. Bootstrap heading

h3. Bootstrap heading

h4. Bootstrap heading

h5. Bootstrap heading

h6. Bootstrap heading

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

Customizing headings

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

Fancy display heading With faded secondary text

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

Display headings

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

Display 1
Display 2
Display 3
Display 4

Display 5
Display 6

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

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

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

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

Lead

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

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

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

Inline text elements

Styling for common inline HTML5 elements.

You can use the mark tag to highlight text.

T~~his line of text is meant to be treated as deleted text.~~

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

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

This line of text will render as underlined.

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

This line rendered as bold text.

This line rendered as italicized text.

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

Beware that those tags should be used for semantic purpose:

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

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

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

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

Text utilities

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

Abbreviations

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

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

attr

HTML

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

Blockquotes

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

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

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

Naming a source

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

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

Someone famous in *Source Title*

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

Alignment

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

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

Someone famous in *Source Title*

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

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

Someone famous in *Source Title*

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

Lists

Unstyled

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

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

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

Inline

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

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

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

Description list alignment

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

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

And some more placeholder definition text.

Another term

This definition is short, so no extra paragraphs or anything.

Truncated term is truncated

This can be useful when space is tight. Adds an ellipsis at the end.

Nesting

Nested definition list

I heard you like definition lists. Let me put a definition list inside your definition list.

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

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

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

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

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

Responsive font sizes

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

Sass

Variables

Headings have some dedicated variables for sizing and spacing.

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

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

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

$small-font-size:             .875em;

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

$text-muted:                  $gray-600;

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

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

$hr-margin-y:                 $spacer;
$hr-color:                    inherit;
$hr-height:                   $border-width;
$hr-opacity:                  .25;

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

$mark-padding:                .2em;

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

$nested-kbd-font-weight:      $font-weight-bold;

$list-inline-padding:         .5rem;

$mark-bg:                     #fcf8e3;
```

Mixins

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

Images

Documentation and examples for opting images into responsive behavior (so they never become 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. This applies max-width: 100%; and height: auto; to the image so that it scales with the parent element.

Responsive image

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

Image thumbnails

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

200x200

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

Aligning images

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

200x200200x200

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

200x200

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

200x200

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

Picture

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

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

```
</picture>
```

Sass

Variables

Variables are available for image thumbnails.

```
$thumbnail-padding:                 .25rem;
$thumbnail-bg:                      $body-bg;
$thumbnail-border-width:            $border-width;
$thumbnail-border-color:            $gray-300;
$thumbnail-border-radius:           $border-radius;
$thumbnail-box-shadow:              $box-shadow-sm;
```

Tables

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

Overview

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

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

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

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

Variants

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

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

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

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

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

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (e.g. the visible text), or is included through alternative means, such as additional text hidden with the .visually-hidden class.

Accented tables

Striped rows

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

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

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

These classes can also be added to table variants:


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

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

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

Hoverable rows

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

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

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


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

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

#	First	Last	Handle
1	Mark	Otto	@mdo

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

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

Active tables

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

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

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


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

How do the variants and accented tables work?

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

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

Behind the scenes it looks like this:

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

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

    color: $color;
    border-color: mix($color, $background, percentage($table-border-factor));
  }
}
```

Table borders

Bordered tables

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

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

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

Border color utilities can be added to change colors:

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

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

Tables without borders

Add .table-borderless for a table without borders.

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

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


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

Small tables

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

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

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



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

Vertical alignment

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

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

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

Nesting

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

#	First	Last	Handle
1	Mark	Otto	@mdo

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

3	Larry	the Bird	@twitter

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

How nesting works

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

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

Anatomy

Table head

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

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

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

1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

```
</table>
```

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

Table foot

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

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

Captions

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

List of users

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

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

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

List of users

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

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

Responsive tables

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

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

Always responsive

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

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

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

Breakpoint specific

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

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

#	Heading	Heading	Heading	Heading	Heading	Heading	Heading	Heading
1	Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell

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

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

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

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

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

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

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

Sass

Variables

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

$table-cell-vertical-align:   top;

$table-color:                 $body-color;
$table-bg:                    transparent;
$table-accent-bg:             transparent;

$table-th-font-weight:        null;

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

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

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

$table-border-factor:        .1;
$table-border-width:         $border-width;
$table-border-color:        $border-color;

$table-striped-order:        odd;

$table-group-separator-color: currentColor;

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

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

Loop

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

Customizing

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

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 <img> to make it responsive.

400x300A caption for the above image.

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

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

400x300A caption for the above image.

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

Sass

Variables

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

Forms

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

Form control

Style textual inputs and textareas with support for multiple states.

Select

Improve browser default select elements with a custom initial appearance.

Checks & radios

Use our custom radio buttons and checkboxes in forms for selecting input options.

Range

Replace browser default range inputs with our custom version.

Input group

Attach labels and buttons to your inputs for increased semantic value.

Floating labels

Create beautifully simple form labels that float over your input fields.

Layout

Create inline, horizontal, or complex grid-based layouts with your forms.

Validation

Validate your forms with custom or native validation behaviors and styles.

Overview

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

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

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

Email address

We'll never share your email with anyone else.

Password

☐ Check me out

Submit

```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail1" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp">
    <div id="emailHelp" class="form-text">We'll never share your email with anyone else.</div>
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword1" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

Form text

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

Associating form text with form controls

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

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

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

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

Password

Must be 8-20 characters long.

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

Disabled forms

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

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

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

However, if your form also includes custom button-like elements such as <a class="btn btn-*">...</a>, these will only be given a style of pointer-events: none, meaning they are still focusable and operable using the keyboard. In this case, you must manually modify these controls by adding tabindex="-1" to prevent them from receiving focus and aria-disabled="disabled" to signal their state to assistive technologies.

Disabled fieldset example

Disabled input

Disabled select menu

☐ Can't check this

Submit

```
<form>
  <fieldset disabled>
    <legend>Disabled fieldset example</legend>
    <div class="mb-3">
      <label for="disabledTextInput" class="form-label">Disabled input</label>
      <input type="text" id="disabledTextInput" class="form-control" placeholder="Disabled input">
    </div>
    <div class="mb-3">
      <label for="disabledSelect" class="form-label">Disabled select menu</label>
      <select id="disabledSelect" class="form-select">
        <option>Disabled select</option>
      </select>
    </div>
    <div class="mb-3">
      <div class="form-check">
        <input class="form-check-input" type="checkbox" id="disabledFieldsetCheck" disabled>
        <label class="form-check-label" for="disabledFieldsetCheck">
          Can't check this
        </label>
      </div>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
  </fieldset>
</form>
```

Accessibility

Ensure that all form controls have an appropriate accessible name so that their purpose can be conveyed to users of assistive technologies. The simplest way to achieve this is to use a `<label>` element, or—in the case of buttons—to include sufficiently descriptive text as part of the `<button>...</button>` content.

For situations where it's not possible to include a visible `<label>` or appropriate text content, there are alternative ways of still providing an accessible name, such as:

- `<label>` elements hidden using the `.visually-hidden` class
- Pointing to an existing element that can act as a label using `aria-labelledby`
- Providing a `title` attribute
- Explicitly setting the accessible name on an element using `aria-label`

If none of these are present, assistive technologies may resort to using the `placeholder` attribute as a fallback for the accessible name on `<input>` and `<textarea>` elements. The examples in this section provide a few suggested, case-specific approaches.

While using visually hidden content (`.visually-hidden`, `aria-label`, and even `placeholder` content, which disappears once a form field has content) will benefit assistive technology users, a lack of visible label text may still be problematic for certain users. Some form of visible label is generally the best approach, both for accessibility and usability.

Sass

Many form variables are set at a general level to be re-used and extended by individual form components. You'll see these most often as `$btn-input-*` and `$input-*` variables.

Variables

`$btn-input-*` variables are shared global variables between our buttons and our form components. You'll find these frequently reassigned as values to other component-specific variables.

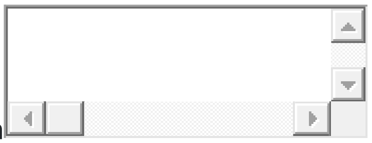
```
$input-btn-padding-y:         .375rem;
$input-btn-padding-x:         .75rem;
$input-btn-font-family:       null;
$input-btn-font-size:         $font-size-base;
$input-btn-line-height:       $line-height-base;

$input-btn-focus-width:         .25rem;
$input-btn-focus-color-opacity: .25;
$input-btn-focus-color:         rgba($component-active-bg, $input-btn-focus-color-opacity);
$input-btn-focus-blur:          0;
$input-btn-focus-box-shadow:    0 0 $input-btn-focus-blur $input-btn-focus-width $input-btn-focus-color;

$input-btn-padding-y-sm:      .25rem;
$input-btn-padding-x-sm:      .5rem;
$input-btn-font-size-sm:      $font-size-sm;

$input-btn-padding-y-lg:      .5rem;
$input-btn-padding-x-lg:      1rem;
$input-btn-font-size-lg:      $font-size-lg;

$input-btn-border-width:      $border-width;
```

Form controls

Give textual form controls like <input>s and <textarea>s an upgrade with custom styles, sizing, focus states, and more.

Example

Email address

Example textarea 

```
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Email address</label>
  <input type="email" class="form-control" id="exampleFormControlInput1" placeholder="name@example.com">
</div>
<div class="mb-3">
  <label for="exampleFormControlTextarea1" class="form-label">Example textarea</label>
  <textarea class="form-control" id="exampleFormControlTextarea1" rows="3"></textarea>
</div>
```

Sizing

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

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

Disabled

Add the disabled boolean attribute on an input to give it a grayed out appearance and remove pointer events.

Disabled re

```
<input class="form-control" type="text" placeholder="Disabled input" aria-label="Disabled input example"
disabled>
<input class="form-control" type="text" value="Disabled readonly input" aria-label="Disabled input example"
disabled readonly>
```

Readonly

Add the `readonly` boolean attribute on an input to prevent modification of the input's value.

Readonly i

```
<input class="form-control" type="text" value="Readonly input here..." aria-label="readonly input example"
readonly>
```

Readonly plain text

If you want to have `<input readonly>` elements in your form styled as plain text, use the `.form-control-plaintext` class to remove the default form field styling and preserve the correct margin and padding.

Email

email@exa

Password

```
 <div class="mb-3 row">
   <label for="staticEmail" class="col-sm-2 col-form-label">Email</label>
   <div class="col-sm-10">
     <input type="text" readonly class="form-control-plaintext" id="staticEmail" value="email@example.com">
   </div>
 </div>
 <div class="mb-3 row">
   <label for="inputPassword" class="col-sm-2 col-form-label">Password</label>
   <div class="col-sm-10">
     <input type="password" class="form-control" id="inputPassword">
   </div>
 </div>
```

Email
email@exa

Password

Confirm identity

```
<form class="row g-3">
  <div class="col-auto">
    <label for="staticEmail2" class="visually-hidden">Email</label>
    <input type="text" readonly class="form-control-plaintext" id="staticEmail2" value="email@example.com">
  </div>
  <div class="col-auto">
    <label for="inputPassword2" class="visually-hidden">Password</label>
    <input type="password" class="form-control" id="inputPassword2" placeholder="Password">
  </div>
  <div class="col-auto">
    <button type="submit" class="btn btn-primary mb-3">Confirm identity</button>
  </div>
</form>
```

File input

Default file input example

Multiple files input example

Disabled file input example

Small file input example

Large file input example

```
<div class="mb-3">
  <label for="formFile" class="form-label">Default file input example</label>
  <input class="form-control" type="file" id="formFile">
</div>
<div class="mb-3">
  <label for="formFileMultiple" class="form-label">Multiple files input example</label>
  <input class="form-control" type="file" id="formFileMultiple" multiple>
</div>
<div class="mb-3">
  <label for="formFileDisabled" class="form-label">Disabled file input example</label>
  <input class="form-control" type="file" id="formFileDisabled" disabled>
</div>
<div class="mb-3">
  <label for="formFileSm" class="form-label">Small file input example</label>
  <input class="form-control form-control-sm" id="formFileSm" type="file">
</div>
<div>
  <label for="formFileLg" class="form-label">Large file input example</label>
  <input class="form-control form-control-lg" id="formFileLg" type="file">
```

```
</div>
```

Color

Color picker

```
<label for="exampleColorInput" class="form-label">Color picker</label>
<input type="color" class="form-control form-control-color" id="exampleColorInput" value="#563d7c"
title="Choose your color">
```

Datalists

Datalists allow you to create a group of `<option>`s that can be accessed (and autocompleted) from within an `<input>`. These are similar to `<select>` elements, but come with more menu styling limitations and differences. While most browsers and operating systems include some support for `<datalist>` elements, their styling is inconsistent at best.

Learn more about [support for datalist elements](#).

Datalist example

```
<label for="exampleDataList" class="form-label">Datalist example</label>
<input class="form-control" list="datalistOptions" id="exampleDataList" placeholder="Type to search...">
<datalist id="datalistOptions">
  <option value="San Francisco">
  <option value="New York">
  <option value="Seattle">
  <option value="Los Angeles">
  <option value="Chicago">
</datalist>
```

Sass

Variables

`$input-*` are shared across most of our form controls (and not buttons).

```
$input-padding-y:                $input-btn-padding-y;
$input-padding-x:                $input-btn-padding-x;
$input-font-family:              $input-btn-font-family;
$input-font-size:              $input-btn-font-size;
```

```
$input-font-weight:                    $font-weight-base;
$input-line-height:                    $input-btn-line-height;

$input-padding-y-sm:                   $input-btn-padding-y-sm;
$input-padding-x-sm:                   $input-btn-padding-x-sm;
$input-font-size-sm:                   $input-btn-font-size-sm;

$input-padding-y-lg:                   $input-btn-padding-y-lg;
$input-padding-x-lg:                   $input-btn-padding-x-lg;
$input-font-size-lg:                   $input-btn-font-size-lg;

$input-bg:                             $white;
$input-disabled-bg:                    $gray-200;
$input-disabled-border-color:          null;

$input-color:                          $body-color;
$input-border-color:                   $gray-400;
$input-border-width:                   $input-btn-border-width;
$input-box-shadow:                     $box-shadow-inset;

$input-border-radius:                  $border-radius;
$input-border-radius-sm:               $border-radius-sm;
$input-border-radius-lg:               $border-radius-lg;

$input-focus-bg:                       $input-bg;
$input-focus-border-color:             tint-color($component-active-bg, 50%);
$input-focus-color:                    $input-color;
$input-focus-width:                    $input-btn-focus-width;
$input-focus-box-shadow:               $input-btn-focus-box-shadow;

$input-placeholder-color:              $gray-600;
$input-plaintext-color:                $body-color;

$input-height-border:                  $input-border-width * 2;

$input-height-inner:                   add($input-line-height * 1em, $input-padding-y * 2);
$input-height-inner-half:              add($input-line-height * .5em, $input-padding-y);
$input-height-inner-quarter:           add($input-line-height * .25em, $input-padding-y * .5);

$input-height:                         add($input-line-height * 1em, add($input-padding-y * 2, $input-height-border, false));
$input-height-sm:                      add($input-line-height * 1em, add($input-padding-y-sm * 2, $input-height-border,
false));
$input-height-lg:                      add($input-line-height * 1em, add($input-padding-y-lg * 2, $input-height-border,
false));

$input-transition:                     border-color .15s ease-in-out, box-shadow .15s ease-in-out;
```

`$form-label-*` and `$form-text-*` are for our `<label>`s and `.form-text` component.

```
$form-label-margin-bottom:             .5rem;
$form-label-font-size:                 null;
$form-label-font-style:                null;
$form-label-font-weight:               null;
$form-label-color:                     null;
```

```
$form-text-margin-top:                  .25rem;
$form-text-font-size:                   $small-font-size;
$form-text-font-style:                  null;
$form-text-font-weight:                 null;
$form-text-color:                       $text-muted;
```

$form-file-* are for file input.

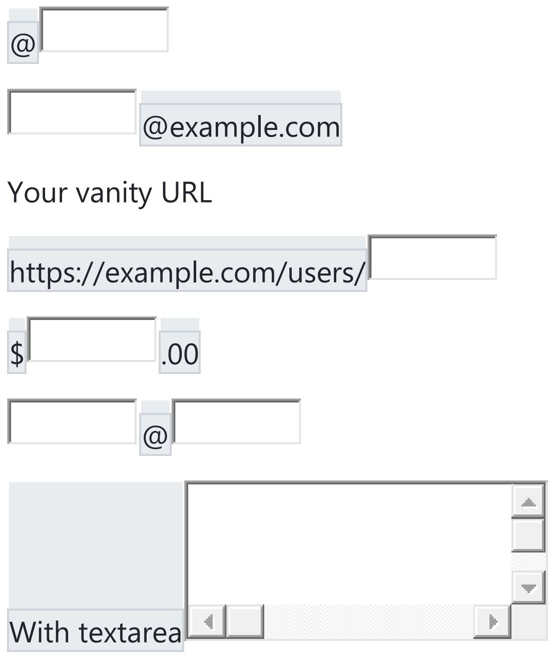
```
$form-file-button-color:          $input-color;
$form-file-button-bg:             $input-group-addon-bg;
$form-file-button-hover-bg:       shade-color($form-file-button-bg, 5%);
```

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.

Basic example

Place one add-on or button on either side of an input. You may also place one on both sides of an input. Remember to place `<label>`s outside the input group.



```
<div class="input-group mb-3">
  <span class="input-group-text" id="basic-addon1">@</span>
  <input type="text" class="form-control" placeholder="Username" aria-label="Username" aria-describedby="basic-
addon1">
</div>

<div class="input-group mb-3">
  <input type="text" class="form-control" placeholder="Recipient's username" aria-label="Recipient's username"
aria-describedby="basic-addon2">
  <span class="input-group-text" id="basic-addon2">@example.com</span>
</div>

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

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

<div class="input-group mb-3">
  <input type="text" class="form-control" placeholder="Username" aria-label="Username">
  <span class="input-group-text">@</span>
  <input type="text" class="form-control" placeholder="Server" aria-label="Server">
</div>

<div class="input-group">
  <span class="input-group-text">With textarea</span>
  <textarea class="form-control" aria-label="With textarea"></textarea>
</div>
```

Wrapping

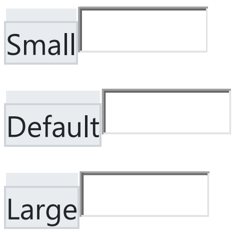
Input groups wrap by default via `flex-wrap: wrap` in order to accommodate custom form field validation within an input group. You may disable this with `.flex-nowrap`.

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

Sizing

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

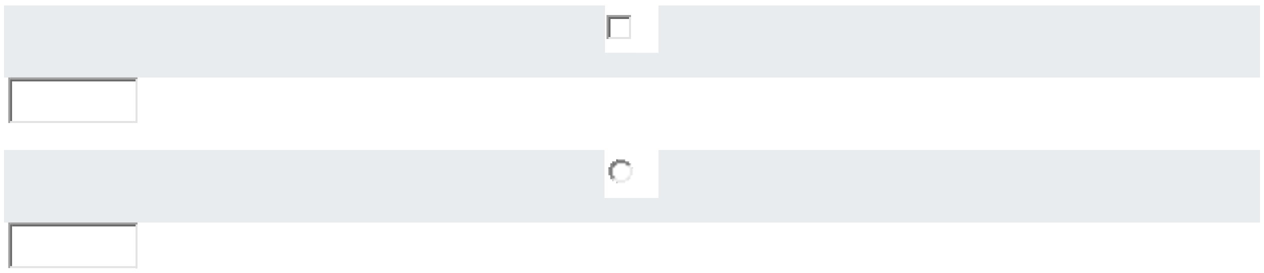
Sizing on the individual input group elements isn't supported.



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

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

<div class="input-group input-group-lg">
  <span class="input-group-text" id="inputGroup-sizing-lg">Large</span>
  <input type="text" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-
lg">
</div>
```

Checkboxes and radios

Place any checkbox or radio option within an input group's addon instead of text. We recommend adding .mt-0 to the .form-check-input when there's no visible text next to the input.



```
<div class="input-group mb-3">
  <div class="input-group-text">
    <input class="form-check-input mt-0" type="checkbox" value="" aria-label="Checkbox for following text
input">
  </div>
  <input type="text" class="form-control" aria-label="Text input with checkbox">
</div>

<div class="input-group">
  <div class="input-group-text">
    <input class="form-check-input mt-0" type="radio" value="" aria-label="Radio button for following text input">
  </div>
  <input type="text" class="form-control" aria-label="Text input with radio button">
</div>
```

Multiple inputs

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

First and last name

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

Multiple addons

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

$0.00

$0.00

```
<div class="input-group mb-3">
  <span class="input-group-text">$</span>
  <span class="input-group-text">0.00</span>
  <input type="text" class="form-control" aria-label="Dollar amount (with dot and two decimal places)">
</div>

<div class="input-group">
  <input type="text" class="form-control" aria-label="Dollar amount (with dot and two decimal places)">
  <span class="input-group-text">$</span>
  <span class="input-group-text">0.00</span>
</div>
```

Button addons

Button

Button

ButtonButton

ButtonButton

```
<div class="input-group mb-3">
  <button class="btn btn-outline-secondary" type="button" id="button-addon1">Button</button>
  <input type="text" class="form-control" placeholder="" aria-label="Example text with button addon" aria-
describedby="button-addon1">
</div>

<div class="input-group mb-3">
  <input type="text" class="form-control" placeholder="Recipient's username" aria-label="Recipient's username"
aria-describedby="button-addon2">
  <button class="btn btn-outline-secondary" type="button" id="button-addon2">Button</button>
</div>

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

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

Buttons with dropdowns

Dropdown

Dropdown

Dropdown Dropdown

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

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

<div class="input-group">
  <button class="btn btn-outline-secondary dropdown-toggle" type="button" data-bs-toggle="dropdown" aria-
expanded="false">Dropdown</button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action before</a></li>
    <li><a class="dropdown-item" href="#">Another action before</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
  <input type="text" class="form-control" aria-label="Text input with 2 dropdown buttons">
  <button class="btn btn-outline-secondary dropdown-toggle" type="button" data-bs-toggle="dropdown" aria-
expanded="false">Dropdown</button>
  <ul class="dropdown-menu dropdown-menu-end">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>
```

Segmented buttons

ActionToggle Dropdown

ActionToggle Dropdown

```
<div class="input-group mb-3">
  <button type="button" class="btn btn-outline-secondary">Action</button>
  <button type="button" class="btn btn-outline-secondary dropdown-toggle dropdown-toggle-split" data-bs-
toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
  <input type="text" class="form-control" aria-label="Text input with segmented dropdown button">
</div>

<div class="input-group">
  <input type="text" class="form-control" aria-label="Text input with segmented dropdown button">
  <button type="button" class="btn btn-outline-secondary">Action</button>
  <button type="button" class="btn btn-outline-secondary dropdown-toggle dropdown-toggle-split" data-bs-
toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu dropdown-menu-end">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>
```

Custom forms

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

Custom select

Options Choose...

Choose... Options

Button Choose...

Choose... Button

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

<div class="input-group mb-3">
```

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

<div class="input-group mb-3">
  <button class="btn btn-outline-secondary" type="button">Button</button>
  <select class="form-select" id="inputGroupSelect03" aria-label="Example select with button addon">
    <option selected>Choose...</option>
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </select>
</div>

<div class="input-group">
  <select class="form-select" id="inputGroupSelect04" aria-label="Example select with button addon">
    <option selected>Choose...</option>
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </select>
  <button class="btn btn-outline-secondary" type="button">Button</button>
</div>
```

Custom file input

Upload

Upload

Button

Button

```
<div class="input-group mb-3">
  <label class="input-group-text" for="inputGroupFile01">Upload</label>
  <input type="file" class="form-control" id="inputGroupFile01">
</div>

<div class="input-group mb-3">
  <input type="file" class="form-control" id="inputGroupFile02">
  <label class="input-group-text" for="inputGroupFile02">Upload</label>
</div>

<div class="input-group mb-3">
  <button class="btn btn-outline-secondary" type="button" id="inputGroupFileAddon03">Button</button>
  <input type="file" class="form-control" id="inputGroupFile03" aria-describedby="inputGroupFileAddon03" aria-
label="Upload">
```

```
</div>

<div class="input-group">
  <input type="file" class="form-control" id="inputGroupFile04" aria-describedby="inputGroupFileAddon04" aria-label="Upload">
  <button class="btn btn-outline-secondary" type="button" id="inputGroupFileAddon04">Button</button>
</div>
```

Sass

Variables

```
$input-group-addon-padding-y:          $input-padding-y;
$input-group-addon-padding-x:          $input-padding-x;
$input-group-addon-font-weight:        $input-font-weight;
$input-group-addon-color:              $input-color;
$input-group-addon-bg:                 $gray-200;
$input-group-addon-border-color:       $input-border-color;
```

Floating labels

Create beautifully simple form labels that float over your input fields.

Example

Wrap a pair of `<input class="form-control">` and `<label>` elements in `.form-floating` to enable floating labels with Bootstrap's textual form fields. A `placeholder` is required on each `<input>` as our method of CSS-only floating labels uses the `:placeholder-shown` pseudo-element. Also note that the `<input>` must come first so we can utilize a sibling selector (e.g., `~`).

Email address

Password

```
<div class="form-floating mb-3">
  <input type="email" class="form-control" id="floatingInput" placeholder="name@example.com">
  <label for="floatingInput">Email address</label>
</div>
<div class="form-floating">
  <input type="password" class="form-control" id="floatingPassword" placeholder="Password">
  <label for="floatingPassword">Password</label>
</div>
```

When there's a `value` already defined, `<label>`s will automatically adjust to their floated position.

Input with value

```
<form class="form-floating">
  <input type="email" class="form-control" id="floatingInputValue" placeholder="name@example.com"
value="test@example.com">
  <label for="floatingInputValue">Input with value</label>
</form>
```

Form validation styles also work as expected.

Invalid input

```
<form class="form-floating">
  <input type="email" class="form-control is-invalid" id="floatingInputInvalid" placeholder="name@example.com"
value="test@example.com">
```

```
  <label for="floatingInputInvalid">Invalid input</label>
</form>
```

Textareas

By default, <textarea>s with .form-control will be the same height as <input>s.

```
<div class="form-floating">
  <textarea class="form-control" placeholder="Leave a comment here" id="floatingTextarea"></textarea>
  <label for="floatingTextarea">Comments</label>
</div>
```

To set a custom height on your <textarea>, do not use the rows attribute. Instead, set an explicit height (either inline or via custom CSS).

```
<div class="form-floating">
  <textarea class="form-control" placeholder="Leave a comment here" id="floatingTextarea2" style="height:
100px"></textarea>
  <label for="floatingTextarea2">Comments</label>
</div>
```

Selects

Other than .form-control, floating labels are only available on .form-selects. They work in the same way, but unlike <input>s, they'll always show the <label> in its floated state. **Selects with size and multiple are not supported.**

```
<div class="form-floating">
  <select class="form-select" id="floatingSelect" aria-label="Floating label select example">
```

```
    <option selected>Open this select menu</option>
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </select>
  <label for="floatingSelect">Works with selects</label>
</div>
```

Layout

When working with the Bootstrap grid system, be sure to place form elements within column classes.

Email address

Open this select menu Works with selects

```
<div class="row g-2">
  <div class="col-md">
    <div class="form-floating">
      <input type="email" class="form-control" id="floatingInputGrid" placeholder="name@example.com"
value="mdo@example.com">
      <label for="floatingInputGrid">Email address</label>
    </div>
  </div>
  <div class="col-md">
    <div class="form-floating">
      <select class="form-select" id="floatingSelectGrid" aria-label="Floating label select example">
        <option selected>Open this select menu</option>
        <option value="1">One</option>
        <option value="2">Two</option>
        <option value="3">Three</option>
      </select>
      <label for="floatingSelectGrid">Works with selects</label>
    </div>
  </div>
</div>
```

Sass

Variables

```
$form-floating-height:           add(3.5rem, $input-height-border);
$form-floating-line-height:        1.25;
$form-floating-padding-x:          $input-padding-x;
$form-floating-padding-y:          1rem;
```

```
$form-floating-input-padding-t:   1.625rem;
$form-floating-input-padding-b:   .625rem;
$form-floating-label-opacity:     .65;
$form-floating-label-transform:   scale(.85) translateY(-.5rem) translateX(.15rem);
$form-floating-transition:        opacity .1s ease-in-out, transform .1s ease-in-out;
```

Validation

Provide valuable, actionable feedback to your users with HTML5 form validation, via browser default behaviors or custom styles and JavaScript.

We are aware that currently the client-side custom validation styles and tooltips are not accessible, since they are not exposed to assistive technologies. While we work on a solution, we'd recommend either using the server-side option or the default browser validation method.

How it works

Here's how form validation works with Bootstrap:

- HTML form validation is applied via CSS's two pseudo-classes, `:invalid` and `:valid`. It applies to `<input>`, `<select>`, and `<textarea>` elements.
- Bootstrap scopes the `:invalid` and `:valid` styles to parent `.was-validated` class, usually applied to the `<form>`. Otherwise, any required field without a value shows up as invalid on page load. This way, you may choose when to activate them (typically after form submission is attempted).
- To reset the appearance of the form (for instance, in the case of dynamic form submissions using AJAX), remove the `.was-validated` class from the `<form>` again after submission.
- As a fallback, `.is-invalid` and `.is-valid` classes may be used instead of the pseudo-classes for server-side validation. They do not require a `.was-validated` parent class.
- Due to constraints in how CSS works, we cannot (at present) apply styles to a `<label>` that comes before a form control in the DOM without the help of custom JavaScript.
- All modern browsers support the constraint validation API, a series of JavaScript methods for validating form controls.
- Feedback messages may utilize the browser defaults (different for each browser, and unstylable via CSS) or our custom feedback styles with additional HTML and CSS.
- You may provide custom validity messages with `setCustomValidity` in JavaScript.

With that in mind, consider the following demos for our custom form validation styles, optional server-side classes, and browser defaults.

Custom styles

For custom Bootstrap form validation messages, you'll need to add the `novalidate` boolean attribute to your `<form>`. This disables the browser default feedback tooltips, but still provides access to the form validation APIs in JavaScript. Try to submit the form below;

our JavaScript will intercept the submit button and relay feedback to you. When attempting to submit, you'll see the :invalid and :valid styles applied to your form controls.

Custom feedback styles apply custom colors, borders, focus styles, and background icons to better communicate feedback. Background icons for <select>s are only available with .form-select, and not .form-control.

First name Mark

Last name Otto

Username

@

City

State Choose...

Zip

☐ Agree to terms and conditions

Submit form

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

```
        Please choose a username.
      </div>
    </div>
  </div>
  <div class="col-md-6">
    <label for="validationCustom03" class="form-label">City</label>
    <input type="text" class="form-control" id="validationCustom03" required>
    <div class="invalid-feedback">
      Please provide a valid city.
    </div>
  </div>
  <div class="col-md-3">
    <label for="validationCustom04" class="form-label">State</label>
    <select class="form-select" id="validationCustom04" required>
      <option selected disabled value="">Choose...</option>
      <option>...</option>
    </select>
    <div class="invalid-feedback">
      Please select a valid state.
    </div>
  </div>
  <div class="col-md-3">
    <label for="validationCustom05" class="form-label">Zip</label>
    <input type="text" class="form-control" id="validationCustom05" required>
    <div class="invalid-feedback">
      Please provide a valid zip.
    </div>
  </div>
  <div class="col-12">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" value="" id="invalidCheck" required>
      <label class="form-check-label" for="invalidCheck">
        Agree to terms and conditions
      </label>
      <div class="invalid-feedback">
        You must agree before submitting.
      </div>
    </div>
  </div>
  <div class="col-12">
    <button class="btn btn-primary" type="submit">Submit form</button>
  </div>
</form>


// Example starter JavaScript for disabling form submissions if there are invalid fields
(function () {
  'use strict'

  // Fetch all the forms we want to apply custom Bootstrap validation styles to
  var forms = document.querySelectorAll('.needs-validation')

  // Loop over them and prevent submission
  Array.prototype.slice.call(forms)
    .forEach(function (form) {
      form.addEventListener('submit', function (event) {
        if (!form.checkValidity()) {
          event.preventDefault()
          event.stopPropagation()
        }

        form.classList.add('was-validated')
      }, false)
    })
})()
```

Browser defaults

Not interested in custom validation feedback messages or writing JavaScript to change form behaviors? All good, you can use the browser defaults. Try submitting the form below. Depending on your browser and OS, you'll see a slightly different style of feedback.

While these feedback styles cannot be styled with CSS, you can still customize the feedback text through JavaScript.

First name Mark

Last name Otto

Username

@

City

State Choose...

Zip

☐ Agree to terms and conditions

Submit form

```
<form class="row g-3">
  <div class="col-md-4">
    <label for="validationDefault01" class="form-label">First name</label>
    <input type="text" class="form-control" id="validationDefault01" value="Mark" required>
  </div>
  <div class="col-md-4">
    <label for="validationDefault02" class="form-label">Last name</label>
    <input type="text" class="form-control" id="validationDefault02" value="Otto" required>
  </div>
  <div class="col-md-4">
    <label for="validationDefaultUsername" class="form-label">Username</label>
    <div class="input-group">
      <span class="input-group-text" id="inputGroupPrepend2">@</span>
      <input type="text" class="form-control" id="validationDefaultUsername"  aria-
describedby="inputGroupPrepend2" required>
    </div>
  </div>
  <div class="col-md-6">
    <label for="validationDefault03" class="form-label">City</label>
    <input type="text" class="form-control" id="validationDefault03" required>
  </div>
  <div class="col-md-3">
    <label for="validationDefault04" class="form-label">State</label>
    <select class="form-select" id="validationDefault04" required>
      <option selected disabled value="">Choose...</option>
      <option>...</option>
    </select>
  </div>
  <div class="col-md-3">
    <label for="validationDefault05" class="form-label">Zip</label>
    <input type="text" class="form-control" id="validationDefault05" required>
  </div>
  <div class="col-12">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" value="" id="invalidCheck2" required>
      <label class="form-check-label" for="invalidCheck2">
        Agree to terms and conditions
      </label>
    </div>
  </div>
  <div class="col-12">
    <button class="btn btn-primary" type="submit">Submit form</button>
  </div>
</form>
```

Server side

We recommend using client-side validation, but in case you require server-side validation, you can indicate invalid and valid form fields with `.is-invalid` and `.is-valid`. Note that `.invalid-feedback` is also supported with these classes.

For invalid fields, ensure that the invalid feedback/error message is associated with the relevant form field using `aria-describedby` (noting that this attribute allows more than one `id` to be referenced, in case the field already points to additional form text).

To fix issues with border radii, input groups require an additional `.has-validation` class.

First name Mark

Looks good!

Last name Otto

Looks good!

Username

@

Please choose a username.

City

Please provide a valid city.

State Choose...

Please select a valid state.

Zip

Please provide a valid zip.

☐ Agree to terms and conditions

You must agree before submitting.

Submit form

```
<form class="row g-3">
  <div class="col-md-4">
    <label for="validationServer01" class="form-label">First name</label>
    <input type="text" class="form-control is-valid" id="validationServer01" value="Mark" required>
    <div class="valid-feedback">
      Looks good!
    </div>
  </div>
  <div class="col-md-4">
    <label for="validationServer02" class="form-label">Last name</label>
    <input type="text" class="form-control is-valid" id="validationServer02" value="Otto" required>
    <div class="valid-feedback">
      Looks good!
    </div>
  </div>
  <div class="col-md-4">
    <label for="validationServerUsername" class="form-label">Username</label>
    <div class="input-group has-validation">
      <span class="input-group-text" id="inputGroupPrepend3">@</span>
      <input type="text" class="form-control is-invalid" id="validationServerUsername" aria-
describedby="inputGroupPrepend3 validationServerUsernameFeedback" required>
      <div id="validationServerUsernameFeedback" class="invalid-feedback">
        Please choose a username.
      </div>
    </div>
  </div>
  <div class="col-md-6">
    <label for="validationServer03" class="form-label">City</label>
    <input type="text" class="form-control is-invalid" id="validationServer03" aria-
describedby="validationServer03Feedback" required>
    <div id="validationServer03Feedback" class="invalid-feedback">
      Please provide a valid city.
    </div>
  </div>
  <div class="col-md-3">
    <label for="validationServer04" class="form-label">State</label>
    <select class="form-select is-invalid" id="validationServer04" aria-describedby="validationServer04Feedback"
required>
      <option selected disabled value="">Choose...</option>
      <option>...</option>
    </select>
    <div id="validationServer04Feedback" class="invalid-feedback">
      Please select a valid state.
    </div>
  </div>
  <div class="col-md-3">
    <label for="validationServer05" class="form-label">Zip</label>
    <input type="text" class="form-control is-invalid" id="validationServer05" aria-
describedby="validationServer05Feedback" required>
    <div id="validationServer05Feedback" class="invalid-feedback">
      Please provide a valid zip.
    </div>
  </div>
  <div class="col-12">
    <div class="form-check">
      <input class="form-check-input is-invalid" type="checkbox" value="" id="invalidCheck3" aria-
describedby="invalidCheck3Feedback" required>
      <label class="form-check-label" for="invalidCheck3">
        Agree to terms and conditions
      </label>
      <div id="invalidCheck3Feedback" class="invalid-feedback">
        You must agree before submitting.
      </div>
    </div>
  </div>
  <div class="col-12">
    <button class="btn btn-primary" type="submit">Submit form</button>
  </div>
</form>
```

Supported elements

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

- `<input>`s and `<textarea>`s with `.form-control` (including up to one `.form-control` in input groups)
- `<select>`s with `.form-select`
- `.form-check`s

Textarea

Please enter a message in the textarea.

☐ Check this checkbox

Example invalid feedback text

○ Toggle this radio

○ Or toggle this other radio

More example invalid feedback text

Example invalid select feedback

Example invalid form file feedback

Submit form

```
<form class="was-validated">
  <div class="mb-3">
    <label for="validationTextarea" class="form-label">Textarea</label>
    <textarea class="form-control is-invalid" id="validationTextarea" placeholder="Required example textarea"
required></textarea>
    <div class="invalid-feedback">
      Please enter a message in the textarea.
    </div>
  </div>

  <div class="form-check mb-3">
    <input type="checkbox" class="form-check-input" id="validationFormCheck1" required>
    <label class="form-check-label" for="validationFormCheck1">Check this checkbox</label>
    <div class="invalid-feedback">Example invalid feedback text</div>
  </div>

  <div class="form-check">
    <input type="radio" class="form-check-input" id="validationFormCheck2" name="radio-stacked" required>
    <label class="form-check-label" for="validationFormCheck2">Toggle this radio</label>
  </div>
  <div class="form-check mb-3">
    <input type="radio" class="form-check-input" id="validationFormCheck3" name="radio-stacked" required>
    <label class="form-check-label" for="validationFormCheck3">Or toggle this other radio</label>
    <div class="invalid-feedback">More example invalid feedback text</div>
  </div>

  <div class="mb-3">
    <select class="form-select" required aria-label="select example">
      <option value="">Open this select menu</option>
      <option value="1">One</option>
      <option value="2">Two</option>
      <option value="3">Three</option>
    </select>
    <div class="invalid-feedback">Example invalid select feedback</div>
  </div>

  <div class="mb-3">
    <input type="file" class="form-control" aria-label="file example" required>
    <div class="invalid-feedback">Example invalid form file feedback</div>
  </div>

  <div class="mb-3">
    <button class="btn btn-primary" type="submit" disabled>Submit form</button>
  </div>
</form>
```

Tooltips

If your form layout allows it, you can swap the `.{valid|invalid}-feedback` classes for `.{valid|invalid}-tooltip` classes to display validation feedback in a styled tooltip. Be sure to have a parent with `position: relative` on it for tooltip positioning. In the example below, our column classes have this already, but your project may require an alternative setup.

First name Mark

Last name Otto

Username

@

City

State Choose...

Zip

Submit form

```
<form class="row g-3 needs-validation" novalidate>
  <div class="col-md-4 position-relative">
    <label for="validationTooltip01" class="form-label">First name</label>
    <input type="text" class="form-control" id="validationTooltip01" value="Mark" required>
    <div class="valid-tooltip">
      Looks good!
    </div>
  </div>
  <div class="col-md-4 position-relative">
    <label for="validationTooltip02" class="form-label">Last name</label>
    <input type="text" class="form-control" id="validationTooltip02" value="Otto" required>
    <div class="valid-tooltip">
      Looks good!
    </div>
  </div>
  <div class="col-md-4 position-relative">
    <label for="validationTooltipUsername" class="form-label">Username</label>
    <div class="input-group has-validation">
      <span class="input-group-text" id="validationTooltipUsernamePrepend">@</span>
      <input type="text" class="form-control" id="validationTooltipUsername" aria-
describedby="validationTooltipUsernamePrepend" required>
      <div class="invalid-tooltip">
        Please choose a unique and valid username.
      </div>
    </div>
  </div>
  <div class="col-md-6 position-relative">
    <label for="validationTooltip03" class="form-label">City</label>
    <input type="text" class="form-control" id="validationTooltip03" required>
    <div class="invalid-tooltip">
      Please provide a valid city.
    </div>
  </div>
  <div class="col-md-3 position-relative">
    <label for="validationTooltip04" class="form-label">State</label>
    <select class="form-select" id="validationTooltip04" required>
      <option selected disabled value="">Choose...</option>
      <option>...</option>
    </select>
    <div class="invalid-tooltip">
      Please select a valid state.
    </div>
  </div>
  <div class="col-md-3 position-relative">
    <label for="validationTooltip05" class="form-label">Zip</label>
    <input type="text" class="form-control" id="validationTooltip05" required>
    <div class="invalid-tooltip">
```

```
      Please provide a valid zip.
    </div>
  </div>
  <div class="col-12">
    <button class="btn btn-primary" type="submit">Submit form</button>
  </div>
</form>
```

Sass

Variables

```
$form-feedback-margin-top:          $form-text-margin-top;
$form-feedback-font-size:           $form-text-font-size;
$form-feedback-font-style:          $form-text-font-style;
$form-feedback-valid-color:         $success;
$form-feedback-invalid-color:       $danger;

$form-feedback-icon-valid-color:    $form-feedback-valid-color;
$form-feedback-icon-valid:          url("data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg' viewBox='0 0 8 8'><path fill='#{$form-feedback-icon-valid-color}' d='M2.3 6.73L.6 4.53c-.4-1.04.46-1.4 1.1-.8l1.1 1.4 3.4-3.8c.6-.63 1.6-.27 1.2.7l-4 4.6c-.43.5-.8.4-1.1.1z'/></svg>");
$form-feedback-icon-invalid-color:  $form-feedback-invalid-color;
$form-feedback-icon-invalid:        url("data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg' viewBox='0 0 12 12' width='12' height='12' fill='none' stroke='#{$form-feedback-icon-invalid-color}'><circle cx='6' cy='6' r='4.5'/><path stroke-linejoin='round' d='M5.8 3.6h.4L6 6.5z'/><circle cx='6' cy='8.2' r='.6' fill='#{$form-feedback-icon-invalid-color}' stroke='none'/></svg>");
```

Mixins

Two mixins are combined together, through our loop, to generate our form validation feedback styles.

```
@mixin form-validation-state-selector($state) {
  @if ($state == "valid" or $state == "invalid") {
    .was-validated #{if(&, "&", "")}:#{$state},
    #{if(&, "&", "")}.is-#{$state} {
      @content;
    }
  } @else {
    #{if(&, "&", "")}.is-#{$state} {
      @content;
    }
  }
}

@mixin form-validation-state(
  $state,
  $color,
  $icon,
  $tooltip-color: color-contrast($color),
  $tooltip-bg-color: rgba($color, $form-feedback-tooltip-opacity),
  $focus-box-shadow: 0 0 $input-btn-focus-blur $input-focus-width rgba($color, $input-btn-focus-color-opacity)
) {
  .#{$state}-feedback {
    display: none;
    width: 100%;
    margin-top: $form-feedback-margin-top;
    @include font-size($form-feedback-font-size);
    font-style: $form-feedback-font-style;
    color: $color;
  }

  .#{$state}-tooltip {
    position: absolute;
    top: 100%;
    z-index: 5;
    display: none;
    max-width: 100%; // Contain to parent when possible
    padding: $form-feedback-tooltip-padding-y $form-feedback-tooltip-padding-x;
    margin-top: .1rem;
    @include font-size($form-feedback-tooltip-font-size);
    line-height: $form-feedback-tooltip-line-height;
    color: $tooltip-color;
    background-color: $tooltip-bg-color;
    @include border-radius($form-feedback-tooltip-border-radius);
  }

  @include form-validation-state-selector($state) {
    ~ .#{$state}-feedback,
    ~ .#{$state}-tooltip {
      display: block;
    }
  }

  .form-control {
    @include form-validation-state-selector($state) {
      border-color: $color;

      @if $enable-validation-icons {
        padding-right: $input-height-inner;
        background-image: escape-svg($icon);
        background-repeat: no-repeat;
        background-position: right $input-height-inner-quarter center;
        background-size: $input-height-inner-half $input-height-inner-half;
      }

      &:focus {
        border-color: $color;
        box-shadow: $focus-box-shadow;
      }
    }
  }

  // stylelint-disable-next-line selector-no-qualifying-type
```

```
textarea.form-control {
  @include form-validation-state-selector($state) {
    @if $enable-validation-icons {
      padding-right: $input-height-inner;
      background-position: top $input-height-inner-quarter right $input-height-inner-quarter;
    }
  }
}

.form-select {
  @include form-validation-state-selector($state) {
    border-color: $color;

    @if $enable-validation-icons {
      &:not([multiple]):not([size]),
      &:not([multiple])[size="1"] {
        padding-right: $form-select-feedback-icon-padding-end;
        background-image: escape-svg($form-select-indicator), escape-svg($icon);
        background-position: $form-select-bg-position, $form-select-feedback-icon-position;
        background-size: $form-select-bg-size, $form-select-feedback-icon-size;
      }
    }

    &:focus {
      border-color: $color;
      box-shadow: $focus-box-shadow;
    }
  }
}

.form-check-input {
  @include form-validation-state-selector($state) {
    border-color: $color;

    &:checked {
      background-color: $color;
    }

    &:focus {
      box-shadow: $focus-box-shadow;
    }

    ~ .form-check-label {
      color: $color;
    }
  }
}
.form-check-inline .form-check-input {
  ~ .#{$state}-feedback {
    margin-left: .5em;
  }
}

.input-group .form-control,
.input-group .form-select {
  @include form-validation-state-selector($state) {
      @if $state == "valid" {
        z-index: 1;
      } @else if $state == "invalid" {
        z-index: 2;
      }
      &:focus {
        z-index: 3;
      }
    }
  }
}
```

Map

This is the validation Sass map from _variables.scss. Override or extend this to generate different or additional states.

```
$form-validation-states: (
  "valid": (
    "color": $form-feedback-valid-color,
    "icon": $form-feedback-icon-valid
  ),
  "invalid": (
    "color": $form-feedback-invalid-color,
    "icon": $form-feedback-icon-invalid
  )
);
```

Maps of $form-validation-states can contain three optional parameters to override tooltips and focus styles.

Loop

Used to iterate over $form-validation-states map values to generate our validation styles. Any modifications to the above Sass map will be reflected in your compiled CSS via this loop.

```
@each $state, $data in $form-validation-states {
  @include form-validation-state($state, $data...);
}
```

Customizing

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

Please note that **we do not recommend customizing $form-validation-states values without also modifying the form-validation-state mixin**.

Accordion

Build vertically collapsing accordions in combination with our Collapse JavaScript plugin.

How it works

The accordion uses collapse internally to make it collapsible. To render an accordion that's expanded, add the .open class on the .accordion.

The animation effect of this component is dependent on the prefers-reduced-motion media query. See the reduced motion section of our accessibility documentation.

Example

Click the accordions below to expand/collapse the accordion content.

Accordion Item #1

This is the first item's accordion body. It is shown by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the .accordion-body, though the transition does limit overflow.

Accordion Item #2

Accordion Item #3

```
<div class="accordion" id="accordionExample">
  <div class="accordion-item">
    <h2 class="accordion-header" id="headingOne">
      <button class="accordion-button" type="button" data-bs-toggle="collapse" data-bs-target="#collapseOne" aria-
expanded="true" aria-controls="collapseOne">
        Accordion Item #1
      </button>
    </h2>
    <div id="collapseOne" class="accordion-collapse collapse show" aria-labelledby="headingOne" data-bs-
parent="#accordionExample">
      <div class="accordion-body">
        <strong>This is the first item's accordion body.</strong> It is shown by default, until the collapse plugin adds
the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the
showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default
variables. It's also worth noting that just about any HTML can go within the <code>.accordion-body</code>, though
the transition does limit overflow.
```

```
      </div>
    </div>
  </div>
  <div class="accordion-item">
    <h2 class="accordion-header" id="headingTwo">
      <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#collapseTwo" aria-expanded="false" aria-controls="collapseTwo">
        Accordion Item #2
      </button>
    </h2>
    <div id="collapseTwo" class="accordion-collapse collapse" aria-labelledby="headingTwo" data-bs-parent="#accordionExample">
      <div class="accordion-body">
        <strong>This is the second item's accordion body.</strong> It is hidden by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the <code>.accordion-body</code>, though the transition does limit overflow.
      </div>
    </div>
  </div>
  <div class="accordion-item">
    <h2 class="accordion-header" id="headingThree">
      <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#collapseThree" aria-expanded="false" aria-controls="collapseThree">
        Accordion Item #3
      </button>
    </h2>
    <div id="collapseThree" class="accordion-collapse collapse" aria-labelledby="headingThree" data-bs-parent="#accordionExample">
      <div class="accordion-body">
        <strong>This is the third item's accordion body.</strong> It is hidden by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the <code>.accordion-body</code>, though the transition does limit overflow.
      </div>
    </div>
  </div>
</div>
```

Flush

Add `.accordion-flush` to remove the default `background-color`, some borders, and some rounded corners to render accordions edge-to-edge with their parent container.

Accordion Item #1

Accordion Item #2

Accordion Item #3

```
<div class="accordion accordion-flush" id="accordionFlushExample">
  <div class="accordion-item">
    <h2 class="accordion-header" id="flush-headingOne">
      <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#flush-collapseOne" aria-expanded="false" aria-controls="flush-collapseOne">
        Accordion Item #1
      </button>
    </h2>
    <div id="flush-collapseOne" class="accordion-collapse collapse" aria-labelledby="flush-headingOne" data-bs-parent="#accordionFlushExample">
      <div class="accordion-body">Placeholder content for this accordion, which is intended to demonstrate the <code>.accordion-flush</code> class. This is the first item's accordion body.</div>
    </div>
  </div>
  <div class="accordion-item">
    <h2 class="accordion-header" id="flush-headingTwo">
      <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#flush-collapseTwo" aria-expanded="false" aria-controls="flush-collapseTwo">
        Accordion Item #2
      </button>
    </h2>
    <div id="flush-collapseTwo" class="accordion-collapse collapse" aria-labelledby="flush-headingTwo" data-bs-parent="#accordionFlushExample">
      <div class="accordion-body">Placeholder content for this accordion, which is intended to demonstrate the <code>.accordion-flush</code> class. This is the second item's accordion body. Let's imagine this being filled with some actual content.</div>
    </div>
  </div>
  <div class="accordion-item">
    <h2 class="accordion-header" id="flush-headingThree">
      <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#flush-collapseThree" aria-expanded="false" aria-controls="flush-collapseThree">
        Accordion Item #3
      </button>
    </h2>
    <div id="flush-collapseThree" class="accordion-collapse collapse" aria-labelledby="flush-headingThree" data-bs-parent="#accordionFlushExample">
      <div class="accordion-body">Placeholder content for this accordion, which is intended to demonstrate the <code>.accordion-flush</code> class. This is the third item's accordion body. Nothing more exciting happening here in terms of content, but just filling up the space to make it look, at least at first glance, a bit more representative of how this would look in a real-world application.</div>
    </div>
  </div>
</div>
```

Always open

Omit the `data-bs-parent` attribute on each `.accordion-collapse` to make accordion items stay open when another item is opened.

Accordion Item #1

This is the first item's accordion body. It is shown by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the `.accordion-body`, though the transition does limit overflow.

Accordion Item #2

Accordion Item #3

```
<div class="accordion" id="accordionPanelsStayOpenExample">
  <div class="accordion-item">
    <h2 class="accordion-header" id="panelsStayOpen-headingOne">
      <button class="accordion-button" type="button" data-bs-toggle="collapse" data-bs-target="#panelsStayOpen-
collapseOne" aria-expanded="true" aria-controls="panelsStayOpen-collapseOne">
        Accordion Item #1
      </button>
    </h2>
    <div id="panelsStayOpen-collapseOne" class="accordion-collapse collapse show" aria-
labelledby="panelsStayOpen-headingOne">
      <div class="accordion-body">
        <strong>This is the first item's accordion body.</strong> It is shown by default, until the collapse plugin adds
the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the
showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default
variables. It's also worth noting that just about any HTML can go within the <code>.accordion-body</code>, though
the transition does limit overflow.
      </div>
    </div>
  </div>
  <div class="accordion-item">
    <h2 class="accordion-header" id="panelsStayOpen-headingTwo">
      <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-
target="#panelsStayOpen-collapseTwo" aria-expanded="false" aria-controls="panelsStayOpen-collapseTwo">
        Accordion Item #2
      </button>
    </h2>
    <div id="panelsStayOpen-collapseTwo" class="accordion-collapse collapse" aria-labelledby="panelsStayOpen-
headingTwo">
      <div class="accordion-body">
        <strong>This is the second item's accordion body.</strong> It is hidden by default, until the collapse plugin
adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well
as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our
default variables. It's also worth noting that just about any HTML can go within the <code>.accordion-body</code>,
though the transition does limit overflow.
      </div>
    </div>
  </div>
  <div class="accordion-item">
    <h2 class="accordion-header" id="panelsStayOpen-headingThree">
      <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-
target="#panelsStayOpen-collapseThree" aria-expanded="false" aria-controls="panelsStayOpen-collapseThree">
        Accordion Item #3
      </button>
    </h2>
    <div id="panelsStayOpen-collapseThree" class="accordion-collapse collapse" aria-labelledby="panelsStayOpen-headingThree">
      <div class="accordion-body">
        <strong>This is the third item's accordion body.</strong> It is hidden by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the <code>.accordion-body</code>, though the transition does limit overflow.
      </div>
    </div>
  </div>
</div>
```

Accessibility

Please read the collapse accessibility section for more information.

Sass

Variables

```
$accordion-padding-y:                     1rem;
$accordion-padding-x:                     1.25rem;
$accordion-color:                         $body-color;
$accordion-bg:                            $body-bg;
$accordion-border-width:                  $border-width;
$accordion-border-color:                  rgba($black, .125);
$accordion-border-radius:                 $border-radius;
$accordion-inner-border-radius:           subtract($accordion-border-radius, $accordion-border-width);

$accordion-body-padding-y:                $accordion-padding-y;
$accordion-body-padding-x:                $accordion-padding-x;

$accordion-button-padding-y:              $accordion-padding-y;
$accordion-button-padding-x:              $accordion-padding-x;
$accordion-button-color:                  $accordion-color;
$accordion-button-bg:                     $accordion-bg;
$accordion-transition:                    $btn-transition, border-radius .15s ease;
$accordion-button-active-bg:              tint-color($component-active-bg, 90%);
$accordion-button-active-color:           shade-color($primary, 10%);

$accordion-button-focus-border-color:     $input-focus-border-color;
$accordion-button-focus-box-shadow:       $btn-focus-box-shadow;

$accordion-icon-width:                    1.25rem;
$accordion-icon-color:                    $accordion-color;
$accordion-icon-active-color:             $accordion-button-active-color;
$accordion-icon-transition:               transform .2s ease-in-out;
```

```
$accordion-icon-transform:              rotate(-180deg);

$accordion-button-icon:         url("data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg' viewBox='0 0 16
16' fill='#{$accordion-icon-color}'><path fill-rule='evenodd' d='M1.646 4.646a.5.5 0 0 1 .708 0L8 10.293l5.646-
5.647a.5.5 0 0 1 .708.708l-6 6a.5.5 0 0 1-.708 0l-6-6a.5.5 0 0 1 0-.708z'/></svg>");
$accordion-button-active-icon:  url("data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg' viewBox='0 0
16 16' fill='#{$accordion-icon-active-color}'><path fill-rule='evenodd' d='M1.646 4.646a.5.5 0 0 1 .708 0L8
10.293l5.646-5.647a.5.5 0 0 1 .708.708l-6 6a.5.5 0 0 1-.708 0l-6-6a.5.5 0 0 1 0-.708z'/></svg>");
```

Alerts

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

Examples

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

A simple primary alert—check it out!

A simple secondary alert—check it out!

A simple success alert—check it out!

A simple danger alert—check it out!

A simple warning alert—check it out!

A simple info alert—check it out!

A simple light alert—check it out!

A simple dark alert—check it out!

```
<div class="alert alert-primary" role="alert">
  A simple primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert">
  A simple secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  A simple success alert—check it out!
</div>
<div class="alert alert-danger" role="alert">
  A simple danger alert—check it out!
</div>
<div class="alert alert-warning" role="alert">
  A simple warning alert—check it out!
</div>
<div class="alert alert-info" role="alert">
  A simple info alert—check it out!
</div>
<div class="alert alert-light" role="alert">
  A simple light alert—check it out!
</div>
```

```
<div class="alert alert-dark" role="alert">
  A simple dark alert—check it out!
</div>
```

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (e.g. the visible text), or is included through alternative means, such as additional text hidden with the .visually-hidden class.

Link color

Use the .alert-link utility class to quickly provide matching colored links within any alert.

A simple primary alert with **an example link**. Give it a click if you like.

A simple secondary alert with **an example link**. Give it a click if you like.

A simple success alert with **an example link**. Give it a click if you like.

A simple danger alert with **an example link**. Give it a click if you like.

A simple warning alert with **an example link**. Give it a click if you like.

A simple info alert with **an example link**. Give it a click if you like.

A simple light alert with **an example link**. Give it a click if you like.

A simple dark alert with **an example link**. Give it a click if you like.

```
<div class="alert alert-primary" role="alert">
  A simple primary alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
</div>
<div class="alert alert-secondary" role="alert">
  A simple secondary alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
</div>
<div class="alert alert-success" role="alert">
  A simple success alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
</div>
<div class="alert alert-danger" role="alert">
  A simple danger alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
</div>
<div class="alert alert-warning" role="alert">
  A simple warning alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
</div>
<div class="alert alert-info" role="alert">
  A simple info alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
</div>
<div class="alert alert-light" role="alert">
```

```
  A simple light alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
</div>
<div class="alert alert-dark" role="alert">
  A simple dark alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
</div>
```

Additional content

Alerts can also contain additional HTML elements like headings, paragraphs and dividers.

Well done!

Aww yeah, you successfully read this important alert message. This example text is going to run a bit longer so that you can see how spacing within an alert works with this kind of content.

Whenever you need to, be sure to use margin utilities to keep things nice and tidy.

```
<div class="alert alert-success" role="alert">
  <h4 class="alert-heading">Well done!</h4>
  <p>Aww yeah, you successfully read this important alert message. This example text is going to run a bit longer so that you can see how spacing within an alert works with this kind of content.</p>
  <hr>
  <p class="mb-0">Whenever you need to, be sure to use margin utilities to keep things nice and tidy.</p>
</div>
```

Icons

Similarly, you can use flexbox utilities and Bootstrap Icons to create alerts with icons. Depending on your icons and content, you may want to add more utilities or custom styles.

An example alert with an icon

```
<div class="alert alert-primary d-flex align-items-center" role="alert">
  <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" fill="currentColor" class="bi bi-exclamation-triangle-fill flex-shrink-0 me-2" viewBox="0 0 16 16" role="img" aria-label="Warning:">
    <path d="M8.982 1.566a1.13 1.13 0 0 0-1.96 0L.165 13.233c-.457.778.091 1.767.98 1.767h13.713c.889 0 1.438-.99.98-1.767L8.982 1.566zM8 5c.535 0 .954.462.9.995l-.35 3.507a.552.552 0 0 1-1.1 0L7.1 5.995A.905.905 0 0 1 8 5zm.002 6a1 1 0 1 1 0 2 1 1 0 0 1 0-2z"/>
  </svg>
  <div>
    An example alert with an icon
  </div>
</div>
```

Need more than one icon for your alerts? Consider using more Bootstrap Icons and making a local SVG sprite like so to easily reference the same icons repeatedly.

An example alert with an icon

An example success alert with an icon

An example warning alert with an icon

An example danger alert with an icon

```
<svg xmlns="http://www.w3.org/2000/svg" style="display: none;">
  <symbol id="check-circle-fill" fill="currentColor" viewBox="0 0 16 16">
    <path d="M16 8A8 8 0 1 1 0 8a8 8 0 0 1 16 0zm-3.97-3.03a.75.75 0 0 0-1.08.022L7.477 9.417 5.384 7.323a.75.75 0 0 0-1.06 1.06L6.97 11.03a.75.75 0 0 0 1.079-.02l3.992-4.99a.75.75 0 0 0-.01-1.05z"/>
  </symbol>
  <symbol id="info-fill" fill="currentColor" viewBox="0 0 16 16">
    <path d="M8 16A8 8 0 1 0 8 0a8 8 0 0 0 0 16zm.93-9.412-1 4.705c-.07.34.029.533.304.533.194 0 .487-.07.686-.246l-.088.416c-.287.346-.92.598-1.465.598-.703 0-1.002-.422-.808-1.319l.738-3.468c.064-.293.006-.399-.287-.47l-.451-.081.082-.381 2.29-.287zM8 5.5a1 1 0 1 1 0-2 1 1 0 0 1 0 2z"/>
  </symbol>
  <symbol id="exclamation-triangle-fill" fill="currentColor" viewBox="0 0 16 16">
    <path d="M8.982 1.566a1.13 1.13 0 0 0-1.96 0L.165 13.233c-.457.778.091 1.767.98 1.767h13.713c.889 0 1.438-.99.98-1.767L8.982 1.566zM8 5c.535 0 .954.462.9.995l-.35 3.507a.552.552 0 0 1-1.1 0L7.1 5.995A.905.905 0 0 1 8 5zm.002 6a1 1 0 1 1 0 2 1 1 0 0 1 0-2z"/>
  </symbol>
</svg>

<div class="alert alert-primary d-flex align-items-center" role="alert">
  <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-label="Info:"><use xlink:href="#info-fill"/></svg>
  <div>
    An example alert with an icon
  </div>
</div>
<div class="alert alert-success d-flex align-items-center" role="alert">
  <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-label="Success:"><use xlink:href="#check-circle-fill"/></svg>
  <div>
    An example success alert with an icon
  </div>
</div>
<div class="alert alert-warning d-flex align-items-center" role="alert">
  <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-label="Warning:"><use xlink:href="#exclamation-triangle-fill"/></svg>
  <div>
    An example warning alert with an icon
  </div>
</div>
<div class="alert alert-danger d-flex align-items-center" role="alert">
```

```
  <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-label="Danger:"><use
xlink:href="#exclamation-triangle-fill"/></svg>
  <div>
    An example danger alert with an icon
  </div>
</div>
```

Dismissing

Using the alert JavaScript plugin, it's possible to dismiss any alert inline. Here's how:

- Be sure you've loaded the alert plugin, or the compiled Bootstrap JavaScript.
- Add a close button and the `.alert-dismissible` class, which adds extra padding to the right of the alert and positions the close button.
- On the close button, add the `data-bs-dismiss="alert"` attribute, which triggers the JavaScript functionality. Be sure to use the `<button>` element with it for proper behavior across all devices.
- To animate alerts when dismissing them, be sure to add the `.fade` and `.show` classes.

You can see this in action with a live demo:

Holy guacamole! You should check in on some of those fields below.

```
<div class="alert alert-warning alert-dismissible fade show" role="alert">
  <strong>Holy guacamole!</strong> You should check in on some of those fields below.
  <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
</div>
```

When an alert is dismissed, the element is completely removed from the page structure. If a keyboard user dismisses the alert using the close button, their focus will suddenly be lost and, depending on the browser, reset to the start of the page/document. For this reason, we recommend including additional JavaScript that listens for the `closed.bs.alert` event and programmatically sets `focus()` to the most appropriate location in the page. If you're planning to move focus to a non-interactive element that normally does not receive focus, make sure to add `tabindex="-1"` to the element.

Sass

Variables

```
$alert-padding-y:           $spacer;
$alert-padding-x:           $spacer;
$alert-margin-bottom:         1rem;
```

```
$alert-border-radius:          $border-radius;
$alert-link-font-weight:       $font-weight-bold;
$alert-border-width:           $border-width;
$alert-bg-scale:              -80%;
$alert-border-scale:          -70%;
$alert-color-scale:            40%;
$alert-dismissible-padding-r:  $alert-padding-x * 3; // 3x covers width of x plus default padding on either side
```

Variant mixin

Used in combination with `$theme-colors` to create contextual modifier classes for our alerts.

```
@mixin alert-variant($background, $border, $color) {
  color: $color;
  @include gradient-bg($background);
  border-color: $border;

  .alert-link {
    color: shade-color($color, 20%);
  }
}
```

Loop

Loop that generates the modifier classes with the `alert-variant()` mixin.

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

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

JavaScript behavior

Triggers

Enable dismissal of an alert via JavaScript:

```
var alertList = document.querySelectorAll('.alert')
alertList.forEach(function (alert) {
  new bootstrap.Alert(alert)
})
```

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

```
<button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
```

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

Methods

You can create an alert instance with the alert constructor, for example:

```
var myAlert = document.getElementById('myAlert')
var bsAlert = new bootstrap.Alert(myAlert)
```

This makes an alert listen for click events on descendant elements which have the `data-bs-dismiss="alert"` attribute. (Not necessary when using the data-api's auto-initialization.)

Method	Description
`close`	Closes an alert by removing it from the DOM. If the `.fade` and `.show` classes are present on the elemen will fade out before it is removed.
`dispose`	Destroys an element's alert. (Removes stored data on the DOM element)
`getInstance`	Static method which allows you to get the alert instance associated to a DOM element, you can use it this: `bootstrap.Alert.getInstance(alert)`
`getOrCreateInstance`	Static method which returns an alert instance associated to a DOM element or create a new one in ca initialised. You can use it like this: `bootstrap.Alert.getOrCreateInstance(element)`

```
var alertNode = document.querySelector('.alert')
var alert = bootstrap.Alert.getInstance(alertNode)
alert.close()
```

Events

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

Event	Description
close.bs.alert	Fires immediately when the close instance method is called.
closed.bs.alert	Fired when the alert has been closed and CSS transitions have completed.

```
var myAlert = document.getElementById('myAlert')
myAlert.addEventListener('closed.bs.alert', function () {
  // do something, for instance, explicitly move focus to the most appropriate element,
  // so it doesn't get lost/reset to the start of the page
  // document.getElementById('...').focus()
})
```

Badges

Documentation and examples for badges, our small count and labeling component.

Examples

Badges scale to match the size of the immediate parent element by using relative font sizing and `em` units. As of v5, badges no longer have focus or hover styles for links.

Headings

Example heading

Example heading

Example heading

Example heading
Example heading
Example heading

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

Buttons

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

Notifications

```
<button type="button" class="btn btn-primary">
  Notifications <span class="badge bg-secondary">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.

Positioned

Use utilities to modify a .badge and position it in the corner of a link or button.

Inbox

```
<button type="button" class="btn btn-primary position-relative">
  Inbox
  <span class="position-absolute top-0 start-100 translate-middle badge rounded-pill bg-danger">
    99+
    <span class="visually-hidden">unread messages</span>
  </span>
</button>
```

You can also replace the .badge class with a few more utilities without a count for a more generic indicator.

ProfileNew alerts

```
<button type="button" class="btn btn-primary position-relative">
  Profile
  <span class="position-absolute top-0 start-100 translate-middle p-2 bg-danger border border-light rounded-circle">
    <span class="visually-hidden">New alerts</span>
  </span>
</button>
```

Background colors

Use our background utility classes to quickly change the appearance of a badge. Please note that when using Bootstrap’s default .bg-light, you’ll likely need a text color utility like .text-dark for proper styling. This is because background utilities do not set anything but background-color.

Warning Info Light

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

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (e.g. the visible text), or is included through alternative means, such as additional text hidden with the .visually-hidden class.

Pill badges

Use the .rounded-pill utility class to make badges more rounded with a larger border-radius.

Warning Info Light

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

Sass

Variables

```
$badge-font-size:                   .75em;
$badge-font-weight:                   $font-weight-bold;
$badge-color:                     $white;
$badge-padding-y:                     .35em;
$badge-padding-x:                     .65em;
$badge-border-radius:                  $border-radius;
```

Breadcrumb

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

Example

Use an ordered or unordered list with linked list items to create a minimally styled breadcrumb. Use our utilities to add additional styles as desired.

1. Home

1. Home
2. Library

1. Home
2. Library
3. Data

```
<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item active" aria-current="page">Home</li>
  </ol>
</nav>

<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Library</li>
  </ol>
</nav>

<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item"><a href="#">Library</a></li>
    <li class="breadcrumb-item active" aria-current="page">Data</li>
  </ol>
</nav>
```

Dividers

Dividers are automatically added in CSS through ::before and content. They can be changed by modifying a local CSS custom property `--bs-breadcrumb-divider`, or through the $breadcrumb-divider Sass variable — and $breadcrumb-divider-flipped for its RTL counterpart, if needed. We default to our Sass variable, which is set as a fallback to the custom property. This way, you get a global divider that you can override without recompiling CSS at any time.

1. Home
2. Library

```
<nav style="--bs-breadcrumb-divider: '>';" aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Library</li>
  </ol>
</nav>
```

When modifying via Sass, the quote function is required to generate the quotes around a string. For example, using > as the divider, you can use this:

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

It's also possible to use an **embedded SVG icon**. Apply it via our CSS custom property, or use the Sass variable.

1. Home
2. Library

```
<nav style="--bs-breadcrumb-divider: url(&#34;data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg'
width='8' height='8'%3E%3Cpath d='M2.5 0L1 1.5 3.5 4 1 6.5 2.5 8l4-4-4-4z'
fill='currentColor'/%3E%3C/svg%3E&#34;);" aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Library</li>
  </ol>
</nav>
```

```
$breadcrumb-divider: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='8'
height='8'%3E%3Cpath d='M2.5 0L1 1.5 3.5 4 1 6.5 2.5 8l4-4-4-4z' fill='currentColor'/%3E%3C/svg%3E");
```

You can also remove the divider setting --bs-breadcrumb-divider: ''; (empty strings in CSS custom properties counts as a value), or setting the Sass variable to $breadcrumb-divider: none;.

1. Home
2. Library

```
<nav style="--bs-breadcrumb-divider: ';'" aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Library</li>
  </ol>
</nav>
```

```
$breadcrumb-divider: none;
```

Accessibility

Since breadcrumbs provide a navigation, it's a good idea to add a meaningful label such as `aria-label="breadcrumb"` to describe the type of navigation provided in the `<nav>` element, as well as applying an `aria-current="page"` to the last item of the set to indicate that it represents the current page.

For more information, see the [WAI-ARIA Authoring Practices for the breadcrumb pattern].

Sass

Variables

```
$breadcrumb-font-size:              null;
$breadcrumb-padding-y:              0;
$breadcrumb-padding-x:              0;
$breadcrumb-item-padding-x:         .5rem;
$breadcrumb-margin-bottom:          1rem;
$breadcrumb-bg:                     null;
$breadcrumb-divider-color:          $gray-600;
$breadcrumb-active-color:           $gray-600;
$breadcrumb-divider:                quote("/");
$breadcrumb-divider-flipped:        $breadcrumb-divider;
$breadcrumb-border-radius:          null;
```

Buttons

Use Bootstrap's custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more.

Examples

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

Primary Secondary Success Danger Warning Info Light Dark Link

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

<button type="button" class="btn btn-link">Link</button>
```

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (e.g. the visible text), or is included through alternative means, such as additional text hidden with the `.visually-hidden` class.

Disable text wrapping

If you don't want the button text to wrap, you can add the `.text-nowrap` class to the button. In Sass, you can set `$btn-white-space: nowrap` to disable text wrapping for each button.

Button tags

The `.btn` classes are designed to be used with the `<button>` element. However, you can also use these classes on `<a>` or `<input>` elements (though some browsers may apply a slightly different rendering).

When using button classes on <a> elements that are used to trigger in-page functionality (like collapsing content), rather than linking to new pages or sections within the current page, these links should be given a role="button" to appropriately convey their purpose to assistive technologies such as screen readers.

Link Button Submit Reset

```
<a class="btn btn-primary" href="#" role="button">Link</a>
<button class="btn btn-primary" type="submit">Button</button>
<input class="btn btn-primary" type="button" value="Input">
<input class="btn btn-primary" type="submit" value="Submit">
<input class="btn btn-primary" type="reset" value="Reset">
```

Outline buttons

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

Primary Secondary Success Danger Warning Info Light Dark

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

Some of the button styles use a relatively light foreground color, and should only be used on a dark background in order to have sufficient contrast.

Sizes

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

Large button Large button

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

Small button Small button

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

Disabled state

Make buttons look inactive by adding the `disabled` boolean attribute to any `<button>` element. Disabled buttons have `pointer-events: none` applied to, preventing hover and active states from triggering.

Primary button Button

```
<button type="button" class="btn btn-lg btn-primary" disabled>Primary button</button>
<button type="button" class="btn btn-secondary btn-lg" disabled>Button</button>
```

Disabled buttons using the `<a>` element behave a bit different:

- `<a>`s don't support the `disabled` attribute, so you must add the `.disabled` class to make it visually appear disabled.
- Some future-friendly styles are included to disable all `pointer-events` on anchor buttons.
- Disabled buttons should include the `aria-disabled="true"` attribute to indicate the state of the element to assistive technologies.

Primary link Link

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

Link functionality caveat

The `.disabled` class uses `pointer-events: none` to try to disable the link functionality of `<a>`s, but that CSS property is not yet standardized. In addition, even in browsers that do support `pointer-events: none`, keyboard navigation remains unaffected, meaning that sighted keyboard users and users of assistive technologies will still be able to activate these links. So to be safe, in addition to `aria-disabled="true"`, also include a `tabindex="-1"` attribute on these links to prevent them from receiving keyboard focus, and use custom JavaScript to disable their functionality altogether.

Block buttons

Create responsive stacks of full-width, “block buttons” like those in Bootstrap 4 with a mix of our display and gap utilities. By using utilities instead of button specific classes, we have much greater control over spacing, alignment, and responsive behaviors.

ButtonButton

```
<div class="d-grid gap-2">
  <button class="btn btn-primary" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

Here we create a responsive variation, starting with vertically stacked buttons until the `md` breakpoint, where `.d-md-block` replaces the `.d-grid` class, thus nullifying the `gap-2` utility. Resize your browser to see them change.

Button Button

```
<div class="d-grid gap-2 d-md-block">
  <button class="btn btn-primary" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

You can adjust the width of your block buttons with grid column width classes. For example, for a half-width “block button”, use `.col-6`. Center it horizontally with `.mx-auto`, too.

ButtonButton

```
<div class="d-grid gap-2 col-6 mx-auto">
  <button class="btn btn-primary" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

Additional utilities can be used to adjust the alignment of buttons when horizontal. Here we’ve taken our previous responsive example and added some flex utilities and a margin utility on the button to right align the buttons when they’re no longer stacked.

ButtonButton

```
<div class="d-grid gap-2 d-md-flex justify-content-md-end">
  <button class="btn btn-primary me-md-2" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

Button plugin

The button plugin allows you to create simple on/off toggle buttons.

Visually, these toggle buttons are identical to the checkbox toggle buttons. However, they are conveyed differently by assistive technologies: the checkbox toggles will be announced by screen readers as "checked"/"not checked" (since, despite their appearance, they are fundamentally still checkboxes), whereas these toggle buttons will be announced as "button"/"button pressed". The choice between these two approaches will depend on the type of toggle you are creating, and whether or not the toggle will make sense to users when announced as a checkbox or as an actual button.

Toggle states

Add `data-bs-toggle="button"` to toggle a button's `active` state. If you're pre-toggling a button, you must manually add the `.active` class **and** `aria-pressed="true"` to ensure that it is conveyed appropriately to assistive technologies.

Toggle button Active toggle button Disabled toggle button

```
<button type="button" class="btn btn-primary" data-bs-toggle="button" autocomplete="off">Toggle
button</button>
<button type="button" class="btn btn-primary active" data-bs-toggle="button" autocomplete="off" aria-
pressed="true">Active toggle button</button>
<button type="button" class="btn btn-primary" disabled data-bs-toggle="button" autocomplete="off">Disabled
toggle button</button>
```

Toggle link Active toggle link Disabled toggle link

```
<a href="#" class="btn btn-primary" role="button" data-bs-toggle="button">Toggle link</a>
<a href="#" class="btn btn-primary active" role="button" data-bs-toggle="button" aria-pressed="true">Active
toggle link</a>
<a href="#" class="btn btn-primary disabled" tabindex="-1" aria-disabled="true" role="button" data-bs-
toggle="button">Disabled toggle link</a>
```

Methods

You can create a button instance with the button constructor, for example:

```
var button = document.getElementById('myButton')
var bsButton = new bootstrap.Button(button)
```

Method	Description
toggle	Toggles push state. Gives the button the appearance that it has been activated.
dispose	Destroys an element's button. (Removes stored data on the DOM element)
getInstance	Static method which allows you to get the button instance associated to a DOM element, you can use this: bootstrap.Button.getInstance(element)
getOrCreateInstance	Static method which returns a button instance associated to a DOM element or create a new one in c initialised. You can use it like this: bootstrap.Button.getOrCreateInstance(element)

For example, to toggle all buttons

```
var buttons = document.querySelectorAll('.btn')
buttons.forEach(function (button) {
  var button = new bootstrap.Button(button)
  button.toggle()
})
```

Sass

Variables

```
$btn-padding-y:              $input-btn-padding-y;
$btn-padding-x:              $input-btn-padding-x;
$btn-font-family:            $input-btn-font-family;
$btn-font-size:              $input-btn-font-size;
$btn-line-height:            $input-btn-line-height;
$btn-white-space:            null; // Set to `nowrap` to prevent text wrapping

$btn-padding-y-sm:           $input-btn-padding-y-sm;
$btn-padding-x-sm:           $input-btn-padding-x-sm;
$btn-font-size-sm:           $input-btn-font-size-sm;

$btn-padding-y-lg:           $input-btn-padding-y-lg;
$btn-padding-x-lg:           $input-btn-padding-x-lg;
$btn-font-size-lg:           $input-btn-font-size-lg;

$btn-border-width:           $input-btn-border-width;

$btn-font-weight:            $font-weight-normal;
$btn-box-shadow:             inset 0 1px 0 rgba($white, .15), 0 1px 1px rgba($black, .075);
$btn-focus-width:            $input-btn-focus-width;
$btn-focus-box-shadow:       $input-btn-focus-box-shadow;
$btn-disabled-opacity:       .65;
```

```
$btn-active-box-shadow:       inset 0 3px 5px rgba($black, .125);

$btn-link-color:              $link-color;
$btn-link-hover-color:        $link-hover-color;
$btn-link-disabled-color:     $gray-600;

// Allows for customizing button radius independently from global border radius
$btn-border-radius:           $border-radius;
$btn-border-radius-sm:        $border-radius-sm;
$btn-border-radius-lg:        $border-radius-lg;

$btn-transition:              color .15s ease-in-out, background-color .15s ease-in-out, border-color .15s ease-in-out,
box-shadow .15s ease-in-out;

$btn-hover-bg-shade-amount:       15%;
$btn-hover-bg-tint-amount:        15%;
$btn-hover-border-shade-amount:   20%;
$btn-hover-border-tint-amount:    10%;
$btn-active-bg-shade-amount:      20%;
$btn-active-bg-tint-amount:       20%;
$btn-active-border-shade-amount:  25%;
$btn-active-border-tint-amount:   10%;
```

Mixins

There are three mixins for buttons: button and button outline variant mixins (both based on $theme-colors), plus a button size mixin.

```
@mixin button-variant(
  $background,
  $border,
  $color: color-contrast($background),
  $hover-background: if($color == $color-contrast-light, shade-color($background, $btn-hover-bg-shade-amount),
tint-color($background, $btn-hover-bg-tint-amount)),
  $hover-border: if($color == $color-contrast-light, shade-color($border, $btn-hover-border-shade-amount), tint-
color($border, $btn-hover-border-tint-amount)),
  $hover-color: color-contrast($hover-background),
  $active-background: if($color == $color-contrast-light, shade-color($background, $btn-active-bg-shade-amount),
tint-color($background, $btn-active-bg-tint-amount)),
  $active-border: if($color == $color-contrast-light, shade-color($border, $btn-active-border-shade-amount), tint-
color($border, $btn-active-border-tint-amount)),
  $active-color: color-contrast($active-background),
  $disabled-background: $background,
  $disabled-border: $border,
  $disabled-color: color-contrast($disabled-background)
) {
  color: $color;
  @include gradient-bg($background);
  border-color: $border;
  @include box-shadow($btn-box-shadow);

  &:hover {
    color: $hover-color;
    @include gradient-bg($hover-background);
    border-color: $hover-border;
  }

  .btn-check:focus + &,
  &:focus {
    color: $hover-color;
    @include gradient-bg($hover-background);
    border-color: $hover-border;
    @if $enable-shadows {
      @include box-shadow($btn-box-shadow, 0 0 0 $btn-focus-width rgba(mix($color, $border, 15%), .5));
    } @else {
      // Avoid using mixin so we can pass custom focus shadow properly
      box-shadow: 0 0 0 $btn-focus-width rgba(mix($color, $border, 15%), .5);
    }
  }

  .btn-check:checked + &,
  .btn-check:active + &,
  &:active,
  &.active,
  .show > &.dropdown-toggle {
    color: $active-color;
    background-color: $active-background;
    // Remove CSS gradients if they're enabled
    background-image: if($enable-gradients, none, null);
    border-color: $active-border;

    &:focus {
      @if $enable-shadows {
        @include box-shadow($btn-active-box-shadow, 0 0 0 $btn-focus-width rgba(mix($color, $border, 15%), .5));
      } @else {
        // Avoid using mixin so we can pass custom focus shadow properly
        box-shadow: 0 0 0 $btn-focus-width rgba(mix($color, $border, 15%), .5);
      }
    }
  }

  &:disabled,
  &.disabled {
    color: $disabled-color;
    background-color: $disabled-background;
    // Remove CSS gradients if they're enabled
    background-image: if($enable-gradients, none, null);
    border-color: $disabled-border;
  }
}


@mixin button-outline-variant(
  $color,
  $color-hover: color-contrast($color),
  $active-background: $color,
  $active-border: $color,
  $active-color: color-contrast($active-background)
```

```
) {
  color: $color;
  border-color: $color;

  &:hover {
    color: $color-hover;
    background-color: $active-background;
    border-color: $active-border;
  }

  .btn-check:focus + &,
  &:focus {
    box-shadow: 0 0 0 $btn-focus-width rgba($color, .5);
  }

  .btn-check:checked + &,
  .btn-check:active + &,
  &:active,
  &.active,
  &.dropdown-toggle.show {
    color: $active-color;
    background-color: $active-background;
    border-color: $active-border;

    &:focus {
      @if $enable-shadows {
        @include box-shadow($btn-active-box-shadow, 0 0 0 $btn-focus-width rgba($color, .5));
      } @else {
        // Avoid using mixin so we can pass custom focus shadow properly
        box-shadow: 0 0 0 $btn-focus-width rgba($color, .5);
      }
    }
  }

  &:disabled,
  &.disabled {
    color: $color;
    background-color: transparent;
  }
}


@mixin button-size($padding-y, $padding-x, $font-size, $border-radius) {
  padding: $padding-y $padding-x;
  @include font-size($font-size);
  // Manually declare to provide an override to the browser default
  @include border-radius($border-radius, 0);
}
```

Loops

Button variants (for regular and outline buttons) use their respective mixins with our `$theme-colors` map to generate the modifier classes in `scss/_buttons.scss`.

```scss
@each $color, $value in $theme-colors {
  .btn-#{$color} {
    @include button-variant($value, $value);
  }
}

@each $color, $value in $theme-colors {
  .btn-outline-#{$color} {
    @include button-outline-variant($value);
  }
}
```

Button group

Group a series of buttons together on a single line or stack them in a vertical column.

Basic example

Wrap a series of buttons with .btn in .btn-group.

LeftMiddleRight

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

Ensure correct role and provide a label

In order for assistive technologies (such as screen readers) to convey that a series of buttons is grouped, an appropriate role attribute needs to be provided. For button groups, this would be role="group", while toolbars should have a role="toolbar".

In addition, groups and toolbars should be given an explicit label, as most assistive technologies will otherwise not announce them, despite the presence of the correct role attribute. In the examples provided here, we use aria-label, but alternatives such as aria-labelledby can also be used.

These classes can also be added to groups of links, as an alternative to the .nav navigation components.

Active link LinkLink

```
<div class="btn-group">
  <a href="#" class="btn btn-primary active" aria-current="page">Active link</a>
  <a href="#" class="btn btn-primary">Link</a>
  <a href="#" class="btn btn-primary">Link</a>
</div>
```

Mixed styles

LeftMiddleRight

```
<div class="btn-group" role="group" aria-label="Basic mixed styles example">
  <button type="button" class="btn btn-danger">Left</button>
  <button type="button" class="btn btn-warning">Middle</button>
  <button type="button" class="btn btn-success">Right</button>
</div>
```

Outlined styles

LeftMiddleRight

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

Checkbox and radio button groups

Combine button-like checkbox and radio toggle buttons into a seamless looking button group.

☐ Checkbox 1☐ Checkbox 2☐ Checkbox 3

```
<div class="btn-group" role="group" aria-label="Basic checkbox toggle button group">
  <input type="checkbox" class="btn-check" id="btncheck1" autocomplete="off">
  <label class="btn btn-outline-primary" for="btncheck1">Checkbox 1</label>

  <input type="checkbox" class="btn-check" id="btncheck2" autocomplete="off">
  <label class="btn btn-outline-primary" for="btncheck2">Checkbox 2</label>

  <input type="checkbox" class="btn-check" id="btncheck3" autocomplete="off">
  <label class="btn btn-outline-primary" for="btncheck3">Checkbox 3</label>
</div>
```

◉ Radio 1○ Radio 2○ Radio 3

```
<div class="btn-group" role="group" aria-label="Basic radio toggle button group">
  <input type="radio" class="btn-check" name="btnradio" id="btnradio1" autocomplete="off" checked>
  <label class="btn btn-outline-primary" for="btnradio1">Radio 1</label>

  <input type="radio" class="btn-check" name="btnradio" id="btnradio2" autocomplete="off">
  <label class="btn btn-outline-primary" for="btnradio2">Radio 2</label>

  <input type="radio" class="btn-check" name="btnradio" id="btnradio3" autocomplete="off">
  <label class="btn btn-outline-primary" for="btnradio3">Radio 3</label>
</div>
```

Button toolbar

Combine sets of button groups into button toolbars for more complex components. Use utility classes as needed to space out groups, buttons, and more.

1234

567

8

```
<div class="btn-toolbar" role="toolbar" aria-label="Toolbar with button groups">
  <div class="btn-group me-2" role="group" aria-label="First group">
    <button type="button" class="btn btn-primary">1</button>
    <button type="button" class="btn btn-primary">2</button>
    <button type="button" class="btn btn-primary">3</button>
    <button type="button" class="btn btn-primary">4</button>
  </div>
  <div class="btn-group me-2" role="group" aria-label="Second group">
    <button type="button" class="btn btn-secondary">5</button>
    <button type="button" class="btn btn-secondary">6</button>
    <button type="button" class="btn btn-secondary">7</button>
  </div>
  <div class="btn-group" role="group" aria-label="Third group">
    <button type="button" class="btn btn-info">8</button>
  </div>
</div>
```

Feel free to mix input groups with button groups in your toolbars. Similar to the example above, you'll likely need some utilities though to space things properly.

1234

@

1234

@

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

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

Sizing

Instead of applying button sizing classes to every button in a group, just add .btn-group-* to each .btn-group, including each one when nesting multiple groups.

LeftMiddleRight

LeftMiddleRight

LeftMiddleRight

```
<div class="btn-group btn-group-lg" role="group" aria-label="...">...</div>
<div class="btn-group" role="group" aria-label="...">...</div>
<div class="btn-group btn-group-sm" role="group" aria-label="...">...</div>
```

Nesting

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

12

Dropdown

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

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

Vertical variation

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

ButtonButtonButtonButtonButtonButton

ButtonButton

Dropdown

ButtonButton

Dropdown

Dropdown

Dropdown

● Radio 1 ○ Radio 2 ○ Radio 3

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

Cards

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

About

A **card** is a flexible and extensible content container. It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options. If you're familiar with Bootstrap 3, cards replace our old panels, wells, and thumbnails. Similar functionality to those components is available as modifier classes for cards.

Example

Cards are built with as little markup and styles as possible, but still manage to deliver a ton of control and customization. Built with flexbox, they offer easy alignment and mix well with other Bootstrap components. They have no `margin` by default, so use spacing utilities as needed.

Below is an example of a basic card with mixed content and a fixed width. Cards have no fixed width to start, so they'll naturally fill the full width of its parent element. This is easily customized with our various sizing options.

Image cap

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

Go somewhere

```
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Content types

Cards support a wide variety of content, including images, text, list groups, links, and more. Below are examples of what's supported.

Body

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

This is some text within a card body.

```
<div class="card">
  <div class="card-body">
    This is some text within a card body.
  </div>
</div>
```

Titles, text, and links

Card titles are used by adding .card-title to a <h*> tag. In the same way, links are added and placed next to each other by adding .card-link to an <a> tag.

Subtitles are used by adding a .card-subtitle to a <h*> tag. If the .card-title and the .card-subtitle items are placed in a .card-body item, the card title and subtitle are aligned nicely.

Card title

Card subtitle

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

Card link Another link

```
<div class="card" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <h6 class="card-subtitle mb-2 text-muted">Card subtitle</h6>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
    <a href="#" class="card-link">Card link</a>
    <a href="#" class="card-link">Another link</a>
  </div>
</div>
```

Images

.card-img-top places an image to the top of the card. With .card-text, text can be added to the card. Text within .card-text can also be styled with the standard HTML tags.

Image cap

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

```
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
</div>
```

List groups

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

- An item
- A second item
- A third item

```
<div class="card" style="width: 18rem;">
  <ul class="list-group list-group-flush">
    <li class="list-group-item">An item</li>
    <li class="list-group-item">A second item</li>
    <li class="list-group-item">A third item</li>
  </ul>
</div>
```

Featured

- An item
- A second item
- A third item

```
<div class="card" style="width: 18rem;">
  <div class="card-header">
    Featured
  </div>
  <ul class="list-group list-group-flush">
    <li class="list-group-item">An item</li>
    <li class="list-group-item">A second item</li>
    <li class="list-group-item">A third item</li>
  </ul>
</div>
```

- An item
- A second item
- A third item

Card footer

```
<div class="card" style="width: 18rem;">
  <ul class="list-group list-group-flush">
    <li class="list-group-item">An item</li>
    <li class="list-group-item">A second item</li>
    <li class="list-group-item">A third item</li>
  </ul>
  <div class="card-footer">
    Card footer
  </div>
</div>
```

Kitchen sink

Mix and match multiple content types to create the card you need, or throw everything in there. Shown below are image styles, blocks, text styles, and a list group—all wrapped in a fixed-width card.

Image cap

Card title

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

- An item
- A second item
- A third item

Card link Another link

```
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
  <ul class="list-group list-group-flush">
    <li class="list-group-item">An item</li>
    <li class="list-group-item">A second item</li>
    <li class="list-group-item">A third item</li>
  </ul>
  <div class="card-body">
    <a href="#" class="card-link">Card link</a>
    <a href="#" class="card-link">Another link</a>
  </div>
</div>
```

Header and footer

Add an optional header and/or footer within a card.

Featured

Special title treatment

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

Go somewhere

```
<div class="card">
  <div class="card-header">
    Featured
  </div>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Card headers can be styled by adding `.card-header` to `<h*>` elements.

Featured

Special title treatment

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

Go somewhere

```
<div class="card">
  <h5 class="card-header">Featured</h5>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
```

```
</div>
```
Quote

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

Someone famous in *Source Title*

```
<div class="card">
  <div class="card-header">
    Quote
  </div>
  <div class="card-body">
    <blockquote class="blockquote mb-0">
      <p>A well-known quote, contained in a blockquote element.</p>
      <footer class="blockquote-footer">Someone famous in <cite title="Source Title">Source Title</cite></footer>
    </blockquote>
  </div>
</div>
```
Featured

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

Go somewhere

2 days ago

```
<div class="card text-center">
  <div class="card-header">
    Featured
  </div>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
  <div class="card-footer text-muted">
    2 days ago
  </div>
</div>
```

Sizing

Cards assume no specific `width` to start, so they'll be 100% wide unless otherwise stated. You can change this as needed with custom CSS, grid classes, grid Sass mixins, or utilities.

Using grid markup

Using the grid, wrap cards in columns and rows as needed.

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

Go somewhere

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

Go somewhere

```
<div class="row">
  <div class="col-sm-6">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Special title treatment</h5>
        <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
    </div>
  </div>
  <div class="col-sm-6">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Special title treatment</h5>
        <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
    </div>
  </div>
</div>
```

Using utilities

Use our handful of available sizing utilities to quickly set a card's width.

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

Button

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

Button

```
<div class="card w-75">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Button</a>
  </div>
</div>

<div class="card w-50">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Button</a>
  </div>
</div>
```

Using custom CSS

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

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

Go somewhere

```
<div class="card" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Text alignment

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

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

Go somewhere

Special title treatment

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

Go somewhere

Special title treatment

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

Go somewhere

```
<div class="card" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>

<div class="card text-center" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>

<div class="card text-end" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Navigation

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

- Active
- Link
- Disabled

Special title treatment

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

Go somewhere

```
<div class="card text-center">
  <div class="card-header">
    <ul class="nav nav-tabs card-header-tabs">
      <li class="nav-item">
        <a class="nav-link active" aria-current="true" href="#">Active</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
  </div>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

- Active
- Link
- Disabled

Special title treatment

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

Go somewhere

```
<div class="card text-center">
  <div class="card-header">
    <ul class="nav nav-pills card-header-pills">
      <li class="nav-item">
        <a class="nav-link active" href="#">Active</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
  </div>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
```

```
</div>
```

Images

Cards include a few options for working with images. Choose from appending "image caps" at either end of a card, overlaying images with card content, or simply embedding the image in a card.

Image caps

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

Image cap

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

Image cap

```
<div class="card mb-3">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">This is a wider card with supporting text below as a natural lead-in to additional content. 
This content is a little bit longer.</p>
    <p class="card-text"><small class="text-muted">Last updated 3 mins ago</small></p>
  </div>
</div>
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">This is a wider card with supporting text below as a natural lead-in to additional content. 
This content is a little bit longer.</p>
    <p class="card-text"><small class="text-muted">Last updated 3 mins ago</small></p>
  </div>
  <img src="..." class="card-img-bottom" alt="...">
</div>
```

Image overlays

Turn an image into a card background and overlay your card's text. Depending on the image, you may or may not need additional styles or utilities.

Card image

Card title
This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

```
<div class="card bg-dark text-white">
  <img src="..." class="card-img" alt="...">
  <div class="card-img-overlay">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">This is a wider card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
    <p class="card-text">Last updated 3 mins ago</p>
  </div>
</div>
```

Note that content should not be larger than the height of the image. If content is larger than the image the content will be displayed outside the image.

Horizontal

Using a combination of grid and utility classes, cards can be made horizontal in a mobile-friendly and responsive way. In the example below, we remove the grid gutters with `.g-0` and use `.col-md-*` classes to make the card horizontal at the `md` breakpoint. Further adjustments may be needed depending on your card content.

Image

Card title
This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

```
<div class="card mb-3" style="max-width: 540px;">
  <div class="row g-0">
    <div class="col-md-4">
      <img src="..." class="img-fluid rounded-start" alt="...">
    </div>
    <div class="col-md-8">
```

```
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a wider card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
        <p class="card-text"><small class="text-muted">Last updated 3 mins ago</small></p>
      </div>
    </div>
  </div>
</div>
```

Card styles

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

Background and color

Use text color and background utilities to change the appearance of a card.

Header

Primary card title

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

Header

Secondary card title

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

Header

Success card title

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

Header

Danger card title

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

Header

Warning card title

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

Header

Info card title

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

Header

Light card title

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

Header

Dark card title

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

```
<div class="card text-white bg-primary mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Primary card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-white bg-secondary mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Secondary card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-white bg-success mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Success card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-white bg-danger mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Danger card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-dark bg-warning mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Warning card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-dark bg-info mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Info card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-dark bg-light mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Light card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
  </div>
</div>
<div class="card text-white bg-dark mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Dark card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
  </div>
</div>
```

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (e.g. the visible text), or is included through alternative means, such as additional text hidden with the `.visually-hidden` class.

Border

Use border utilities to change just the `border-color` of a card. Note that you can put `.text-{color}` classes on the parent `.card` or a subset of the card's contents as shown below.

Header

Primary card title

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

Header

Secondary card title

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

Header

Success card title

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

Header

Danger card title

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

Header

Warning card title

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

Header

Info card title

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

Header

Light card title

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

Header

Dark card title

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

```
<div class="card border-primary mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body text-primary">
    <h5 class="card-title">Primary card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
</div>
<div class="card border-secondary mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body text-secondary">
    <h5 class="card-title">Secondary card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
</div>
<div class="card border-success mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body text-success">
    <h5 class="card-title">Success card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
</div>
<div class="card border-danger mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body text-danger">
    <h5 class="card-title">Danger card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
</div>
<div class="card border-warning mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Warning card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
</div>
<div class="card border-info mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Info card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
</div>
<div class="card border-light mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Light card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
</div>
<div class="card border-dark mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body text-dark">
    <h5 class="card-title">Dark card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
</div>
```

Mixins utilities

You can also change the borders on the card header and footer as needed, and even remove their `background-color` with `.bg-transparent`.

Header

Success card title

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

```
<div class="card border-success mb-3" style="max-width: 18rem;">
  <div class="card-header bg-transparent border-success">Header</div>
  <div class="card-body text-success">
    <h5 class="card-title">Success card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
  </div>
  <div class="card-footer bg-transparent border-success">Footer</div>
</div>
```

Card layout

In addition to styling the content within cards, Bootstrap includes a few options for laying out series of cards. For the time being, **these layout options are not yet responsive**.

Card groups

Use card groups to render cards as a single, attached element with equal width and height columns. Card groups start off stacked and use `display: flex;` to become attached with uniform dimensions starting at the `sm` breakpoint.

Image cap

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago
Image cap

Card title

This card has supporting text below as a natural lead-in to additional content.

Last updated 3 mins ago
Image cap

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This card has even longer content than the first to show that equal height action.

Last updated 3 mins ago

```
<div class="card-group">
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This is a wider card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
      <p class="card-text"><small class="text-muted">Last updated 3 mins ago</small></p>
    </div>
  </div>
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This card has supporting text below as a natural lead-in to additional content.</p>
      <p class="card-text"><small class="text-muted">Last updated 3 mins ago</small></p>
    </div>
  </div>
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This is a wider card with supporting text below as a natural lead-in to additional content.
This card has even longer content than the first to show that equal height action.</p>
      <p class="card-text"><small class="text-muted">Last updated 3 mins ago</small></p>
    </div>
  </div>
</div>
```

When using card groups with footers, their content will automatically line up.

Image cap

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

Image cap

Card title

This card has supporting text below as a natural lead-in to additional content.

Last updated 3 mins ago

Image cap

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This card has even longer content than the first to show that equal height action.

Last updated 3 mins ago

```
<div class="card-group">
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This is a wider card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 3 mins ago</small>
    </div>
  </div>
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This card has supporting text below as a natural lead-in to additional content.</p>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 3 mins ago</small>
    </div>
  </div>
  <div class="card">
    <img src="..." class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">This is a wider card with supporting text below as a natural lead-in to additional content.
This card has even longer content than the first to show that equal height action.</p>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 3 mins ago</small>
    </div>
  </div>
</div>
```

Grid cards

Use the Bootstrap grid system and its `.row-cols` classes to control how many grid columns (wrapped around your cards) you show per row. For example, here's `.row-cols-1` laying out the cards on one column, and `.row-cols-md-2` splitting four cards to equal width across multiple rows, from the medium breakpoint up.

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content.

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

```
<div class="row row-cols-1 row-cols-md-2 g-4">
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional
content.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
      </div>
    </div>
  </div>
</div>
```

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

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content.

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

```
<div class="row row-cols-1 row-cols-md-3 g-4">
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional
content.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
      </div>
    </div>
  </div>
</div>
```

When you need equal height, add `.h-100` to the cards. If you want equal heights by default, you can set `$card-height: 100%` in Sass.

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Image cap

Card title

This is a short card.

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content.

Image cap

Card title

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

```
<div class="row row-cols-1 row-cols-md-3 g-4">
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a short card.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional
content.</p>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a longer card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
      </div>
    </div>
  </div>
</div>
```
Just like with card groups, card footers will automatically line up.

Image cap

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Last updated 3 mins ago

Image cap

Card title

This card has supporting text below as a natural lead-in to additional content.

Last updated 3 mins ago

Image cap

Card title

This is a wider card with supporting text below as a natural lead-in to additional content. This card has even longer content than the first to show that equal height action.
Last updated 3 mins ago

```
<div class="row row-cols-1 row-cols-md-3 g-4">
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a wider card with supporting text below as a natural lead-in to additional content.
This content is a little bit longer.</p>
      </div>
      <div class="card-footer">
        <small class="text-muted">Last updated 3 mins ago</small>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This card has supporting text below as a natural lead-in to additional content.</p>
      </div>
      <div class="card-footer">
        <small class="text-muted">Last updated 3 mins ago</small>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card h-100">
      <img src="..." class="card-img-top" alt="...">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a wider card with supporting text below as a natural lead-in to additional content.
This card has even longer content than the first to show that equal height action.</p>
      </div>
      <div class="card-footer">
        <small class="text-muted">Last updated 3 mins ago</small>
      </div>
    </div>
  </div>
</div>
```

Masonry

In `v4` we used a CSS-only technique to mimic the behavior of Masonry-like columns, but this technique came with lots of unpleasant side effects. If you want to have this type of layout in v5, you can just make use of Masonry plugin. **Masonry is not included in Bootstrap**, but we've made a demo example to help you get started.

Sass

Variables

```
$card-spacer-y:                    $spacer;
$card-spacer-x:                    $spacer;
$card-title-spacer-y:              $spacer * .5;
$card-border-width:                $border-width;
$card-border-radius:               $border-radius;
$card-border-color:                rgba($black, .125);
$card-inner-border-radius:         subtract($card-border-radius, $card-border-width);
$card-cap-padding-y:               $card-spacer-y * .5;
$card-cap-padding-x:               $card-spacer-x;
$card-cap-bg:                      rgba($black, .03);
$card-cap-color:                   null;
$card-height:                      null;
$card-color:                       null;
$card-bg:                          $white;
$card-img-overlay-padding:         $spacer;
$card-group-margin:                $grid-gutter-width * .5;
```

Carousel

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

How it works

The carousel is a slideshow for cycling through a series of content, built with CSS 3D transforms and a bit of JavaScript. It works with a series of images, text, or custom markup. It also includes support for previous/next controls and indicators.

In browsers where the Page Visibility API is supported, the carousel will avoid sliding when the webpage is not visible to the user (such as when the browser tab is inactive, the browser window is minimized, etc.).

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

Please be aware that nested carousels are not supported, and carousels are generally not compliant with accessibility standards.

Example

Carousels don't automatically normalize slide dimensions. As such, you may need to use additional utilities or custom styles to appropriately size content. While carousels support previous/next controls and indicators, they're not explicitly required. Add and customize as you see fit.

The `.active` class needs to be added to one of the slides otherwise the carousel will not be visible. Also be sure to set a unique `id` on the `.carousel` for optional controls, especially if you're using multiple carousels on a single page. Control and indicator elements must have a `data-bs-target` attribute (or `href` for links) that matches the `id` of the `.carousel` element.

Slides only

Here's a carousel with slides only. Note the presence of the `.d-block` and `.w-100` on carousel images to prevent browser default image alignment.

Third slide

```
<div id="carouselExampleSlidesOnly" class="carousel slide" data-bs-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
</div>
```

With controls

Adding in the previous and next controls. We recommend using `<button>` elements, but you can also use `<a>` elements with `role="button"`.

Second slide

Third slide

PreviousNext

```
<div id="carouselExampleControls" class="carousel slide" data-bs-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleControls" data-bs-
slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleControls" data-bs-
slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

With indicators

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

Second slide

PreviousNext

```
<div id="carouselExampleIndicators" class="carousel slide" data-bs-ride="carousel">
  <div class="carousel-indicators">
    <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="0" class="active" aria-
current="true" aria-label="Slide 1"></button>
    <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="1" aria-label="Slide
2"></button>
    <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="2" aria-label="Slide
3"></button>
  </div>
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleIndicators" data-bs-
slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleIndicators" data-bs-
slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

With captions

Add captions to your slides easily with the `.carousel-caption` element within any `.carousel-item`. They can be easily hidden on smaller viewports, as shown below, with optional display utilities. We hide them initially with `.d-none` and bring them back on medium-sized devices with `.d-md-block`.

Third slide

PreviousNext

```
<div id="carouselExampleCaptions" class="carousel slide" data-bs-ride="carousel">
  <div class="carousel-indicators">
    <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="0" class="active" aria-
current="true" aria-label="Slide 1"></button>
    <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="1" aria-label="Slide
2"></button>
    <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="2" aria-label="Slide
3"></button>
  </div>
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>First slide label</h5>
        <p>Some representative placeholder content for the first slide.</p>
      </div>
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>Second slide label</h5>
        <p>Some representative placeholder content for the second slide.</p>
      </div>
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>Third slide label</h5>
        <p>Some representative placeholder content for the third slide.</p>
      </div>
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleCaptions" data-bs-
slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleCaptions" data-bs-
slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

Crossfade

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

Third slide

PreviousNext

```
<div id="carouselExampleFade" class="carousel slide carousel-fade" data-bs-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleFade" data-bs-
slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleFade" data-bs-
slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

Individual .carousel-item interval

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

First slide

PreviousNext

```
<div id="carouselExampleInterval" class="carousel slide" data-bs-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active" data-bs-interval="10000">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item" data-bs-interval="2000">
```

```
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleInterval" data-bs-
slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleInterval" data-bs-
slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

Disable touch swiping

Carousels support swiping left/right on touchscreen devices to move between slides. This can be disabled using the `data-bs-touch` attribute. The example below also does not include the `data-bs-ride` attribute and has `data-bs-interval="false"` so it doesn't autoplay.

First slide

PreviousNext

```
<div id="carouselExampleControlsNoTouching" class="carousel slide" data-bs-touch="false" data-bs-
interval="false">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleControlsNoTouching"
data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleControlsNoTouching"
data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
```

```
</div>
```

Dark variant

Add .carousel-dark to the .carousel for darker controls, indicators, and captions. Controls have been inverted from their default white fill with the filter CSS property. Captions and controls have additional Sass variables that customize the color and background-color.

First slide

First slide label

Some representative placeholder content for the first slide.

PreviousNext

```
<div id="carouselExampleDark" class="carousel carousel-dark slide" data-bs-ride="carousel">
  <div class="carousel-indicators">
    <button type="button" data-bs-target="#carouselExampleDark" data-bs-slide-to="0" class="active" aria-
current="true" aria-label="Slide 1"></button>
    <button type="button" data-bs-target="#carouselExampleDark" data-bs-slide-to="1" aria-label="Slide
2"></button>
    <button type="button" data-bs-target="#carouselExampleDark" data-bs-slide-to="2" aria-label="Slide
3"></button>
  </div>
  <div class="carousel-inner">
    <div class="carousel-item active" data-bs-interval="10000">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>First slide label</h5>
        <p>Some representative placeholder content for the first slide.</p>
      </div>
    </div>
    <div class="carousel-item" data-bs-interval="2000">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>Second slide label</h5>
        <p>Some representative placeholder content for the second slide.</p>
      </div>
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
      <div class="carousel-caption d-none d-md-block">
        <h5>Third slide label</h5>
        <p>Some representative placeholder content for the third slide.</p>
      </div>
    </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleDark" data-bs-
slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleDark" data-bs-
slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```

Custom transition

The transition duration of `.carousel-item` can be changed with the `$carousel-transition-duration` Sass variable before compiling or custom styles if you're using the compiled CSS. If multiple transitions are applied, make sure the transform transition is defined first (eg. `transition: transform 2s ease, opacity .5s ease-out`).

Sass

Variables

```
$carousel-control-color:            $white;
$carousel-control-width:            15%;
$carousel-control-opacity:          .5;
$carousel-control-hover-opacity:    .9;
$carousel-control-transition:       opacity .15s ease;

$carousel-indicator-width:          30px;
$carousel-indicator-height:         3px;
$carousel-indicator-hit-area-height: 10px;
$carousel-indicator-spacer:         3px;
$carousel-indicator-opacity:        .5;
$carousel-indicator-active-bg:      $white;
$carousel-indicator-active-opacity: 1;
$carousel-indicator-transition:     opacity .6s ease;

$carousel-caption-width:            70%;
$carousel-caption-color:            $white;
$carousel-caption-padding-y:        1.25rem;
$carousel-caption-spacer:           1.25rem;

$carousel-control-icon-width:       2rem;

$carousel-control-prev-icon-bg:     url("data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg' viewBox='0
0 16 16' fill='#{$carousel-control-color}'><path d='M11.354 1.646a.5.5 0 0 1 0 .708L5.707 8l5.647 5.646a.5.5 0 0
1-.708.708l-6-6a.5.5 0 0 1 0-.708l6-6a.5.5 0 0 1 .708 0z'/></svg>");
$carousel-control-next-icon-bg:     url("data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg' viewBox='0
0 16 16' fill='#{$carousel-control-color}'><path d='M4.646 1.646a.5.5 0 0 1 .708 0l6 6a.5.5 0 0 1 0 .708l-6 6a.5.5 0
0 1-.708-.708L10.293 8 4.646 2.354a.5.5 0 0 1 0-.708z'/></svg>");
```

```
$carousel-transition-duration:       .6s;
$carousel-transition:               transform $carousel-transition-duration ease-in-out; // Define transform transition first
if using multiple transitions (e.g., `transform 2s ease, opacity .5s ease-out`)

$carousel-dark-indicator-active-bg:  $black;
$carousel-dark-caption-color:        $black;
$carousel-dark-control-icon-filter:  invert(1) grayscale(100);
```

Usage

Via data attributes

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

The data-bs-ride="carousel" attribute is used to mark a carousel as animating starting at page load. If you don't use data-bs-ride="carousel" to initialize your carousel, you have to initialize it yourself. **It cannot be used in combination with (redundant and unnecessary) explicit JavaScript initialization of the same carousel.**

Via JavaScript

Call carousel manually with:

```
var myCarousel = document.querySelector('#myCarousel')
var carousel = new bootstrap.Carousel(myCarousel)
```

Name	Type	Default	Description
interval	number	5000	The amount of time to delay between automatically cycling an item. If false, carousel will not automatically cycle.
keyboard	boolean	true	Whether the carousel should react to keyboard events.
pause	string | boolean	'hover'	If set to 'hover', pauses the cycling of the carousel on mouseenter and resumes the cycling of the carousel on mouseleave. If set to false, hovering over the carousel won't pause it. On touch-enabled devices, when set to 'hover', cycling will pause on touchend (once the user finished interacting with the carousel) for two intervals, before automatically

			resuming. Note that this is in addition to the above mouse behavior.
ride	string | boolean	false	Autoplays the carousel after the user manually cycles the first item. If set to 'carousel', autoplays the carousel on load.
wrap	boolean	true	Whether the carousel should cycle continuously or have hard stops.
touch	boolean	true	Whether the carousel should support left/right swipe interactions on touchscreen devices.

Options

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

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.
You can create a carousel instance with the carousel constructor, for example, to initialize with additional options and start cycling through items:

Method	Description
cycle	Cycles through the carousel items from left to right.
pause	Stops the carousel from cycling through items.
prev	Cycles to the previous item. **Returns to the caller before the previous item has been shown** (e.g., before the slid.bs.carousel event occurs).
next	Cycles to the next item. **Returns to the caller before the next item has been shown** (e.g., before the slid.bs.carousel event occurs).
nextWhenVisible	Don't cycle carousel to next when the page isn't visible or the carousel or its parent isn't visible. **Returns to the caller before the target item has been shown**
to	Cycles the carousel to a particular frame (0 based, similar to an array). **Returns to the caller before the target item has been shown** (e.g., before the slid.bs.carousel event occurs).
dispose	Destroys an element's carousel. (Removes stored data on the DOM element)

getInstance	Static method which allows you to get the carousel instance associated to a DOM element, you can use it like this: bootstrap.Carousel.getInstance(element)
getOrCreateInstance	Static method which returns a carousel instance associated to a DOM element or create a new one in case it wasn't initialised. You can use it like this: bootstrap.Carousel.getOrCreateInstance(element)

```
var myCarousel = document.querySelector('#myCarousel')
var carousel = new bootstrap.Carousel(myCarousel, {
  interval: 2000,
  wrap: false
})
```

Events

Bootstrap's carousel class exposes two events for hooking into carousel functionality. Both events have the following additional properties:

- direction: The direction in which the carousel is sliding (either "left" or "right").
- relatedTarget: The DOM element that is being slid into place as the active item.
- from: The index of the current item
- to: The index of the next item

All carousel events are fired at the carousel itself (i.e. at the <div class="carousel">).

Event type	Description
slide.bs.carousel	Fires immediately when the slide instance method is invoked.
slid.bs.carousel	Fired when the carousel has completed its slide transition.

```
var myCarousel = document.getElementById('myCarousel')

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

Dropdowns

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

Overview

Dropdowns are toggleable, contextual overlays for displaying lists of links and more. They're made interactive with the included Bootstrap dropdown JavaScript plugin. They're toggled by clicking, not by hovering; this is an intentional design decision.

Dropdowns are built on a third party library, Popper, which provides dynamic positioning and viewport detection. Be sure to include popper.min.js before Bootstrap's JavaScript or use `bootstrap.bundle.min.js` / `bootstrap.bundle.js` which contains Popper. Popper isn't used to position dropdowns in navbars though as dynamic positioning isn't required.

Accessibility

The WAI ARIA standard defines an actual `role="menu"` widget, but this is specific to application-like menus which trigger actions or functions. ARIA menus can only contain menu items, checkbox menu items, radio button menu items, radio button groups, and sub-menus.

Bootstrap's dropdowns, on the other hand, are designed to be generic and applicable to a variety of situations and markup structures. For instance, it is possible to create dropdowns that contain additional inputs and form controls, such as search fields or login forms. For this reason, Bootstrap does not expect (nor automatically add) any of the `role` and `aria-` attributes required for true ARIA menus. Authors will have to include these more specific attributes themselves.

However, Bootstrap does add built-in support for most standard keyboard menu interactions, such as the ability to move through individual `.dropdown-item` elements using the cursor keys and close the menu with the ESC key.

Examples

Wrap the dropdown's toggle (your button or link) and the dropdown menu within `.dropdown`, or another element that declares `position: relative;`. Dropdowns can be triggered from <a> or <button> elements to better fit your potential needs. The examples shown here use semantic <ul> elements where appropriate, but custom markup is supported.

Single button

Any single .btn can be turned into a dropdown toggle with some markup changes. Here's how you can put them to work with either <button> elements:

Dropdown button

```
<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-expanded="false">
    Dropdown button
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
  </ul>
</div>
```

And with <a> elements:

Dropdown link

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

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

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

Primary

Secondary

Success

Info

Warning

Danger

```
<!-- Example single danger button -->
<div class="btn-group">
  <button type="button" class="btn btn-danger dropdown-toggle" data-bs-toggle="dropdown" aria-
expanded="false">
    Action
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>
```

Split button

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

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

PrimaryToggle Dropdown

SecondaryToggle Dropdown

SuccessToggle Dropdown

InfoToggle Dropdown

WarningToggle Dropdown

DangerToggle Dropdown

```
<!-- Example split danger button -->
<div class="btn-group">
  <button type="button" class="btn btn-danger">Action</button>
  <button type="button" class="btn btn-danger dropdown-toggle dropdown-toggle-split" data-bs-toggle="dropdown"
aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>
```

Sizing

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

Large button

Large split buttonToggle Dropdown

```
<!-- Large button groups (default and split) -->
<div class="btn-group">
  <button class="btn btn-secondary btn-lg dropdown-toggle" type="button" data-bs-toggle="dropdown" aria-
expanded="false">
    Large button
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
```

```
<div class="btn-group">
  <button class="btn btn-secondary btn-lg" type="button">
    Large split button
  </button>
  <button type="button" class="btn btn-lg btn-secondary dropdown-toggle dropdown-toggle-split" data-bs-
toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
```

Small button

Small split buttonToggle Dropdown

```
<div class="btn-group">
  <button class="btn btn-secondary btn-sm dropdown-toggle" type="button" data-bs-toggle="dropdown" aria-
expanded="false">
    Small button
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
<div class="btn-group">
  <button class="btn btn-secondary btn-sm" type="button">
    Small split button
  </button>
  <button type="button" class="btn btn-sm btn-secondary dropdown-toggle dropdown-toggle-split" data-bs-
toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
```

Dark dropdowns

Opt into darker dropdowns to match a dark navbar or custom style by adding `.dropdown-menu-dark` onto an existing `.dropdown-menu`. No changes are required to the dropdown items.

Dropdown button

```
<div class="dropdown">
```

```
  <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButton2" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropdown button
  </button>
  <ul class="dropdown-menu dropdown-menu-dark" aria-labelledby="dropdownMenuButton2">
    <li><a class="dropdown-item active" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Separated link</a></li>
  </ul>
</div>
```

And putting it to use in a navbar:

- Dropdown

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNavDarkDropdown" aria-controls="navbarNavDarkDropdown" aria-expanded="false" aria-
label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavDarkDropdown">
      <ul class="navbar-nav">
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDarkDropdownMenuLink" role="button" data-bs-
toggle="dropdown" aria-expanded="false">
            Dropdown
          </a>
          <ul class="dropdown-menu dropdown-menu-dark" aria-labelledby="navbarDarkDropdownMenuLink">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Directions

RTL

Directions are mirrored when using Bootstrap in RTL, meaning `.dropstart` will appear on the right side.

Dropup

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

Dropup

Split dropupToggle Dropdown

```
<!-- Default dropup button -->
<div class="btn-group dropup">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-toggle="dropdown" aria-
expanded="false">
    Dropup
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>

<!-- Split dropup button -->
<div class="btn-group dropup">
  <button type="button" class="btn btn-secondary">
    Split dropup
  </button>
  <button type="button" class="btn btn-secondary dropdown-toggle dropdown-toggle-split" data-bs-
toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>
```

Dropright

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

Dropright

Split dropendToggle Dropright

```
<!-- Default dropend button -->
<div class="btn-group dropend">
```

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

<!-- Split dropend button -->
<div class="btn-group dropend">
  <button type="button" class="btn btn-secondary">
    Split dropend
  </button>
  <button type="button" class="btn btn-secondary dropdown-toggle dropdown-toggle-split" data-bs-
toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropright</span>
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>
```

Dropleft

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

Dropleft

Toggle Dropleft

Split dropstart

```
<!-- Default dropstart button -->
<div class="btn-group dropstart">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-toggle="dropdown" aria-
expanded="false">
    Dropstart
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>

<!-- Split dropstart button -->
<div class="btn-group">
  <div class="btn-group dropstart" role="group">
    <button type="button" class="btn btn-secondary dropdown-toggle dropdown-toggle-split" data-bs-
toggle="dropdown" aria-expanded="false">
```

```
      <span class="visually-hidden">Toggle Dropstart</span>
    </button>
    <ul class="dropdown-menu">
      <!-- Dropdown menu links -->
    </ul>
  </div>
  <button type="button" class="btn btn-secondary">
    Split dropstart
  </button>
</div>
```

Menu items

You can use `<a>` or `<button>` elements as dropdown items.

Dropdown

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

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

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

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

Active

Add .active to items in the dropdown to **style them as active**. To convey the active state to assistive technologies, use the aria-current attribute — using the page value for the current page, or true for the current item in a set.

- Regular link
- Active link
- Another link

```
<ul class="dropdown-menu">
  <li><a class="dropdown-item" href="#">Regular link</a></li>
  <li><a class="dropdown-item active" href="#" aria-current="true">Active link</a></li>
  <li><a class="dropdown-item" href="#">Another link</a></li>
</ul>
```

Disabled

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

- Regular link
- Disabled link
- Another link

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

Menu alignment

By default, a dropdown menu is automatically positioned 100% from the top and along the left side of its parent. You can change this with the directional .drop* classes, but you can also control them with additional modifier classes.

Add .dropdown-menu-end to a .dropdown-menu to right align the dropdown menu. Directions are mirrored when using Bootstrap in RTL, meaning .dropdown-menu-end will appear on the left side.

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

Right-aligned menu example

```
<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-toggle="dropdown" aria-
expanded="false">
    Right-aligned menu example
  </button>
  <ul class="dropdown-menu dropdown-menu-end">
    <li><button class="dropdown-item" type="button">Action</button></li>
    <li><button class="dropdown-item" type="button">Another action</button></li>
    <li><button class="dropdown-item" type="button">Something else here</button></li>
  </ul>
</div>
```

Responsive alignment

If you want to use responsive alignment, disable dynamic positioning by adding the `data-bs-display="static"` attribute and use the responsive variation classes.

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

Left-aligned but right aligned when large screen

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

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

Right-aligned but left aligned when large screen

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

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

Alignment options

Taking most of the options shown above, here's a small kitchen sink demo of various dropdown alignment options in one place.

Dropdown

Right-aligned menu

Left-aligned, right-aligned lg

Right-aligned, left-aligned lg

Dropstart

Dropend

Dropup

```
<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButton" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropdown
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
```

```
</div>

<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-toggle="dropdown" aria-
expanded="false">
    Right-aligned menu
  </button>
  <ul class="dropdown-menu dropdown-menu-end">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-toggle="dropdown" data-bs-
display="static" aria-expanded="false">
    Left-aligned, right-aligned lg
  </button>
  <ul class="dropdown-menu dropdown-menu-lg-end">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-toggle="dropdown" data-bs-
display="static" aria-expanded="false">
    Right-aligned, left-aligned lg
  </button>
  <ul class="dropdown-menu dropdown-menu-end dropdown-menu-lg-start">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group dropstart">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-toggle="dropdown" aria-
expanded="false">
    Dropstart
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group dropend">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-toggle="dropdown" aria-
expanded="false">
    Dropend
  </button>
  <ul class="dropdown-menu">
```

```
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group dropup">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-toggle="dropdown" aria-
expanded="false">
    Dropup
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>
```

Menu content

Headers

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

- **Dropdown header**
- Action
- Another action

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

Dividers

Separate groups of related menu items with a divider.

- Action
- Another action
- Something else here
- ---
- Separated link

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

Text

Place any freeform text within a dropdown menu with text and use spacing utilities. Note that you'll likely need additional sizing styles to constrain the menu width.

Some example text that's free-flowing within the dropdown menu.

And this is more example text.

```
<div class="dropdown-menu p-4 text-muted" style="max-width: 200px;">
  <p>
    Some example text that's free-flowing within the dropdown menu.
  </p>
  <p class="mb-0">
    And this is more example text.
  </p>
</div>
```

Forms

Put a form within a dropdown menu, or make it into a dropdown menu, and use margin or padding utilities to give it the negative space you require.

Email address

Password

☐ Remember me

Sign in

New around here? Sign upForgot password?

```
<div class="dropdown-menu">
  <form class="px-4 py-3">
    <div class="mb-3">
      <label for="exampleDropdownFormEmail1" class="form-label">Email address</label>
      <input type="email" class="form-control" id="exampleDropdownFormEmail1"
placeholder="email@example.com">
```

```
    </div>
    <div class="mb-3">
      <label for="exampleDropdownFormPassword1" class="form-label">Password</label>
      <input type="password" class="form-control" id="exampleDropdownFormPassword1"
placeholder="Password">
    </div>
    <div class="mb-3">
      <div class="form-check">
        <input type="checkbox" class="form-check-input" id="dropdownCheck">
        <label class="form-check-label" for="dropdownCheck">
          Remember me
        </label>
      </div>
    </div>
    <button type="submit" class="btn btn-primary">Sign in</button>
  </form>
  <div class="dropdown-divider"></div>
  <a class="dropdown-item" href="#">New around here? Sign up</a>
  <a class="dropdown-item" href="#">Forgot password?</a>
</div>
```

Email address

Password

☐ Remember me

Sign in

```
<form class="dropdown-menu p-4">
  <div class="mb-3">
    <label for="exampleDropdownFormEmail2" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleDropdownFormEmail2"
placeholder="email@example.com">
  </div>
  <div class="mb-3">
    <label for="exampleDropdownFormPassword2" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleDropdownFormPassword2" placeholder="Password">
  </div>
  <div class="mb-3">
    <div class="form-check">
      <input type="checkbox" class="form-check-input" id="dropdownCheck2">
      <label class="form-check-label" for="dropdownCheck2">
        Remember me
      </label>
    </div>
  </div>
  <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

Dropdown options

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

Offset

ReferenceToggle Dropdown

```
<div class="d-flex">
  <div class="dropdown me-1">
    <button type="button" class="btn btn-secondary dropdown-toggle" id="dropdownMenuOffset" data-bs-
toggle="dropdown" aria-expanded="false" data-bs-offset="10,20">
      Offset
    </button>
    <ul class="dropdown-menu" aria-labelledby="dropdownMenuOffset">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
    </ul>
  </div>
  <div class="btn-group">
    <button type="button" class="btn btn-secondary">Reference</button>
    <button type="button" class="btn btn-secondary dropdown-toggle dropdown-toggle-split"
id="dropdownMenuReference" data-bs-toggle="dropdown" aria-expanded="false" data-bs-reference="parent">
      <span class="visually-hidden">Toggle Dropdown</span>
    </button>
    <ul class="dropdown-menu" aria-labelledby="dropdownMenuReference">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
      <li><hr class="dropdown-divider"></li>
      <li><a class="dropdown-item" href="#">Separated link</a></li>
    </ul>
  </div>
</div>
```

Auto close behavior

By default, the dropdown menu is closed when clicking inside or outside the dropdown menu. You can use the autoClose option to change this behavior of the dropdown.

Default dropdown

Clickable outside

Clickable inside

Manual close

```
<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle" type="button" id="defaultDropdown" data-bs-
toggle="dropdown" data-bs-auto-close="true" aria-expanded="false">
    Default dropdown
  </button>
  <ul class="dropdown-menu" aria-labelledby="defaultDropdown">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuClickableOutside" data-bs-
toggle="dropdown" data-bs-auto-close="inside" aria-expanded="false">
    Clickable outside
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuClickableOutside">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuClickableInside" data-bs-
toggle="dropdown" data-bs-auto-close="outside" aria-expanded="false">
    Clickable inside
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuClickableInside">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>

<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuClickable" data-bs-
toggle="dropdown" data-bs-auto-close="false" aria-expanded="false">
    Manual close
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuClickable">
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
    <li><a class="dropdown-item" href="#">Menu item</a></li>
  </ul>
</div>
```

Sass

Variables

Variables for all dropdowns:

```
$dropdown-min-width:               10rem;
$dropdown-padding-x:               0;
$dropdown-padding-y:               .5rem;
$dropdown-spacer:                .125rem;
$dropdown-font-size:              $font-size-base;
$dropdown-color:                 $body-color;
$dropdown-bg:                   $white;
$dropdown-border-color:            rgba($black, .15);
$dropdown-border-radius:           $border-radius;
$dropdown-border-width:            $border-width;
$dropdown-inner-border-radius:      subtract($dropdown-border-radius, $dropdown-border-width);
$dropdown-divider-bg:              $dropdown-border-color;
$dropdown-divider-margin-y:         $spacer * .5;
$dropdown-box-shadow:              $box-shadow;

$dropdown-link-color:              $gray-900;
$dropdown-link-hover-color:         shade-color($gray-900, 10%);
$dropdown-link-hover-bg:            $gray-200;

$dropdown-link-active-color:        $component-active-color;
$dropdown-link-active-bg:           $component-active-bg;

$dropdown-link-disabled-color:      $gray-500;

$dropdown-item-padding-y:           $spacer * .25;
$dropdown-item-padding-x:           $spacer;

$dropdown-header-color:             $gray-600;
$dropdown-header-padding:           $dropdown-padding-y $dropdown-item-padding-x;
```

Variables for the dark dropdown:

```
$dropdown-dark-color:               $gray-300;
$dropdown-dark-bg:                 $gray-800;
$dropdown-dark-border-color:        $dropdown-border-color;
$dropdown-dark-divider-bg:          $dropdown-divider-bg;
$dropdown-dark-box-shadow:          null;
$dropdown-dark-link-color:          $dropdown-dark-color;
$dropdown-dark-link-hover-color:    $white;
$dropdown-dark-link-hover-bg:       rgba($white, .15);
$dropdown-dark-link-active-color:   $dropdown-link-active-color;
$dropdown-dark-link-active-bg:      $dropdown-link-active-bg;
$dropdown-dark-link-disabled-color: $gray-500;
$dropdown-dark-header-color:        $gray-500;
```

Variables for the CSS-based carets that indicate a dropdown's interactivity:

```
$caret-width:                  .3em;
$caret-vertical-align:       $caret-width * .85;
$caret-spacing:              $caret-width * .85;
```

Mixins

Mixins are used to generate the CSS-based carets and can be found in scss/mixins/_caret.scss.

```
@mixin caret-down {
  border-top: $caret-width solid;
  border-right: $caret-width solid transparent;
  border-bottom: 0;
  border-left: $caret-width solid transparent;
}

@mixin caret-up {
  border-top: 0;
  border-right: $caret-width solid transparent;
  border-bottom: $caret-width solid;
  border-left: $caret-width solid transparent;
}

@mixin caret-end {
  border-top: $caret-width solid transparent;
  border-right: 0;
  border-bottom: $caret-width solid transparent;
  border-left: $caret-width solid;
}

@mixin caret-start {
  border-top: $caret-width solid transparent;
  border-right: $caret-width solid;
  border-bottom: $caret-width solid transparent;
}

@mixin caret($direction: down) {
  @if $enable-caret {
    &::after {
      display: inline-block;
      margin-left: $caret-spacing;
      vertical-align: $caret-vertical-align;
      content: "";
      @if $direction == down {
        @include caret-down();
      } @else if $direction == up {
        @include caret-up();
      } @else if $direction == end {
        @include caret-end();
      }
    }
    @if $direction == start {
      &::after {
        display: none;
      }

      &::before {
        display: inline-block;
        margin-right: $caret-spacing;
        vertical-align: $caret-vertical-align;
        content: "";
        @include caret-start();
      }
    }

    &:empty::after {
      margin-left: 0;
    }
  }
}
```

Usage

Via data attributes or JavaScript, the dropdown plugin toggles hidden content (dropdown menus) by toggling the `.show` class on the parent `.dropdown-menu`. The `data-bs-toggle="dropdown"` attribute is relied on for closing dropdown menus at an application level, so it's a good idea to always use it.

On touch-enabled devices, opening a dropdown adds empty `mouseover` handlers to the immediate children of the `<body>` element. This admittedly ugly hack is necessary to work around a quirk in iOS' event delegation, which would otherwise prevent a tap anywhere outside of the dropdown from triggering the code that closes the dropdown. Once the dropdown is closed, these additional empty `mouseover` handlers are removed.

Via data attributes

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

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

Via JavaScript

Call the dropdowns via JavaScript:

```
var dropdownElementList = [].slice.call(document.querySelectorAll('.dropdown-toggle'))
var dropdownList = dropdownElementList.map(function (dropdownToggleEl) {
  return new bootstrap.Dropdown(dropdownToggleEl)
})
```

Name	Type	Default	Description
boundary	string | element	'clippingParents'	Overflow constraint boundary of the dropdown menu (applies only to Popper's preventOverflow modifier). By default it's 'clippingParents' and can accept an HTMLElement reference (via JavaScript only). For more information refer to Popper's detectOverflow docs.
reference	string | element | object	'toggle'	Reference element of the dropdown menu. Accepts the values of 'toggle', 'parent', an HTMLElement reference or an object providing getBoundingClientRect. For more information refer to Popper's constructor docs and virtual element docs.
display	string	'dynamic'	By default, we use Popper for dynamic positioning. Disable this with static.
offset	array | string | function	[0, 2]	Offset of the dropdown relative to its target. You can pass a string in data attributes with comma separated values like: data-bs-offset="10,20" When a function is used to determine the offset, it is called with an object containing the popper placement, the reference, and popper rects as its first argument. The triggering element DOM node is passed as the second argument. The function must return an array with two numbers: [skidding, distance]. For more information refer to Popper's offset docs.

autoClose	boolean | string	true	Configure the auto close behavior of the dropdown: • true - the dropdown will be closed by clicking outside or inside the dropdown menu. • false - the dropdown will be closed by clicking the toggle button and manually calling hide or toggle method. (Also will not be closed by pressing esc key) • 'inside' - the dropdown will be closed (only) by clicking inside the dropdown menu. • 'outside' - the dropdown will be closed (only) by clicking outside the dropdown menu.
popperConfig	null | object | function	null	To change Bootstrap's default Popper config, see Popper's configuration. When a function is used to create the Popper configuration, it's called with an object that contains the Bootstrap's default Popper configuration. It helps you use and merge the default with your own configuration. The function must return a configuration object for Popper.

data-bs-toggle="dropdown" still required

Regardless of whether you call your dropdown via JavaScript or instead use the data-api, data-bs-toggle="dropdown" is always required to be present on the dropdown's trigger element.

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to data-bs-, as in data-bs-offset="". Make sure to change the case type of the option name from camelCase to kebab-case when passing the options via data attributes. For example, instead of using data-bs-autoClose="false", use data-bs-auto-close="false".

Using function with popperConfig

```
var dropdown = new bootstrap.Dropdown(element, {
  popperConfig: function (defaultBsPopperConfig) {
    // var newPopperConfig = {...}
    // use defaultBsPopperConfig if needed...
    // return newPopperConfig
  }
})
```

Methods

Method	Description
toggle	Toggles the dropdown menu of a given navbar or tabbed navigation.
show	Shows the dropdown menu of a given navbar or tabbed navigation.
hide	Hides the dropdown menu of a given navbar or tabbed navigation.
update	Updates the position of an element's dropdown.
dispose	Destroys an element's dropdown. (Removes stored data on the DOM element)
getInstance	Static method which allows you to get the dropdown instance associated to a DOM element, you can use it like this: bootstrap.Dropdown.getInstance(element)
getOrCreateInstance	Static method which returns a dropdown instance associated to a DOM element or create a new one in case it wasn't initialised. You can use it like this: bootstrap.Dropdown.getOrCreateInstance(element)

Events

All dropdown events are fired at the toggling element and then bubbled up. So you can also add event listeners on the .dropdown-menu's parent element. hide.bs.dropdown and hidden.bs.dropdown events have a clickEvent property (only when the original Event type is click) that contains an Event Object for the click event.

Method	Description
show.bs.dropdown	Fires immediately when the show instance method is called.
shown.bs.dropdown	Fired when the dropdown has been made visible to the user and CSS transitions have completed.
hide.bs.dropdown	Fires immediately when the hide instance method has been called.
hidden.bs.dropdown	Fired when the dropdown has finished being hidden from the user and CSS transitions have completed.

```
var myDropdown = document.getElementById('myDropdown')
```

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

List group

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

Basic example

The most basic list group is an unordered list with list items and the proper classes. Build upon it with the options that follow, or with your own CSS as needed.

- An item
- A second item
- A third item
- A fourth item
- And a fifth one

```
<ul class="list-group">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
</ul>
```

Active items

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

- An active item
- A second item
- A third item
- A fourth item
- And a fifth one

```
<ul class="list-group">
  <li class="list-group-item active" aria-current="true">An active item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
</ul>
```

Disabled items

Add .disabled to a .list-group-item to make it *appear* disabled. Note that some elements with .disabled will also require custom JavaScript to fully disable their click events (e.g., links).

- A disabled item
- A second item
- A third item
- A fourth item
- And a fifth one

```
<ul class="list-group">
  <li class="list-group-item disabled" aria-disabled="true">A disabled item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
</ul>
```

Links and buttons

Use <a>s or <button>s to create *actionable* list group items with hover, disabled, and active states by adding .list-group-item-action. We separate these pseudo-classes to ensure list groups made of non-interactive elements (like <li>s or <div>s) don't provide a click or tap affordance.

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

The current link itemA second link itemA third link itemA fourth link itemA disabled link item

```
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action active" aria-current="true">
    The current link item
  </a>
  <a href="#" class="list-group-item list-group-item-action">A second link item</a>
  <a href="#" class="list-group-item list-group-item-action">A third link item</a>
  <a href="#" class="list-group-item list-group-item-action">A fourth link item</a>
  <a href="#" class="list-group-item list-group-item-action disabled" tabindex="-1" aria-disabled="true">A disabled
link item</a>
</div>
```

With <button>s, you can also make use of the disabled attribute instead of the .disabled class. Sadly, <a>s don't support the disabled attribute.

The current buttonA second itemA third button itemA fourth button itemA disabled button item

```
<div class="list-group">
  <button type="button" class="list-group-item list-group-item-action active" aria-current="true">
    The current button
  </button>
  <button type="button" class="list-group-item list-group-item-action">A second item</button>
  <button type="button" class="list-group-item list-group-item-action">A third button item</button>
  <button type="button" class="list-group-item list-group-item-action">A fourth button item</button>
  <button type="button" class="list-group-item list-group-item-action" disabled>A disabled button item</button>
</div>
```

Flush

Add .list-group-flush to remove some borders and rounded corners to render list group items edge-to-edge in a parent container (e.g., cards).

- An item
- A second item
- A third item
- A fourth item
- And a fifth one

```
<ul class="list-group list-group-flush">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
</ul>
```

Numbered

Add the .list-group-numbered modifier class (and optionally use an <ol> element) to opt into numbered list group items. Numbers are generated via CSS (as opposed to a <ol>s default browser styling) for better placement inside list group items and to allow for better customization.

Numbers are generated by counter-reset on the <ol>, and then styled and placed with a ::before pseudo-element on the <li> with counter-increment and content.

1. Cras justo odio
2. Cras justo odio

3. Cras justo odio

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

These work great with custom content as well.

1. Subheading

   Cras justo odio

2. Subheading

   Cras justo odio

3. Subheading

   Cras justo odio

```
<ol class="list-group list-group-numbered">
  <li class="list-group-item d-flex justify-content-between align-items-start">
    <div class="ms-2 me-auto">
      <div class="fw-bold">Subheading</div>
      Cras justo odio
    </div>
    <span class="badge bg-primary rounded-pill">14</span>
  </li>
  <li class="list-group-item d-flex justify-content-between align-items-start">
    <div class="ms-2 me-auto">
      <div class="fw-bold">Subheading</div>
      Cras justo odio
    </div>
    <span class="badge bg-primary rounded-pill">14</span>
  </li>
  <li class="list-group-item d-flex justify-content-between align-items-start">
    <div class="ms-2 me-auto">
      <div class="fw-bold">Subheading</div>
      Cras justo odio
    </div>
    <span class="badge bg-primary rounded-pill">14</span>
  </li>
</ol>
```

Horizontal

Add .list-group-horizontal to change the layout of list group items from vertical to horizontal across all breakpoints. Alternatively, choose a responsive variant .list-group-horizontal-{sm|md|lg|xl|xxl} to make a list group horizontal starting at that breakpoint's min-width. Currently **horizontal list groups cannot be combined with flush list groups.**

ProTip: Want equal-width list group items when horizontal? Add .flex-fill to each list group item.

- An item
- A second item
- A third item

- An item
- A second item
- A third item

- An item
- A second item
- A third item

- An item
- A second item
- A third item

- An item
- A second item
- A third item

- An item
- A second item
- A third item

```
<ul class="list-group list-group-horizontal">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
```

```
<ul class="list-group list-group-horizontal-sm">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
<ul class="list-group list-group-horizontal-md">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
<ul class="list-group list-group-horizontal-lg">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
<ul class="list-group list-group-horizontal-xl">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
<ul class="list-group list-group-horizontal-xxl">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
</ul>
```

Contextual classes

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

- A simple default list group item
- A simple primary list group item
- A simple secondary list group item
- A simple success list group item
- A simple danger list group item
- A simple warning list group item
- A simple info list group item
- A simple light list group item
- A simple dark list group item

```
<ul class="list-group">
  <li class="list-group-item">A simple default list group item</li>

  <li class="list-group-item list-group-item-primary">A simple primary list group item</li>
  <li class="list-group-item list-group-item-secondary">A simple secondary list group item</li>
  <li class="list-group-item list-group-item-success">A simple success list group item</li>
  <li class="list-group-item list-group-item-danger">A simple danger list group item</li>
  <li class="list-group-item list-group-item-warning">A simple warning list group item</li>
  <li class="list-group-item list-group-item-info">A simple info list group item</li>
  <li class="list-group-item list-group-item-light">A simple light list group item</li>
  <li class="list-group-item list-group-item-dark">A simple dark list group item</li>
</ul>
```

Contextual classes also work with .list-group-item-action. Note the addition of the hover styles here not present in the previous example. Also supported is the .active state; apply it to indicate an active selection on a contextual list group item.

A simple default list group itemA simple primary list group itemA simple secondary list group itemA simple success list group itemA simple danger list group itemA simple warning list group itemA simple info list group itemA simple light list group itemA simple dark list group item

```
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action">A simple default list group item</a>

  <a href="#" class="list-group-item list-group-item-action list-group-item-primary">A simple primary list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-secondary">A simple secondary list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-success">A simple success list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-danger">A simple danger list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-warning">A simple warning list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-info">A simple info list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-light">A simple light list group item</a>
  <a href="#" class="list-group-item list-group-item-action list-group-item-dark">A simple dark list group item</a>
</div>
```

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (e.g. the visible text), or is included through alternative means, such as additional text hidden with the .visually-hidden class.

With badges

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

- A list item
- A second list item
- A third list item

```
<ul class="list-group">
```

```
  <li class="list-group-item d-flex justify-content-between align-items-center">
    A list item
    <span class="badge bg-primary rounded-pill">14</span>
  </li>
  <li class="list-group-item d-flex justify-content-between align-items-center">
    A second list item
    <span class="badge bg-primary rounded-pill">2</span>
  </li>
  <li class="list-group-item d-flex justify-content-between align-items-center">
    A third list item
    <span class="badge bg-primary rounded-pill">1</span>
  </li>
</ul>
```

Custom content

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

List group item heading

3 days ago

Some placeholder content in a paragraph.

And some small print.

List group item heading

3 days ago

Some placeholder content in a paragraph.

And some muted small print.

List group item heading

3 days ago

Some placeholder content in a paragraph.

And some muted small print.

```
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action active" aria-current="true">
    <div class="d-flex w-100 justify-content-between">
      <h5 class="mb-1">List group item heading</h5>
      <small>3 days ago</small>
    </div>
    <p class="mb-1">Some placeholder content in a paragraph.</p>
    <small>And some small print.</small>
  </a>
  <a href="#" class="list-group-item list-group-item-action">
    <div class="d-flex w-100 justify-content-between">
      <h5 class="mb-1">List group item heading</h5>
      <small class="text-muted">3 days ago</small>
    </div>
    <p class="mb-1">Some placeholder content in a paragraph.</p>
    <small class="text-muted">And some muted small print.</small>
  </a>
  <a href="#" class="list-group-item list-group-item-action">
    <div class="d-flex w-100 justify-content-between">
      <h5 class="mb-1">List group item heading</h5>
      <small class="text-muted">3 days ago</small>
    </div>
    <p class="mb-1">Some placeholder content in a paragraph.</p>
    <small class="text-muted">And some muted small print.</small>
  </a>
</div>
```

Checkboxes and radios

Place Bootstrap's checkboxes and radios within list group items and customize as needed. You can use them without `<label>`s, but please remember to include an `aria-label` attribute and value for accessibility.

- ☐ First checkbox
- ☐ Second checkbox
- ☐ Third checkbox
- ☐ Fourth checkbox
- ☐ Fifth checkbox

```
<ul class="list-group">
  <li class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="" aria-label="...">
    First checkbox
  </li>
  <li class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="" aria-label="...">
    Second checkbox
  </li>
  <li class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="" aria-label="...">
    Third checkbox
  </li>
  <li class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="" aria-label="...">
    Fourth checkbox
  </li>
  <li class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="" aria-label="...">
    Fifth checkbox
  </li>
</ul>
```

And if you want `<label>`s as the `.list-group-item` for large hit areas, you can do that, too.

☐ First checkbox ☐ Second checkbox ☐ Third checkbox ☐ Fourth checkbox ☐ Fifth checkbox

```
<div class="list-group">
  <label class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="">
    First checkbox
  </label>
  <label class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="">
    Second checkbox
  </label>
  <label class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="">
    Third checkbox
  </label>
  <label class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="">
    Fourth checkbox
  </label>
  <label class="list-group-item">
    <input class="form-check-input me-1" type="checkbox" value="">
    Fifth checkbox
  </label>
</div>
```

Sass

Variables

```
$list-group-color:                 $gray-900;
$list-group-bg:                    $white;
$list-group-border-color:          rgba($black, .125);
$list-group-border-width:          $border-width;
$list-group-border-radius:         $border-radius;

$list-group-item-padding-y:        $spacer * .5;
```

```
$list-group-item-padding-x:        $spacer;
$list-group-item-bg-scale:         -80%;
$list-group-item-color-scale:       40%;

$list-group-hover-bg:              $gray-100;
$list-group-active-color:          $component-active-color;
$list-group-active-bg:             $component-active-bg;
$list-group-active-border-color:    $list-group-active-bg;

$list-group-disabled-color:        $gray-600;
$list-group-disabled-bg:           $list-group-bg;

$list-group-action-color:          $gray-700;
$list-group-action-hover-color:     $list-group-action-color;

$list-group-action-active-color:    $body-color;
$list-group-action-active-bg:       $gray-200;
```

Mixins

Used in combination with `$theme-colors` to generate the contextual variant classes for `.list-group-item`s.

```
@mixin list-group-item-variant($state, $background, $color) {
  .list-group-item-#{$state} {
    color: $color;
    background-color: $background;

    &.list-group-item-action {
      &:hover,
      &:focus {
        color: $color;
        background-color: shade-color($background, 10%);
      }

      &.active {
        color: $white;
        background-color: $color;
        border-color: $color;
      }
    }
  }
}
```

Loop

Loop that generates the modifier classes with the `list-group-item-variant()` mixin.

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

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

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

JavaScript behavior

Use the tab JavaScript plugin—include it individually or through the compiled bootstrap.js file—to extend our list group to create tabbable panes of local content.

HomeProfileMessagesSettings

Some placeholder content in a paragraph relating to "Home". And some more content, used here just to pad out and fill this tab panel. In production, you would obviously have more real content here. And not just text. It could be anything, really. Text, images, forms.

```
<div class="row">
  <div class="col-4">
    <div class="list-group" id="list-tab" role="tablist">
      <a class="list-group-item list-group-item-action active" id="list-home-list" data-bs-toggle="list" href="#list-home" role="tab" aria-controls="list-home">Home</a>
      <a class="list-group-item list-group-item-action" id="list-profile-list" data-bs-toggle="list" href="#list-profile" role="tab" aria-controls="list-profile">Profile</a>
      <a class="list-group-item list-group-item-action" id="list-messages-list" data-bs-toggle="list" href="#list-messages" role="tab" aria-controls="list-messages">Messages</a>
      <a class="list-group-item list-group-item-action" id="list-settings-list" data-bs-toggle="list" href="#list-settings" role="tab" aria-controls="list-settings">Settings</a>
    </div>
  </div>
  <div class="col-8">
    <div class="tab-content" id="nav-tabContent">
      <div class="tab-pane fade show active" id="list-home" role="tabpanel" aria-labelledby="list-home-list">...</div>
      <div class="tab-pane fade" id="list-profile" role="tabpanel" aria-labelledby="list-profile-list">...</div>
      <div class="tab-pane fade" id="list-messages" role="tabpanel" aria-labelledby="list-messages-list">...</div>
      <div class="tab-pane fade" id="list-settings" role="tabpanel" aria-labelledby="list-settings-list">...</div>
    </div>
  </div>
</div>
```

Using data attributes

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

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

  <!-- Tab panes -->
  <div class="tab-content">
    <div class="tab-pane active" id="home" role="tabpanel">...</div>
    <div class="tab-pane" id="profile" role="tabpanel">...</div>
    <div class="tab-pane" id="messages" role="tabpanel">...</div>
    <div class="tab-pane" id="settings" role="tabpanel">...</div>
  </div>
</div>
```

Via JavaScript

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

```
var triggerTabList = [].slice.call(document.querySelectorAll('#myTab a'))
triggerTabList.forEach(function (triggerEl) {
  var tabTrigger = new bootstrap.Tab(triggerEl)

  triggerEl.addEventListener('click', function (event) {
    event.preventDefault()
    tabTrigger.show()
  })
})
```

You can activate individual list item in several ways:

```
var triggerEl = document.querySelector('#myTab a[href="#profile"]')
bootstrap.Tab.getInstance(triggerEl).show() // Select tab by name
```

```
var triggerFirstTabEl = document.querySelector('#myTab li:first-child a')
bootstrap.Tab.getInstance(triggerFirstTabEl).show() // Select first tab
```

Fade effect

To make tabs panel fade in, add `.fade` to each `.tab-pane`. The first tab pane must also have `.show` to make the initial content visible.

```
<div class="tab-content">
  <div class="tab-pane fade show active" id="home" role="tabpanel">...</div>
  <div class="tab-pane fade" id="profile" role="tabpanel">...</div>
  <div class="tab-pane fade" id="messages" role="tabpanel">...</div>
  <div class="tab-pane fade" id="settings" role="tabpanel">...</div>
</div>
```

Methods

constructor

Activates a list item element and content container. Tab should have either a `data-bs-target` or an `href` targeting a container node in the DOM.

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

<div class="tab-content">
  <div class="tab-pane active" id="home" role="tabpanel">...</div>
  <div class="tab-pane" id="profile" role="tabpanel">...</div>
  <div class="tab-pane" id="messages" role="tabpanel">...</div>
  <div class="tab-pane" id="settings" role="tabpanel">...</div>
</div>

<script>
  var firstTabEl = document.querySelector('#myTab a:last-child')
  var firstTab = new bootstrap.Tab(firstTabEl)

  firstTab.show()
</script>
```

show

Selects the given list item and shows its associated pane. Any other list item that was previously selected becomes unselected and its associated pane is hidden. **Returns to**

the caller before the tab pane has actually been shown (for example, before the shown.bs.tab event occurs).

```
var someListItemEl = document.querySelector('#someListItem')
var tab = new bootstrap.Tab(someListItemEl)

tab.show()
```

dispose

Destroys an element's tab.

getInstance

Static method which allows you to get the tab instance associated with a DOM element

```
var triggerEl = document.querySelector('#trigger')
var tab = bootstrap.Tab.getInstance(triggerEl) // Returns a Bootstrap tab instance
```

getOrCreateInstance

Static method which allows you to get the tab instance associated with a DOM element, or create a new one in case it wasn't initialised

```
var triggerEl = document.querySelector('#trigger')
var tab = bootstrap.Tab.getOrCreateInstance(triggerEl) // Returns a Bootstrap tab instance
```

Events

When showing a new tab, the events fire in the following order:

1. hide.bs.tab (on the current active tab)
2. show.bs.tab (on the to-be-shown tab)
3. hidden.bs.tab (on the previous active tab, the same one as for the hide.bs.tab event)
4. shown.bs.tab (on the newly-active just-shown tab, the same one as for the show.bs.tab event)

If no tab was already active, the hide.bs.tab and hidden.bs.tab events will not be fired.

Event type	Description
show.bs.tab	This event fires on tab show, but before the new tab has been shown. Use event.target and event.relatedTarget to target the active tab and the previous active tab (if available) respectively.

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

```
var tabElms = document.querySelectorAll('a[data-bs-toggle="list"]')
tabElms.forEach(function(tabElm) {
  tabElm.addEventListener('shown.bs.tab', function (event) {
    event.target // newly activated tab
    event.relatedTarget // previous active tab
  })
}
```

Modal

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

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:

```
var myModal = document.getElementById('myModal')
var myInput = document.getElementById('myInput')

myModal.addEventListener('shown.bs.modal', function () {
  myInput.focus()
})
```

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

Keep reading for demos and usage guidelines.

Examples

Modal components

Below is a *static* modal example (meaning its position and display have been overridden). Included are the modal header, modal body (required for padding), and modal footer (optional). We ask that you include modal headers with dismiss actions whenever possible, or provide another explicit dismiss action.

Modal title

Modal body text goes here.

CloseSave changes

```
<div class="modal" tabindex="-1">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <p>Modal body text goes here.</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

Live demo

Toggle a working modal demo by clicking the button below. It will slide down and fade in from the top of the page.

Launch demo modal

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

<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-
hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

Static backdrop

When backdrop is set to static, the modal will not close when clicking outside it. Click the button below to try it.

Launch static backdrop modal

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#staticBackdrop">
  Launch static backdrop modal
</button>

<!-- Modal -->
<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1"
aria-labelledby="staticBackdropLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="staticBackdropLabel">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Understood</button>
      </div>
    </div>
  </div>
</div>
```

Scrolling long content

When modals become too long for the user's viewport or device, they scroll independent of the page itself. Try the demo below to see what we mean.

Launch demo modal

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

Launch demo modal

```
<!-- Scrollable modal -->
<div class="modal-dialog modal-dialog-scrollable">
  ...
</div>
```

Vertically centered

Add .modal-dialog-centered to .modal-dialog to vertically center the modal.

Vertically centered modal Vertically centered scrollable modal

```
<!-- Vertically centered modal -->
<div class="modal-dialog modal-dialog-centered">
  ...
</div>

<!-- Vertically centered scrollable modal -->
<div class="modal-dialog modal-dialog-centered modal-dialog-scrollable">
  ...
</div>
```

Tooltips and popovers

Tooltips and popovers can be placed within modals as needed. When modals are closed, any tooltips and popovers within are also automatically dismissed.

Launch demo modal

```
<div class="modal-body">
  <h5>Popover in a modal</h5>
  <p>This <a href="#" role="button" class="btn btn-secondary popover-test" title="Popover title" data-bs-content="Popover body content is set in this attribute.">button</a> triggers a popover on click.</p>
  <hr>
  <h5>Tooltips in a modal</h5>
  <p><a href="#" class="tooltip-test" title="Tooltip">This link</a> and <a href="#" class="tooltip-test" title="Tooltip">that link</a> have tooltips on hover.</p>
</div>
```

Using the grid

Utilize the Bootstrap grid system within a modal by nesting .container-fluid within the .modal-body. Then, use the normal grid system classes as you would anywhere else.

Launch demo modal

```
<div class="modal-body">
  <div class="container-fluid">
    <div class="row">
      <div class="col-md-4">.col-md-4</div>
      <div class="col-md-4 ms-auto">.col-md-4 .ms-auto</div>
    </div>
    <div class="row">
      <div class="col-md-3 ms-auto">.col-md-3 .ms-auto</div>
      <div class="col-md-2 ms-auto">.col-md-2 .ms-auto</div>
    </div>
    <div class="row">
      <div class="col-md-6 ms-auto">.col-md-6 .ms-auto</div>
    </div>
    <div class="row">
      <div class="col-sm-9">
        Level 1: .col-sm-9
        <div class="row">
          <div class="col-8 col-sm-6">
            Level 2: .col-8 .col-sm-6
          </div>
          <div class="col-4 col-sm-6">
            Level 2: .col-4 .col-sm-6
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Varying modal content

Have a bunch of buttons that all trigger the same modal with slightly different contents? Use event.relatedTarget and HTML data-bs-* attributes to vary the contents of the modal depending on which button was clicked.

Below is a live demo followed by example HTML and JavaScript. For more information, read the modal events docs for details on relatedTarget.

Open modal for @mdo Open modal for @fat Open modal for @getbootstrap

```
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#exampleModal" data-bs-whatever="@mdo">Open modal for @mdo</button>
```

```html
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#exampleModal" data-bs-whatever="@fat">Open modal for @fat</button>
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#exampleModal" data-bs-whatever="@getbootstrap">Open modal for @getbootstrap</button>

<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">New message</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <form>
          <div class="mb-3">
            <label for="recipient-name" class="col-form-label">Recipient:</label>
            <input type="text" class="form-control" id="recipient-name">
          </div>
          <div class="mb-3">
            <label for="message-text" class="col-form-label">Message:</label>
            <textarea class="form-control" id="message-text"></textarea>
          </div>
        </form>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Send message</button>
      </div>
    </div>
  </div>
</div>
```

```javascript
var exampleModal = document.getElementById('exampleModal')
exampleModal.addEventListener('show.bs.modal', function (event) {
  // Button that triggered the modal
  var button = event.relatedTarget
  // Extract info from data-bs-* attributes
  var recipient = button.getAttribute('data-bs-whatever')
  // If necessary, you could initiate an AJAX request here
  // and then do the updating in a callback.
  //
  // Update the modal's content.
  var modalTitle = exampleModal.querySelector('.modal-title')
  var modalBodyInput = exampleModal.querySelector('.modal-body input')

  modalTitle.textContent = 'New message to ' + recipient
  modalBodyInput.value = recipient
})
```

Toggle between modals

Toggle between multiple modals with some clever placement of the data-bs-target and data-bs-toggle attributes. For example, you could toggle a password reset modal from within an already open sign in modal. **Please note multiple modals cannot be open at the same time**—this method simply toggles between two separate modals.

Open first modal

```
<div class="modal fade" id="exampleModalToggle" aria-hidden="true" aria-
labelledby="exampleModalToggleLabel" tabindex="-1">
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalToggleLabel">Modal 1</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        Show a second modal and hide this one with the button below.
      </div>
      <div class="modal-footer">
        <button class="btn btn-primary" data-bs-target="#exampleModalToggle2" data-bs-toggle="modal" data-bs-
dismiss="modal">Open second modal</button>
      </div>
    </div>
  </div>
</div>
<div class="modal fade" id="exampleModalToggle2" aria-hidden="true" aria-
labelledby="exampleModalToggleLabel2" tabindex="-1">
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalToggleLabel2">Modal 2</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        Hide this modal and show the first with the button below.
      </div>
      <div class="modal-footer">
        <button class="btn btn-primary" data-bs-target="#exampleModalToggle" data-bs-toggle="modal" data-bs-
dismiss="modal">Back to first</button>
      </div>
    </div>
  </div>
</div>
<a class="btn btn-primary" data-bs-toggle="modal" href="#exampleModalToggle" role="button">Open first
modal</a>
```

Change animation

The $modal-fade-transform variable determines the transform state of .modal-dialog before the modal fade-in animation, the $modal-show-transform variable determines the transform of .modal-dialog at the end of the modal fade-in animation.

If you want for example a zoom-in animation, you can set $modal-fade-transform: scale(.8).

Remove animation

For modals that simply appear rather than fade in to view, remove the .fade class from your modal markup.

```
<div class="modal" tabindex="-1" aria-labelledby="..." aria-hidden="true">
  ...
</div>
```

Dynamic heights

If the height of a modal changes while it is open, you should call myModal.handleUpdate() to readjust the modal's position in case a scrollbar appears.

Accessibility

Be sure to add aria-labelledby="...", referencing the modal title, to .modal. Additionally, you may give a description of your modal dialog with aria-describedby on .modal. Note that you don't need to add role="dialog" since we already add it via JavaScript.

Embedding YouTube videos

Embedding YouTube videos in modals requires additional JavaScript not in Bootstrap to automatically stop playback and more. See this helpful Stack Overflow post for more information.

Optional sizes

Modals have three optional sizes, available via modifier classes to be placed on a .modal-dialog. These sizes kick in at certain breakpoints to avoid horizontal scrollbars on narrower viewports.

Size	Class	Modal max-width
Small	.modal-sm	300px
Default	None	500px
Large	.modal-lg	800px
Extra large	.modal-xl	1140px

Our default modal without modifier class constitutes the “medium” size modal.

Extra large modal Large modal Small modal

```
<div class="modal-dialog modal-xl">...</div>
<div class="modal-dialog modal-lg">...</div>
<div class="modal-dialog modal-sm">...</div>
```

Fullscreen Modal

Another override is the option to pop up a modal that covers the user viewport, available via modifier classes that are placed on a .modal-dialog.

Class	Availability
.modal-fullscreen	Always
.modal-fullscreen-sm-down	Below 576px
.modal-fullscreen-md-down	Below 768px
.modal-fullscreen-lg-down	Below 992px
.modal-fullscreen-xl-down	Below 1200px
.modal-fullscreen-xxl-down	Below 1400px

Full screen Full screen below sm Full screen below md Full screen below lg Full screen below xl Full screen below xxl

```
<!-- Full screen modal -->
<div class="modal-dialog modal-fullscreen-sm-down">
  ...
</div>
```

Sass

Variables

```
$modal-inner-padding:               $spacer;

$modal-footer-margin-between:       .5rem;

$modal-dialog-margin:               .5rem;
$modal-dialog-margin-y-sm-up:       1.75rem;

$modal-title-line-height:           $line-height-base;

$modal-content-color:               null;
$modal-content-bg:                  $white;
$modal-content-border-color:        rgba($black, .2);
$modal-content-border-width:        $border-width;
$modal-content-border-radius:       $border-radius-lg;
$modal-content-inner-border-radius: subtract($modal-content-border-radius, $modal-content-border-width);
$modal-content-box-shadow-xs:       $box-shadow-sm;
$modal-content-box-shadow-sm-up:    $box-shadow;

$modal-backdrop-bg:                 $black;
$modal-backdrop-opacity:            .5;
$modal-header-border-color:         $border-color;
$modal-footer-border-color:         $modal-header-border-color;
$modal-header-border-width:         $modal-content-border-width;
$modal-footer-border-width:         $modal-header-border-width;
$modal-header-padding-y:            $modal-inner-padding;
$modal-header-padding-x:            $modal-inner-padding;
$modal-header-padding:              $modal-header-padding-y $modal-header-padding-x; // Keep this for backwards
compatibility

$modal-sm:                          300px;
$modal-md:                          500px;
$modal-lg:                          800px;
$modal-xl:                          1140px;

$modal-fade-transform:              translate(0, -50px);
$modal-show-transform:              none;
$modal-transition:                  transform .3s ease-out;
$modal-scale-transform:             scale(1.02);
```

Loop

Responsive fullscreen modals are generated via the `$breakpoints` map and a loop in `scss/_modal.scss`.

```
@each $breakpoint in map-keys($grid-breakpoints) {
  $infix: breakpoint-infix($breakpoint, $grid-breakpoints);
  $postfix: if($infix != "", $infix + "-down", "");

  @include media-breakpoint-down($breakpoint) {
    .modal-fullscreen#{$postfix} {
      width: 100vw;
      max-width: none;
      height: 100%;
      margin: 0;

      .modal-content {
        height: 100%;
        border: 0;
        @include border-radius(0);
      }

      .modal-header {
        @include border-radius(0);
      }

      .modal-body {
        overflow-y: auto;
      }

      .modal-footer {
        @include border-radius(0);
      }
    }
  }
}
```

Usage

The modal plugin toggles your hidden content on demand, via data attributes or JavaScript. It also overrides default scrolling behavior and generates a `.modal-backdrop` to provide a click area for dismissing shown modals when clicking outside the modal.

Via data attributes

Activate a modal without writing JavaScript. Set `data-bs-toggle="modal"` on a controller element, like a button, along with a `data-bs-target="#foo"` or `href="#foo"` to target a specific modal to toggle.

```
<button type="button" data-bs-toggle="modal" data-bs-target="#myModal">Launch modal</button>
```

Via JavaScript

Create a modal with a single line of JavaScript:

```
var myModal = new bootstrap.Modal(document.getElementById('myModal'), options)
```

Options

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

Name	Type	Default	Description
backdrop	boolean or the string 'static'	true	Includes a modal-backdrop element. Alternatively, specify static for a backdrop which doesn't close the modal on click.
keyboard	boolean	true	Closes the modal when escape key is pressed
focus	boolean	true	Puts the focus on the modal when initialized.

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.

Passing options

Activates your content as a modal. Accepts an optional options object.

```
var myModal = new bootstrap.Modal(document.getElementById('myModal'), {
  keyboard: false
})
```

toggle

Manually toggles a modal. **Returns to the caller before the modal has actually been shown or hidden** (i.e. before the shown.bs.modal or hidden.bs.modal event occurs).

```
myModal.toggle()
```

show

Manually opens a modal. **Returns to the caller before the modal has actually been shown** (i.e. before the shown.bs.modal event occurs).

```
myModal.show()
```

Also, you can pass a DOM element as an argument that can be received in the modal events (as the relatedTarget property).

```
var modalToggle = document.getElementById('toggleMyModal') // relatedTarget
myModal.show(modalToggle)
```

hide

Manually hides a modal. **Returns to the caller before the modal has actually been hidden** (i.e. before the hidden.bs.modal event occurs).

```
myModal.hide()
```

handleUpdate

Manually readjust the modal's position if the height of a modal changes while it is open (i.e. in case a scrollbar appears).

```
myModal.handleUpdate()
```

dispose

Destroys an element's modal. (Removes stored data on the DOM element)

```
myModal.dispose()
```

getInstance

Static method which allows you to get the modal instance associated with a DOM element

```
var myModalEl = document.getElementById('myModal')
var modal = bootstrap.Modal.getInstance(myModalEl) // Returns a Bootstrap modal instance
```

getOrCreateInstance

Static method which allows you to get the modal instance associated with a DOM element, or create a new one in case it wasn't initialised

```
var myModalEl = document.querySelector('#myModal')
var modal = bootstrap.Modal.getOrCreateInstance(myModalEl) // Returns a Bootstrap modal instance
```

Events

Bootstrap’s modal class exposes a few events for hooking into modal functionality. All modal events are fired at the modal itself (i.e. at the `<div class="modal">`).

Event type	Description
show.bs.modal	This event fires immediately when the show instance method is called. If caused by a click, the clicked element is available as the relatedTarget property of the event.
shown.bs.modal	This event is fired when the modal has been made visible to the user (will wait for CSS transitions to complete). If caused by a click, the clicked element is available as the relatedTarget property of the event.
hide.bs.modal	This event is fired immediately when the hide instance method has been called.
hidden.bs.modal	This event is fired when the modal has finished being hidden from the user (will wait for CSS transitions to complete).
hidePrevented.bs.modal	This event is fired when the modal is shown, its backdrop is static and a click outside the modal or an escape key press is performed with the keyboard option or data-bs-keyboard set to false.

```
var myModalEl = document.getElementById('myModal')
myModalEl.addEventListener('hidden.bs.modal', function (event) {
  // do something...
})
```

Navs and tabs

Documentation and examples for how to use Bootstrap's included navigation components.

Base nav

Navigation available in Bootstrap share general markup and styles, from the base .nav class to the active and disabled states. Swap modifier classes to switch between each style.

The base .nav component is built with flexbox and provide a strong foundation for building all types of navigation components. It includes some style overrides (for working with lists), some link padding for larger hit areas, and basic disabled styling.

The base .nav component does not include any .active state. The following examples include the class, mainly to demonstrate that this particular class does not trigger any special styling.

To convey the active state to assistive technologies, use the aria-current attribute — using the page value for current page, or true for the current item in a set.

- Active
- Link
- Link
- Disabled

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

Classes are used throughout, so your markup can be super flexible. Use <ul>s like above, <ol> if the order of your items is important, or roll your own with a <nav> element.

Because the .nav uses display: flex, the nav links behave the same as nav items would, but without the extra markup.

ActiveLinkLinkDisabled

```
<nav class="nav">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
</nav>
```

Available styles

Change the style of .navs component with modifiers and utilities. Mix and match as needed, or build your own.

Horizontal alignment

Change the horizontal alignment of your nav with flexbox utilities. By default, navs are left-aligned, but you can easily change them to center or right aligned.

Centered with .justify-content-center:

- Active
- Link
- Link
- Disabled

```
<ul class="nav justify-content-center">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

Right-aligned with .justify-content-end:

- Active
- Link
- Link
- Disabled

```
<ul class="nav justify-content-end">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

Vertical

Stack your navigation by changing the flex item direction with the `.flex-column` utility. Need to stack them on some viewports but not others? Use the responsive versions (e.g., `.flex-sm-column`).

- Active
- Link
- Link
- Disabled

```
<ul class="nav flex-column">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

As always, vertical navigation is possible without `<ul>`s, too.

ActiveLinkLinkDisabled

```
<nav class="nav flex-column">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
</nav>
```

Tabs

Takes the basic nav from above and adds the `.nav-tabs` class to generate a tabbed interface. Use them to create tabbable regions with our tab JavaScript plugin.

- Active
- Link
- Link
- Disabled

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

Pills

Take that same HTML, but use `.nav-pills` instead:

- Active
- Link
- Link
- Disabled

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

```
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

Fill and justify

Force your .nav's contents to extend the full available width one of two modifier classes. To proportionately fill all available space with your .nav-items, use .nav-fill. Notice that all horizontal space is occupied, but not every nav item has the same width.

- Active
- Much longer nav link
- Link
- Disabled

```
<ul class="nav nav-pills nav-fill">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Much longer nav link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

When using a <nav>-based navigation, you can safely omit .nav-item as only .nav-link is required for styling <a> elements.

ActiveMuch longer nav linkLinkDisabled

```
<nav class="nav nav-pills nav-fill">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Much longer nav link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
</nav>
```
For equal-width elements, use .nav-justified. All horizontal space will be occupied by nav links, but unlike the .nav-fill above, every nav item will be the same width.

- Active
- Much longer nav link
- Link
- Disabled

```
<ul class="nav nav-pills nav-justified">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Much longer nav link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```
Similar to the .nav-fill example using a <nav>-based navigation.

ActiveMuch longer nav linkLinkDisabled

```
<nav class="nav nav-pills nav-justified">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Much longer nav link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
</nav>
```

Working with flex utilities

If you need responsive nav variations, consider using a series of flexbox utilities. While more verbose, these utilities offer greater customization across responsive breakpoints. In the example below, our nav will be stacked on the lowest breakpoint, then adapt to a horizontal layout that fills the available width starting from the small breakpoint.

ActiveLonger nav linkLinkDisabled

```
<nav class="nav nav-pills flex-column flex-sm-row">
  <a class="flex-sm-fill text-sm-center nav-link active" aria-current="page" href="#">Active</a>
  <a class="flex-sm-fill text-sm-center nav-link" href="#">Longer nav link</a>
  <a class="flex-sm-fill text-sm-center nav-link" href="#">Link</a>
  <a class="flex-sm-fill text-sm-center nav-link disabled" 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 `<ul>`, or wrap a `<nav>` element around the whole navigation. Do not add the role to the `<ul>` itself, as this would prevent it from being announced as an actual list by assistive technologies.

Note that navigation bars, even if visually styled as tabs with the `.nav-tabs` class, should **not** be given `role="tablist"`, `role="tab"` or `role="tabpanel"` attributes. These are only appropriate for dynamic tabbed interfaces, as described in the WAI ARIA Authoring Practices. See JavaScript behavior for dynamic tabbed interfaces in this section for an example. The `aria-current` attribute is not necessary on dynamic tabbed interfaces since our JavaScript handles the selected state by adding `aria-selected="true"` on the active tab.

Using dropdowns

Add dropdown menus with a little extra HTML and the dropdowns JavaScript plugin.

Tabs with dropdowns

- Active
- Dropdown
- Link
- Disabled

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown" href="#" role="button" aria-
expanded="false">Dropdown</a>
    <ul class="dropdown-menu">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
      <li><hr class="dropdown-divider"></li>
      <li><a class="dropdown-item" href="#">Separated link</a></li>
    </ul>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

Pills with dropdowns

- Active
- Dropdown
- Link
- Disabled

```
<ul class="nav nav-pills">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown" href="#" role="button" aria-
expanded="false">Dropdown</a>
    <ul class="dropdown-menu">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
      <li><hr class="dropdown-divider"></li>
      <li><a class="dropdown-item" href="#">Separated link</a></li>
    </ul>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

Sass

Variables

```
$nav-link-padding-y:                .5rem;
$nav-link-padding-x:                1rem;
$nav-link-font-size:               null;
```

```
$nav-link-font-weight:            null;
$nav-link-color:                  $link-color;
$nav-link-hover-color:            $link-hover-color;
$nav-link-transition:             color .15s ease-in-out, background-color .15s ease-in-out, border-color .15s ease-in-
out;
$nav-link-disabled-color:         $gray-600;

$nav-tabs-border-color:           $gray-300;
$nav-tabs-border-width:           $border-width;
$nav-tabs-border-radius:          $border-radius;
$nav-tabs-link-hover-border-color:  $gray-200 $gray-200 $nav-tabs-border-color;
$nav-tabs-link-active-color:      $gray-700;
$nav-tabs-link-active-bg:         $body-bg;
$nav-tabs-link-active-border-color: $gray-300 $gray-300 $nav-tabs-link-active-bg;

$nav-pills-border-radius:         $border-radius;
$nav-pills-link-active-color:     $component-active-color;
$nav-pills-link-active-bg:        $component-active-bg;
```

JavaScript behavior

Use the tab JavaScript plugin—include it individually or through the compiled `bootstrap.js` file—to extend our navigational tabs and pills to create tabbable panes of local content.

Dynamic tabbed interfaces, as described in the WAI ARIA Authoring Practices, require `role="tablist"`, `role="tab"`, `role="tabpanel"`, and additional `aria-` attributes in order to convey their structure, functionality and current state to users of assistive technologies (such as screen readers). As a best practice, we recommend using `<button>` elements for the tabs, as these are controls that trigger a dynamic change, rather than links that navigate to a new page or location.

Note that dynamic tabbed interfaces should *not* contain dropdown menus, as this causes both usability and accessibility issues. From a usability perspective, the fact that the currently displayed tab's trigger element is not immediately visible (as it's inside the closed dropdown menu) can cause confusion. From an accessibility point of view, there is currently no sensible way to map this sort of construct to a standard WAI ARIA pattern, meaning that it cannot be easily made understandable to users of assistive technologies.

- Home
- Profile
- Contact

This is some placeholder content the Home tab's associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other .nav-powered navigation.

```
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="home-tab" data-bs-toggle="tab" data-bs-target="#home" type="button"
role="tab" aria-controls="home" aria-selected="true">Home</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="profile-tab" data-bs-toggle="tab" data-bs-target="#profile" type="button"
role="tab" aria-controls="profile" aria-selected="false">Profile</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="contact-tab" data-bs-toggle="tab" data-bs-target="#contact" type="button"
role="tab" aria-controls="contact" aria-selected="false">Contact</button>
  </li>
</ul>
<div class="tab-content" id="myTabContent">
  <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="home-tab">...</div>
  <div class="tab-pane fade" id="profile" role="tabpanel" aria-labelledby="profile-tab">...</div>
  <div class="tab-pane fade" id="contact" role="tabpanel" aria-labelledby="contact-tab">...</div>
</div>
```

To help fit your needs, this works with <ul>-based markup, as shown above, or with any arbitrary "roll your own" markup. Note that if you're using <nav>, you shouldn't add role="tablist" directly to it, as this would override the element's native role as a navigation landmark. Instead, switch to an alternative element (in the example below, a simple <div>) and wrap the <nav> around it.

HomeProfileContact

This is some placeholder content the Home tab's associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other .nav-powered navigation.

```
<nav>
  <div class="nav nav-tabs" id="nav-tab" role="tablist">
    <button class="nav-link active" id="nav-home-tab" data-bs-toggle="tab" data-bs-target="#nav-home"
type="button" role="tab" aria-controls="nav-home" aria-selected="true">Home</button>
    <button class="nav-link" id="nav-profile-tab" data-bs-toggle="tab" data-bs-target="#nav-profile" type="button"
role="tab" aria-controls="nav-profile" aria-selected="false">Profile</button>
    <button class="nav-link" id="nav-contact-tab" data-bs-toggle="tab" data-bs-target="#nav-contact" type="button"
role="tab" aria-controls="nav-contact" aria-selected="false">Contact</button>
  </div>
</nav>
<div class="tab-content" id="nav-tabContent">
  <div class="tab-pane fade show active" id="nav-home" role="tabpanel" aria-labelledby="nav-home-tab">...</div>
  <div class="tab-pane fade" id="nav-profile" role="tabpanel" aria-labelledby="nav-profile-tab">...</div>
  <div class="tab-pane fade" id="nav-contact" role="tabpanel" aria-labelledby="nav-contact-tab">...</div>
</div>
```

The tabs plugin also works with pills.

- Home
- Profile
- Contact

This is some placeholder content the Home tab's associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other `.nav`-powered navigation.

```
<ul class="nav nav-pills mb-3" id="pills-tab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="pills-home-tab" data-bs-toggle="pill" data-bs-target="#pills-home" 
type="button" role="tab" aria-controls="pills-home" aria-selected="true">Home</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="pills-profile-tab" data-bs-toggle="pill" data-bs-target="#pills-profile" type="button" 
role="tab" aria-controls="pills-profile" aria-selected="false">Profile</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="pills-contact-tab" data-bs-toggle="pill" data-bs-target="#pills-contact" 
type="button" role="tab" aria-controls="pills-contact" aria-selected="false">Contact</button>
  </li>
</ul>
<div class="tab-content" id="pills-tabContent">
  <div class="tab-pane fade show active" id="pills-home" role="tabpanel" aria-labelledby="pills-home-
tab">...</div>
  <div class="tab-pane fade" id="pills-profile" role="tabpanel" aria-labelledby="pills-profile-tab">...</div>
  <div class="tab-pane fade" id="pills-contact" role="tabpanel" aria-labelledby="pills-contact-tab">...</div>
</div>
```

And with vertical pills.

HomeProfileMessagesSettings

This is some placeholder content the Home tab's associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other `.nav`-powered navigation.

```
<div class="d-flex align-items-start">
  <div class="nav flex-column nav-pills me-3" id="v-pills-tab" role="tablist" aria-orientation="vertical">
```

```
    <button class="nav-link active" id="v-pills-home-tab" data-bs-toggle="pill" data-bs-target="#v-pills-home"
type="button" role="tab" aria-controls="v-pills-home" aria-selected="true">Home</button>
    <button class="nav-link" id="v-pills-profile-tab" data-bs-toggle="pill" data-bs-target="#v-pills-profile"
type="button" role="tab" aria-controls="v-pills-profile" aria-selected="false">Profile</button>
    <button class="nav-link" id="v-pills-messages-tab" data-bs-toggle="pill" data-bs-target="#v-pills-messages"
type="button" role="tab" aria-controls="v-pills-messages" aria-selected="false">Messages</button>
    <button class="nav-link" id="v-pills-settings-tab" data-bs-toggle="pill" data-bs-target="#v-pills-settings"
type="button" role="tab" aria-controls="v-pills-settings" aria-selected="false">Settings</button>
  </div>
  <div class="tab-content" id="v-pills-tabContent">
    <div class="tab-pane fade show active" id="v-pills-home" role="tabpanel" aria-labelledby="v-pills-home-
tab">...</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>
```

Using data attributes

You can activate a tab or pill navigation without writing any JavaScript by simply specifying `data-bs-toggle="tab"` or `data-bs-toggle="pill"` on an element. Use these data attributes on `.nav-tabs` or `.nav-pills`.

```
<!-- Nav tabs -->
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="home-tab" data-bs-toggle="tab" data-bs-target="#home" type="button"
role="tab" aria-controls="home" aria-selected="true">Home</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="profile-tab" data-bs-toggle="tab" data-bs-target="#profile" type="button"
role="tab" aria-controls="profile" aria-selected="false">Profile</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="messages-tab" data-bs-toggle="tab" data-bs-target="#messages" type="button"
role="tab" aria-controls="messages" aria-selected="false">Messages</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="settings-tab" data-bs-toggle="tab" data-bs-target="#settings" type="button"
role="tab" aria-controls="settings" aria-selected="false">Settings</button>
  </li>
</ul>

<!-- Tab panes -->
<div class="tab-content">
  <div class="tab-pane active" id="home" role="tabpanel" aria-labelledby="home-tab">...</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):

```
var triggerTabList = [].slice.call(document.querySelectorAll('#myTab a'))
triggerTabList.forEach(function (triggerEl) {
  var tabTrigger = new bootstrap.Tab(triggerEl)

  triggerEl.addEventListener('click', function (event) {
    event.preventDefault()
    tabTrigger.show()
  })
})
```

You can activate individual tabs in several ways:

```
var triggerEl = document.querySelector('#myTab a[href="#profile"]')
bootstrap.Tab.getInstance(triggerEl).show() // Select tab by name

var triggerFirstTabEl = document.querySelector('#myTab li:first-child a')
bootstrap.Tab.getInstance(triggerFirstTabEl).show() // Select first tab
```

Fade effect

To make tabs fade in, add `.fade` to each `.tab-pane`. The first tab pane must also have `.show` to make the initial content visible.

```
<div class="tab-content">
  <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="home-tab">...</div>
  <div class="tab-pane fade" id="profile" role="tabpanel" aria-labelledby="profile-tab">...</div>
  <div class="tab-pane fade" id="messages" role="tabpanel" aria-labelledby="messages-tab">...</div>
  <div class="tab-pane fade" id="settings" role="tabpanel" aria-labelledby="settings-tab">...</div>
</div>
```

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.

constructor

Activates a tab element and content container. Tab should have either a `data-bs-target` or, if using a link, an `href` attribute, targeting a container node in the DOM.

```
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="home-tab" data-bs-toggle="tab" data-bs-target="#home" type="button"
role="tab" aria-controls="home" aria-selected="true">Home</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="profile-tab" data-bs-toggle="tab" data-bs-target="#profile" type="button"
role="tab" aria-controls="profile" aria-selected="false">Profile</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="messages-tab" data-bs-toggle="tab" data-bs-target="#messages" type="button"
role="tab" aria-controls="messages" aria-selected="false">Messages</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="settings-tab" data-bs-toggle="tab" data-bs-target="#settings" type="button"
role="tab" aria-controls="settings" aria-selected="false">Settings</button>
  </li>
</ul>

<div class="tab-content">
  <div class="tab-pane active" id="home" role="tabpanel" aria-labelledby="home-tab">...</div>
  <div class="tab-pane" id="profile" role="tabpanel" aria-labelledby="profile-tab">...</div>
  <div class="tab-pane" id="messages" role="tabpanel" aria-labelledby="messages-tab">...</div>
  <div class="tab-pane" id="settings" role="tabpanel" aria-labelledby="settings-tab">...</div>
</div>

<script>
  var firstTabEl = document.querySelector('#myTab li:last-child a')
  var firstTab = new bootstrap.Tab(firstTabEl)

  firstTab.show()
</script>
```

show

Selects the given tab and shows its associated pane. Any other tab that was previously selected becomes unselected and its associated pane is hidden. **Returns to the caller before the tab pane has actually been shown** (i.e. before the `shown.bs.tab` event occurs).

```
  var someTabTriggerEl = document.querySelector('#someTabTrigger')
  var tab = new bootstrap.Tab(someTabTriggerEl)

  tab.show()
```

dispose

Destroys an element's tab.

getInstance

Static method which allows you to get the tab instance associated with a DOM element

```
var triggerEl = document.querySelector('#trigger')
var tab = bootstrap.Tab.getInstance(triggerEl) // Returns a Bootstrap tab instance
```

getOrCreateInstance

Static method which allows you to get the tab instance associated with a DOM element, or create a new one in case it wasn't initialised

```
var triggerEl = document.querySelector('#trigger')
var tab = bootstrap.Tab.getOrCreateInstance(triggerEl) // Returns a Bootstrap tab instance
```

Events

When showing a new tab, the events fire in the following order:

1. `hide.bs.tab` (on the current active tab)
2. `show.bs.tab` (on the to-be-shown tab)
3. `hidden.bs.tab` (on the previous active tab, the same one as for the `hide.bs.tab` event)
4. `shown.bs.tab` (on the newly-active just-shown tab, the same one as for the `show.bs.tab` event)

If no tab was already active, then the `hide.bs.tab` and `hidden.bs.tab` events will not be fired.

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

Event type	Description
hidden.bs.tab	This event fires after a new tab is shown (and thus the previous active tab is hidden). Use event.target and event.relatedTarget to target the previous active tab and the new active tab, respectively.

```
var tabEl = document.querySelector('button[data-bs-toggle="tab"]')
tabEl.addEventListener('shown.bs.tab', function (event) {
  event.target // newly activated tab
  event.relatedTarget // previous active tab
})
```

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|-xxl}` for responsive collapsing and color scheme classes.
- Navbars and their contents are fluid by default. Change the container to limit their horizontal width in different ways.
- Use our spacing and flex utility classes for controlling spacing and alignment within navbars.
- Navbars are responsive by default, but you can easily modify them to change that. Responsive behavior depends on our Collapse JavaScript plugin.
- Ensure accessibility by using a `<nav>` element or, if using a more generic element such as a `<div>`, add a `role="navigation"` to every navbar to explicitly identify it as a landmark region for users of assistive technologies.
- Indicate the current item by using `aria-current="page"` for the current page or `aria-current="true"` for the current item in a set.

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

Supported content

Navbars come with built-in support for a handful of sub-components. Choose from the following as needed:

- `.navbar-brand` for your company, product, or project name.
- `.navbar-nav` for a full-height and lightweight navigation (including support for dropdowns).
- `.navbar-toggler` for use with our collapse plugin and other navigation toggling behaviors.
- Flex and spacing utilities for any form controls and actions.
- `.navbar-text` for adding vertically centered strings of text.
- `.collapse.navbar-collapse` for grouping and hiding navbar contents by a parent breakpoint.
- Add an optional `.navbar-scroll` to set a `max-height` and scroll expanded navbar content.

Here's an example of all the sub-components included in a responsive light-themed navbar that automatically collapses at the lg (large) breakpoint.

Navbar

- Home
- Link
- Dropdown
- Disabled

Search

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-
toggle="dropdown" aria-expanded="false">
            Dropdown
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><hr class="dropdown-divider"></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
        </li>
      </ul>
      <form class="d-flex">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

This example uses background (bg-light) and spacing (my-2, my-lg-0, me-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.

Text

Add your text within an element with the .navbar-brand class.

Navbar

Navbar

```
<!-- As a link -->
<nav class="navbar navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
  </div>
</nav>

<!-- As a heading -->
<nav class="navbar navbar-light bg-light">
  <div class="container-fluid">
    <span class="navbar-brand mb-0 h1">Navbar</span>
  </div>
</nav>
```

Image

You can replace the text within the .navbar-brand with an <img>.

```
<nav class="navbar navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="#">
      <img src="/docs/5.0/assets/brand/bootstrap-logo.svg" alt="" width="30" height="24">
    </a>
  </div>
</nav>
```

Image and text

You can also make use of some additional utilities to add an image and text at the same time. Note the addition of .d-inline-block and .align-text-top on the <img>.

 Bootstrap

```
<nav class="navbar navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">
      <img src="/docs/5.0/assets/brand/bootstrap-logo.svg" alt="" width="30" height="24" class="d-inline-block
align-text-top">
      Bootstrap
    </a>
  </div>
</nav>
```

Nav

Navbar navigation links build on our `.nav` options with their own modifier class and require the use of toggler classes for proper responsive styling. **Navigation in navbars will also grow to occupy as much horizontal space as possible** to keep your navbar contents securely aligned.

Add the `.active` class on `.nav-link` to indicate the current page.

Please note that you should also add the `aria-current` attribute on the active `.nav-link`.

Navbar

- Home
- Features
- Pricing
- Disabled

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Features</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Pricing</a>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

And because we use classes for our navs, you can avoid the list-based approach entirely if you like.

[Navbar]

[HomeFeaturesPricingDisabled]

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
 <div class="container-fluid">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup"
aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
   <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
   <div class="navbar-nav">
    <a class="nav-link active" aria-current="page" href="#">Home</a>
    <a class="nav-link" href="#">Features</a>
    <a class="nav-link" href="#">Pricing</a>
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
   </div>
  </div>
 </div>
</nav>
```

You can also use dropdowns in your navbar. Dropdown menus require a wrapping element for positioning, so be sure to use separate and nested elements for `.nav-item` and `.nav-link` as shown below.

[Navbar]

- [Home]
- [Features]
- [Pricing]
- [Dropdown link]

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
 <div class="container-fluid">
  <a class="navbar-brand" href="#">Navbar</a>
```

```
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavDropdown"
aria-controls="navbarNavDropdown" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavDropdown">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Features</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Pricing</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink" role="button" data-bs-
toggle="dropdown" aria-expanded="false">
            Dropdown link
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Forms

Place various form controls and components within a navbar:

Search

```
<nav class="navbar navbar-light bg-light">
  <div class="container-fluid">
    <form class="d-flex">
      <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success" type="submit">Search</button>
    </form>
  </div>
</nav>
```

Immediate child elements of `.navbar` use flex layout and will default to `justify-content: space-between`. Use additional flex utilities as needed to adjust this behavior.

Navbar

Search

```
<nav class="navbar navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand">Navbar</a>
    <form class="d-flex">
      <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success" type="submit">Search</button>
    </form>
  </div>
</nav>
```

Input groups work, too. If your navbar is an entire form, or mostly a form, you can use the `<form>` element as the container and save some HTML.

```
<nav class="navbar navbar-light bg-light">
  <form class="container-fluid">
    <div class="input-group">
      <span class="input-group-text" id="basic-addon1">@</span>
      <input type="text" class="form-control" placeholder="Username" aria-label="Username" aria-
describedby="basic-addon1">
    </div>
  </form>
</nav>
```

Various buttons are supported as part of these navbar forms, too. This is also a great reminder that vertical alignment utilities can be used to align different sized elements.

Main buttonSmaller button

```
<nav class="navbar navbar-light bg-light">
  <form class="container-fluid justify-content-start">
    <button class="btn btn-outline-success me-2" type="button">Main button</button>
    <button class="btn btn-sm btn-outline-secondary" type="button">Smaller button</button>
  </form>
</nav>
```

Text

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

Navbar text with an inline element

```
<nav class="navbar navbar-light bg-light">
```

```
  <div class="container-fluid">
    <span class="navbar-text">
      Navbar text with an inline element
    </span>
  </div>
</nav>
```

Mix and match with other components and utilities as needed.

[Navbar w/ text](#)

- [Home](#)
- [Features](#)
- [Pricing](#)

Navbar text with an inline element

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar w/ text</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarText" aria-
controls="navbarText" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarText">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Features</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Pricing</a>
        </li>
      </ul>
      <span class="navbar-text">
        Navbar text with an inline element
      </span>
    </div>
  </div>
</nav>
```

Color schemes

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.

-
- Features
- Pricing
- About

Search

-
- Features
- Pricing
- About

Search
Navbar

- Home
- Features
- Pricing
- About

Search

```
<nav class="navbar navbar-dark bg-dark">
  <!-- Navbar content -->
</nav>

<nav class="navbar navbar-dark bg-primary">
  <!-- Navbar content -->
</nav>

<nav class="navbar navbar-light" style="background-color: #e3f2fd;">
  <!-- Navbar content -->
</nav>
```

Containers

Although it's not required, you can wrap a navbar in a .container to center it on a page–though note that an inner container is still required. Or you can add a container inside the .navbar to only center the contents of a fixed or static top navbar.

Navbar

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

```
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="container-fluid">
      <a class="navbar-brand" href="#">Navbar</a>
    </div>
  </nav>
</div>
```

Use any of the responsive containers to change how wide the content in your navbar is presented.

Navbar

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-md">
    <a class="navbar-brand" href="#">Navbar</a>
  </div>
</nav>
```

Placement

Use our position utilities to place navbars in non-static positions. Choose from fixed to the top, fixed to the bottom, 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**.

Default

```
<nav class="navbar navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Default</a>
  </div>
</nav>
```

Fixed top

```
<nav class="navbar fixed-top navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Fixed top</a>
  </div>
</nav>
```

Fixed bottom

```
<nav class="navbar fixed-bottom navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Fixed bottom</a>
  </div>
</nav>
```

Sticky top

```
<nav class="navbar sticky-top navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Sticky top</a>
  </div>
</nav>
```

Scrolling

Add `.navbar-nav-scroll` to a `.navbar-nav` (or other navbar sub-component) to enable vertical scrolling within the toggleable contents of a collapsed navbar. By default, scrolling kicks in at `75vh` (or 75% of the viewport height), but you can override that with the local CSS custom property `--bs-navbar-height` or custom styles. At larger viewports when the navbar is expanded, content will appear as it does in a default navbar.

Please note that this behavior comes with a potential drawback of `overflow`—when setting `overflow-y: auto` (required to scroll the content here), `overflow-x` is the equivalent of `auto`, which will crop some horizontal content.

Here's an example navbar using `.navbar-nav-scroll` with `style="--bs-scroll-height: 100px;"`, with some extra margin utilities for optimum spacing.

Navbar scroll

- Home
- Link
- Link
- Link

Search

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar scroll</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarScroll" aria-
controls="navbarScroll" aria-expanded="false" aria-label="Toggle navigation">
```

```
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarScroll">
      <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 100px;">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarScrollingDropdown" role="button" data-bs-
toggle="dropdown" aria-expanded="false">
            Link
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarScrollingDropdown">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><hr class="dropdown-divider"></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Link</a>
        </li>
      </ul>
      <form class="d-flex">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

Responsive behaviors

Navbars can use .navbar-toggler, .navbar-collapse, and .navbar-expand{-sm|-md|-lg|-xl|-xxl} classes to determine when their content collapses behind a button. In combination with other utilities, you can easily choose when to show or hide particular elements.

For navbars that never collapse, add the .navbar-expand class on the navbar. For navbars that always collapse, don't add any .navbar-expand class.

Toggler

Navbar togglers are left-aligned by default, but should they follow a sibling element like a .navbar-brand, they'll automatically be aligned to the far right. Reversing your markup will reverse the placement of the toggler. Below are examples of different toggle styles.

With no .navbar-brand shown at the smallest breakpoint:

Hidden brand

- Home
- Link
- Disabled

Search

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarTogglerDemo01" aria-controls="navbarTogglerDemo01" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="#">Hidden brand</a>
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
        </li>
      </ul>
      <form class="d-flex">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

With a brand name shown on the left and toggler on the right:

Navbar

- Home
- Link
- Disabled

Search

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
```

```
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarTogglerDemo02" aria-controls="navbarTogglerDemo02" aria-expanded="false" aria-label="Toggle
navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarTogglerDemo02">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
        </li>
      </ul>
      <form class="d-flex">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

With a toggler on the left and brand name on the right:

Navbar

- Home
- Link
- Disabled

Search

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarTogglerDemo03" aria-controls="navbarTogglerDemo03" aria-expanded="false" aria-label="Toggle
navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <a class="navbar-brand" href="#">Navbar</a>
    <div class="collapse navbar-collapse" id="navbarTogglerDemo03">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
        </li>
      </ul>
      <form class="d-flex">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

External content

Sometimes you want to use the collapse plugin to trigger a container element for content that structurally sits outside of the `.navbar` . Because our plugin works on the `id` and `data-bs-target` matching, that's easily done!

```
<div class="collapse" id="navbarToggleExternalContent">
 <div class="bg-dark p-4">
  <h5 class="text-white h4">Collapsed content</h5>
  <span class="text-muted">Toggleable via the navbar brand.</span>
 </div>
</div>
<nav class="navbar navbar-dark bg-dark">
 <div class="container-fluid">
  <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarToggleExternalContent" aria-controls="navbarToggleExternalContent" aria-expanded="false" aria-
label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
 </div>
</nav>
```

When you do this, we recommend including additional JavaScript to move the focus programmatically to the container when it is opened. Otherwise, keyboard users and users of assistive technologies will likely have a hard time finding the newly revealed content - particularly if the container that was opened comes *before* the toggler in the document's structure. We also recommend making sure that the toggler has the `aria-controls` attribute, pointing to the `id` of the content container. In theory, this allows assistive technology users to jump directly from the toggler to the container it controls– but support for this is currently quite patchy.

Sass

Variables

```
$navbar-padding-y:                  $spacer * .5;
$navbar-padding-x:                  null;

$navbar-nav-link-padding-x:         .5rem;

$navbar-brand-font-size:            $font-size-lg;
// Compute the navbar-brand padding-y so the navbar-brand will have the same height as navbar-text and nav-link
$nav-link-height:                   $font-size-base * $line-height-base + $nav-link-padding-y * 2;
$navbar-brand-height:               $navbar-brand-font-size * $line-height-base;
$navbar-brand-padding-y:            ($nav-link-height - $navbar-brand-height) * .5;
$navbar-brand-margin-end:           1rem;

$navbar-toggler-padding-y:          .25rem;
$navbar-toggler-padding-x:          .75rem;
$navbar-toggler-font-size:          $font-size-lg;
$navbar-toggler-border-radius:      $btn-border-radius;
$navbar-toggler-focus-width:        $btn-focus-width;
$navbar-toggler-transition:         box-shadow .15s ease-in-out;


$navbar-dark-color:                 rgba($white, .55);
$navbar-dark-hover-color:           rgba($white, .75);
$navbar-dark-active-color:          $white;
$navbar-dark-disabled-color:        rgba($white, .25);
$navbar-dark-toggler-icon-bg:       url("data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg' viewBox='0
0 30 30'><path stroke='#{$navbar-dark-color}' stroke-linecap='round' stroke-miterlimit='10' stroke-width='2' d='M4
7h22M4 15h22M4 23h22'/></svg>");
$navbar-dark-toggler-border-color:  rgba($white, .1);

$navbar-light-color:                rgba($black, .55);
$navbar-light-hover-color:          rgba($black, .7);
$navbar-light-active-color:         rgba($black, .9);
$navbar-light-disabled-color:       rgba($black, .3);
$navbar-light-toggler-icon-bg:      url("data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg' viewBox='0
0 30 30'><path stroke='#{$navbar-light-color}' stroke-linecap='round' stroke-miterlimit='10' stroke-width='2' d='M4
7h22M4 15h22M4 23h22'/></svg>");
$navbar-light-toggler-border-color: rgba($black, .1);

$navbar-light-brand-color:                $navbar-light-active-color;
$navbar-light-brand-hover-color:          $navbar-light-active-color;
$navbar-dark-brand-color:                 $navbar-dark-active-color;
$navbar-dark-brand-hover-color:           $navbar-dark-active-color;
```

Loop

Responsive navbar expand/collapse classes (e.g., .navbar-expand-lg) are combined with the $breakpoints map and generated through a loop in scss/_navbar.scss.

```
// Generate series of `.navbar-expand-*` responsive classes for configuring
// where your navbar collapses.
.navbar-expand {
  @each $breakpoint in map-keys($grid-breakpoints) {
    $next: breakpoint-next($breakpoint, $grid-breakpoints);
    $infix: breakpoint-infix($next, $grid-breakpoints);

    // stylelint-disable-next-line scss/selector-no-union-class-name
    &#{$infix} {
      @include media-breakpoint-up($next) {
        flex-wrap: nowrap;
        justify-content: flex-start;

        .navbar-nav {
          flex-direction: row;

          .dropdown-menu {
            position: absolute;
          }

          .nav-link {
            padding-right: $navbar-nav-link-padding-x;
            padding-left: $navbar-nav-link-padding-x;
          }
        }

        .navbar-nav-scroll {
          overflow: visible;
        }

        .navbar-collapse {
          display: flex !important; // stylelint-disable-line declaration-no-important
          flex-basis: auto;
        }

        .navbar-toggler {
          display: none;
        }
      }
    }
  }
}
```

Pagination

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

Overview

We use a large block of connected links for our pagination, making links hard to miss and easily scalable—all while providing large hit areas. Pagination is built with list HTML elements so screen readers can announce the number of available links. Use a wrapping <nav> element to identify it as a navigation section to screen readers and other assistive technologies.

In addition, as pages likely have more than one such navigation section, it's advisable to provide a descriptive aria-label for the <nav> to reflect its purpose. For example, if the pagination component is used to navigate between a set of search results, an appropriate label could be aria-label="Search results pages".

- Previous
- 1
- 2
- 3
- Next

```
<nav aria-label="Page navigation example">
  <ul class="pagination">
    <li class="page-item"><a class="page-link" href="#">Previous</a></li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item"><a class="page-link" href="#">Next</a></li>
  </ul>
</nav>
```

Working with icons

Looking to use an icon or symbol in place of text for some pagination links? Be sure to provide proper screen reader support with aria attributes.

- «
- 1

- 2
- 3
- »

```
<nav aria-label="Page navigation example">
  <ul class="pagination">
    <li class="page-item">
      <a class="page-link" href="#" aria-label="Previous">
        <span aria-hidden="true">&laquo;</span>
      </a>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#" aria-label="Next">
        <span aria-hidden="true">&raquo;</span>
      </a>
    </li>
  </ul>
</nav>
```

Disabled and active states

Pagination links are customizable for different circumstances. Use `.disabled` for links that appear un-clickable and `.active` to indicate the current page.

While the `.disabled` class uses `pointer-events: none` to *try* to disable the link functionality of `<a>`s, that CSS property is not yet standardized and doesn't account for keyboard navigation. As such, you should always add `tabindex="-1"` on disabled links and use custom JavaScript to fully disable their functionality.

- Previous
- 1
- 2
- 3
- Next

```
<nav aria-label="...">
  <ul class="pagination">
    <li class="page-item disabled">
      <a class="page-link" href="#" tabindex="-1" aria-disabled="true">Previous</a>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item active" aria-current="page">
      <a class="page-link" href="#">2</a>
    </li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#">Next</a>
    </li>
  </ul>
</nav>
```

You can optionally swap out active or disabled anchors for `<span>`, or omit the anchor in the case of the prev/next arrows, to remove click functionality and prevent keyboard focus while retaining intended styles.

- Previous
- 1
- 2
- 3
- Next

```
<nav aria-label="...">
 <ul class="pagination">
  <li class="page-item disabled">
    <span class="page-link">Previous</span>
  </li>
  <li class="page-item"><a class="page-link" href="#">1</a></li>
  <li class="page-item active" aria-current="page">
    <span class="page-link">2</span>
  </li>
  <li class="page-item"><a class="page-link" href="#">3</a></li>
  <li class="page-item">
    <a class="page-link" href="#">Next</a>
  </li>
 </ul>
</nav>
```

Sizing

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

- 1
- 2
- 3

```
<nav aria-label="...">
 <ul class="pagination pagination-lg">
  <li class="page-item active" aria-current="page">
    <span class="page-link">1</span>
  </li>
  <li class="page-item"><a class="page-link" href="#">2</a></li>
  <li class="page-item"><a class="page-link" href="#">3</a></li>
 </ul>
</nav>
```

- 1
- 2
- 3

```
<nav aria-label="...">
  <ul class="pagination pagination-sm">
    <li class="page-item active" aria-current="page">
      <span class="page-link">1</span>
    </li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
  </ul>
</nav>
```

Alignment

Change the alignment of pagination components with flexbox utilities.

- Previous
- 1
- 2
- 3
- Next

```
<nav aria-label="Page navigation example">
  <ul class="pagination justify-content-center">
    <li class="page-item disabled">
      <a class="page-link" href="#" tabindex="-1" aria-disabled="true">Previous</a>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#">Next</a>
    </li>
  </ul>
</nav>
```

- Previous
- 1

- 2
- 3
- Next

```
<nav aria-label="Page navigation example">
  <ul class="pagination justify-content-end">
    <li class="page-item disabled">
      <a class="page-link" href="#" tabindex="-1" aria-disabled="true">Previous</a>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#">Next</a>
    </li>
  </ul>
</nav>
```

Sass

Variables

```
$pagination-padding-y:              .375rem;
$pagination-padding-x:              .75rem;
$pagination-padding-y-sm:           .25rem;
$pagination-padding-x-sm:           .5rem;
$pagination-padding-y-lg:           .75rem;
$pagination-padding-x-lg:           1.5rem;

$pagination-color:                  $link-color;
$pagination-bg:                     $white;
$pagination-border-width:           $border-width;
$pagination-border-radius:          $border-radius;
$pagination-margin-start:           -$pagination-border-width;
$pagination-border-color:           $gray-300;

$pagination-focus-color:            $link-hover-color;
$pagination-focus-bg:               $gray-200;
$pagination-focus-box-shadow:       $input-btn-focus-box-shadow;
$pagination-focus-outline:          0;

$pagination-hover-color:            $link-hover-color;
$pagination-hover-bg:               $gray-200;
$pagination-hover-border-color:     $gray-300;

$pagination-active-color:           $component-active-color;
$pagination-active-bg:              $component-active-bg;
$pagination-active-border-color:    $pagination-active-bg;
```

```
$pagination-disabled-color:         $gray-600;
$pagination-disabled-bg:            $white;
$pagination-disabled-border-color:  $gray-300;

$pagination-transition:             color .15s ease-in-out, background-color .15s ease-in-out, border-color .15s ease-in-out, box-shadow .15s ease-in-out;

$pagination-border-radius-sm:       $border-radius-sm;
$pagination-border-radius-lg:       $border-radius-lg;
```

Mixins

```
@mixin pagination-size($padding-y, $padding-x, $font-size, $border-radius) {
  .page-link {
    padding: $padding-y $padding-x;
    @include font-size($font-size);
  }

  .page-item {
    @if $pagination-margin-start == (-$pagination-border-width) {
      &:first-child {
        .page-link {
          @include border-start-radius($border-radius);
        }
      }

      &:last-child {
        .page-link {
          @include border-end-radius($border-radius);
        }
      }
    } @else {
      //Add border-radius to all pageLinks in case they have left margin
      .page-link {
        @include border-radius($border-radius);
      }
    }
  }
}
```

Popovers

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

Overview

Things to know when using the popover plugin:

- Popovers rely on the 3rd party library Popper for positioning. You must include popper.min.js before bootstrap.js or use `bootstrap.bundle.min.js` / `bootstrap.bundle.js` which contains Popper in order for popovers to work!
- Popovers require the tooltip plugin as a dependency.
- Popovers are opt-in for performance reasons, so **you must initialize them yourself**.
- Zero-length `title` and `content` values will never show a popover.
- Specify `container: 'body'` to avoid rendering problems in more complex components (like our input groups, button groups, etc).
- Triggering popovers on hidden elements will not work.
- Popovers for `.disabled` or `disabled` elements must be triggered on a wrapper element.
- When triggered from anchors that wrap across multiple lines, popovers will be centered between the anchors' overall width. Use `.text-nowrap` on your `<a>`s to avoid this behavior.
- Popovers must be hidden before their corresponding elements have been removed from the DOM.
- Popovers can be triggered thanks to an element inside a shadow DOM.

By default, this component uses the built-in content sanitizer, which strips out any HTML elements that are not explicitly allowed. See the sanitizer section in our JavaScript documentation for more details.
The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the reduced motion section of our accessibility documentation.

Keep reading to see how popovers work with some examples.

Example: Enable popovers everywhere

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

```
var popoverTriggerList = [].slice.call(document.querySelectorAll('[data-bs-toggle="popover"]'))
var popoverList = popoverTriggerList.map(function (popoverTriggerEl) {
  return new bootstrap.Popover(popoverTriggerEl)
})
```

Example: Using the container option

When you have some styles on a parent element that interfere with a popover, you'll want to specify a custom container so that the popover's HTML appears within that element instead.

```
var popover = new bootstrap.Popover(document.querySelector('.example-popover'), {
  container: 'body'
})
```

Example

Click to toggle popover

```
<button type="button" class="btn btn-lg btn-danger" data-bs-toggle="popover" title="Popover title" data-bs-content="And here's some amazing content. It's very engaging. Right?">Click to toggle popover</button>
```

Four directions

Four options are available: top, right, bottom, and left aligned. Directions are mirrored when using Bootstrap in RTL.

Popover on top Popover on right Popover on bottom Popover on left

```
<button type="button" class="btn btn-secondary" data-bs-container="body" data-bs-toggle="popover" data-bs-placement="top" data-bs-content="Top popover">
  Popover on top
</button>
<button type="button" class="btn btn-secondary" data-bs-container="body" data-bs-toggle="popover" data-bs-placement="right" data-bs-content="Right popover">
  Popover on right
</button>
<button type="button" class="btn btn-secondary" data-bs-container="body" data-bs-toggle="popover" data-bs-placement="bottom" data-bs-content="Bottom popover">
  Popover on bottom
</button>
<button type="button" class="btn btn-secondary" data-bs-container="body" data-bs-toggle="popover" data-bs-placement="left" data-bs-content="Left popover">
  Popover on left
```

```
</button>
```

Dismiss on next click

Use the `focus` trigger to dismiss popovers on the user's next click of a different element than the toggle element.

Specific markup required for dismiss-on-next-click
For proper cross-browser and cross-platform behavior, you must use the `<a>` tag, *not* the `<button>` tag, and you also must include a `tabindex` attribute.
Dismissible popover

```
<a tabindex="0" class="btn btn-lg btn-danger" role="button" data-bs-toggle="popover" data-bs-trigger="focus"
title="Dismissible popover" data-bs-content="And here's some amazing content. It's very engaging.
Right?">Dismissible popover</a>
```

```
var popover = new bootstrap.Popover(document.querySelector('.popover-dismiss'), {
  trigger: 'focus'
})
```

Disabled elements

Elements with the `disabled` attribute aren't interactive, meaning users cannot hover or click them to trigger a popover (or tooltip). As a workaround, you'll want to trigger the popover from a wrapper `<div>` or `<span>`, ideally made keyboard-focusable using `tabindex="0"`.

For disabled popover triggers, you may also prefer `data-bs-trigger="hover focus"` so that the popover appears as immediate visual feedback to your users as they may not expect to *click* on a disabled element.

Disabled button

```
<span class="d-inline-block" tabindex="0" data-bs-toggle="popover" data-bs-trigger="hover focus" data-bs-
content="Disabled popover">
  <button class="btn btn-primary" type="button" disabled>Disabled button</button>
</span>
```

Sass

Variables

```
$popover-font-size:                $font-size-sm;
$popover-bg:                     $white;
$popover-max-width:                276px;
$popover-border-width:              $border-width;
$popover-border-color:              rgba($black, .2);
$popover-border-radius:             $border-radius-lg;
$popover-inner-border-radius:       subtract($popover-border-radius, $popover-border-width);
$popover-box-shadow:                $box-shadow;

$popover-header-bg:                shade-color($popover-bg, 6%);
$popover-header-color:              $headings-color;
$popover-header-padding-y:          .5rem;
$popover-header-padding-x:          $spacer;

$popover-body-color:               $body-color;
$popover-body-padding-y:            $spacer;
$popover-body-padding-x:            $spacer;

$popover-arrow-width:               1rem;
$popover-arrow-height:              .5rem;
$popover-arrow-color:               $popover-bg;

$popover-arrow-outer-color:         fade-in($popover-border-color, .05);
```

Usage

Enable popovers via JavaScript:

```
var exampleEl = document.getElementById('example')
var popover = new bootstrap.Popover(exampleEl, options)
```

Making popovers work for keyboard and assistive technology users

To allow keyboard users to activate your popovers, you should only add them to HTML elements that are traditionally keyboard-focusable and interactive (such as links or form controls). Although arbitrary HTML elements (such as `<span>`s) can be made focusable by adding the `tabindex="0"` attribute, this will add potentially annoying and confusing tab stops on non-interactive elements for keyboard users, and most assistive technologies currently do not announce the popover's content in this situation. Additionally, do not rely solely on `hover` as the trigger for your popovers, as this will make them impossible to trigger for keyboard users.

While you can insert rich, structured HTML in popovers with the `html` option, we strongly recommend that you avoid adding an excessive amount of content. The way popovers currently work is that, once displayed, their content is tied to the trigger element with

Name	Type	Default	Description
animation	boolean	true	Apply a CSS fade transition to the popover
container	string | element | false	false	Appends the popover to a specific element. Example: container: 'body'. This option is particularly useful in that it allows you to position the popover in the flow of the document near the triggering element - which will prevent the popover from floating away from the triggering element during a window resize.
content	string | element | function	''	Default content value if data-bs-content attribute isn't present. If a function is given, it will be called with its this reference set to the element that the popover is attached to.
delay	number | object	0	Delay showing and hiding the popover (ms) - does not apply to manual trigger type If a number is supplied, delay is applied to both hide/show Object structure is: delay: { "show": 500, "hide": 100 }
html	boolean	false	Insert HTML into the popover. If false, innerText property will be used to insert content into the DOM. Use text if you're worried about XSS attacks.
placement	string | function	'right'	How to position the popover - auto | top | bottom | left | right. When auto is specified, it will dynamically reorient the popover. When a function is used to determine the placement, it is called with the popover DOM node as its first argument and the triggering element DOM node as its second. The this context is set to the popover instance.
selector	string | false	false	If a selector is provided, popover objects will be delegated to the specified targets. In practice, this is used to enable dynamic HTML content to have popovers added. See this and an informative example.

template	string	'<div class="popover" role="tooltip"><div class="popover-arrow"></div><h3 class="popover-header"></h3><div class="popover-body"></div></div>'	Base HTML to use when creating the popover. The popover's title will be injected into the .popover-header. The popover's content will be injected into the .popover-body. .popover-arrow will become the popover's arrow. The outermost wrapper element should have the .popover class.
title	string | element | function	''	Default title value if title attribute isn't present. If a function is given, it will be called with its this reference set to the element that the popover is attached to.
trigger	string	'click'	How popover is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space. manual cannot be combined with any other trigger.
fallbackPlacements	array	['top', 'right', 'bottom', 'left']	Define fallback placements by providing a list of placements in array (in order of preference). For more information refer to Popper's behavior docs
boundary	string | element	'clippingParents'	Overflow constraint boundary of the popover (applies only to Popper's preventOverflow modifier). By default it's 'clippingParents' and can accept an HTMLElement reference (via JavaScript only). For more information refer to Popper's detectOverflow docs.
customClass	string | function	''	Add classes to the popover when it is shown. Note that these classes will be added in addition to any classes specified in the template. To add multiple classes, separate them with spaces: 'class-1 class-2'. You can also pass a function that should return a single string containing additional class names.
sanitize	boolean	true	Enable or disable the sanitization. If activated 'template', 'content' and 'title' options will

the aria-describedby attribute. As a result, the entirety of the popover's content will be announced to assistive technology users as one long, uninterrupted stream.

			be sanitized. See the sanitizer section in our JavaScript documentation.
allowList	object	Default value	Object which contains allowed attributes and tags
sanitizeFn	null | function	null	Here you can supply your own sanitize function. This can be useful if you prefer to use a dedicated library to perform sanitization.
offset	array | string | function	[0, 8]	Offset of the popover relative to its target. You can pass a string in data attributes with comma separated values like: data-bs-offset="10,20" When a function is used to determine the offset, it is called with an object containing the popper placement, the reference, and popper rects as its first argument. The triggering element DOM node is passed as the second argument. The function must return an array with two numbers: [skidding, distance]. For more information refer to Popper's offset docs.
popperConfig	null | object | function	null	To change Bootstrap's default Popper config, see Popper's configuration. When a function is used to create the Popper configuration, it's called with an object that contains the Bootstrap's default Popper configuration. It helps you use and merge the default with your own configuration. The function must return a configuration object for Popper.

Additionally, while it is possible to also include interactive controls (such as form elements or links) in your popover (by adding these elements to the allowList of allowed attributes and tags), be aware that currently the popover does not manage keyboard focus order. When a keyboard user opens a popover, focus remains on the triggering element, and as the popover usually does not immediately follow the trigger in the document's structure, there is no guarantee that moving forward/pressing TAB will move a keyboard user into the popover itself. In short, simply adding interactive controls to a popover is likely to make these controls unreachable/unusable for keyboard users and users of assistive technologies, or at the very least make for an illogical overall focus order. In these cases, consider using a modal dialog instead.

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-bs-`, as in `data-bs-animation=""`. Make sure to change the case type of the option name from camelCase to kebab-case when passing the options via data attributes. For example, instead of using `data-bs-customClass="beautifier"`, use `data-bs-custom-class="beautifier"`.

Note that for security reasons the `sanitize`, `sanitizeFn`, and `allowList` options cannot be supplied using data attributes.

Data attributes for individual popovers

Options for individual popovers can alternatively be specified through the use of data attributes, as explained above.

Using function with `popperConfig`

```
var popover = new bootstrap.Popover(element, {
  popperConfig: function (defaultBsPopperConfig) {
    // var newPopperConfig = {...}
    // use defaultBsPopperConfig if needed...
    // return newPopperConfig
  }
})
```

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.

show

Reveals an element's popover. **Returns to the caller before the popover has actually been shown** (i.e. before the `shown.bs.popover` event occurs). This is considered a "manual" triggering of the popover. Popovers whose title and content are both zero-length are never displayed.

```
myPopover.show()
```

hide

Hides an element's popover. **Returns to the caller before the popover has actually been hidden** (i.e. before the `hidden.bs.popover` event occurs). This is considered a "manual" triggering of the popover.

myPopover.hide()

toggle

Toggles an element's popover. **Returns to the caller before the popover has actually been shown or hidden** (i.e. before the `shown.bs.popover` or `hidden.bs.popover` event occurs). This is considered a "manual" triggering of the popover.

myPopover.toggle()

dispose

Hides and destroys an element's popover (Removes stored data on the DOM element). Popovers that use delegation (which are created using the `selector` option) cannot be individually destroyed on descendant trigger elements.

myPopover.dispose()

enable

Gives an element's popover the ability to be shown. **Popovers are enabled by default.**

myPopover.enable()

disable

Removes the ability for an element's popover to be shown. The popover will only be able to be shown if it is re-enabled.

myPopover.disable()

toggleEnabled

Toggles the ability for an element's popover to be shown or hidden.

myPopover.toggleEnabled()

update

Updates the position of an element's popover.

myPopover.update()

getInstance

Static method which allows you to get the popover instance associated with a DOM element

```
var exampleTriggerEl = document.getElementById('example')
var popover = bootstrap.Popover.getInstance(exampleTriggerEl) // Returns a Bootstrap popover instance
```

getOrCreateInstance

Static method which allows you to get the popover instance associated with a DOM element, or create a new one in case it wasn't initialised

```
var exampleTriggerEl = document.getElementById('example')
var popover = bootstrap.Popover.getOrCreateInstance(exampleTriggerEl) // Returns a Bootstrap popover instance
```

Events

Event type	Description
show.bs.popover	This event fires immediately when the show instance method is called.
shown.bs.popover	This event is fired when the popover has been made visible to the user (will wait for CSS transitions to complete).
hide.bs.popover	This event is fired immediately when the hide instance method has been called.
hidden.bs.popover	This event is fired when the popover has finished being hidden from the user (will wait for CSS transitions to complete).
inserted.bs.popover	This event is fired after the show.bs.popover event when the popover template has been added to the DOM.

```
var myPopoverTrigger = document.getElementById('myPopover')
myPopoverTrigger.addEventListener('hidden.bs.popover', function () {
  // do something...
})
```

Progress

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

How it works

Progress components are built with two HTML elements, some CSS to set the width, and a few attributes. We don't use the HTML5 <progress> element, ensuring you can stack progress bars, animate them, and place text labels over them.

- We use the .progress as a wrapper to indicate the max value of the progress bar.
- We use the inner .progress-bar to indicate the progress so far.
- The .progress-bar requires an inline style, utility class, or custom CSS to set their width.
- The .progress-bar also requires some role and aria attributes to make it accessible.

Put that all together, and you have the following examples.

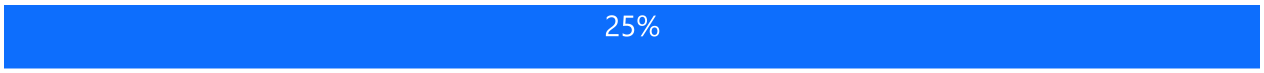
```
<div class="progress">
  <div class="progress-bar" role="progressbar" aria-valuenow="0" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 25%" aria-valuenow="25" aria-valuemin="0" aria-
valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 50%" aria-valuenow="50" aria-valuemin="0" aria-
valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 75%" aria-valuenow="75" aria-valuemin="0" aria-
valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 100%" aria-valuenow="100" aria-valuemin="0" aria-
valuemax="100"></div>
</div>
```

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

Height

We only set a `height` value on the `.progress`, so if you change that value the inner `.progress-bar` will automatically resize accordingly.

```
<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.

```
<div class="progress">
  <div class="progress-bar bg-success" role="progressbar" style="width: 25%" aria-valuenow="25" aria-
valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar bg-info" role="progressbar" style="width: 50%" aria-valuenow="50" aria-valuemin="0"
aria-valuemax="100"></div>
</div>
<div class="progress">
```

```
  <div class="progress-bar bg-warning" role="progressbar" style="width: 75%" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar bg-danger" role="progressbar" style="width: 100%" aria-valuenow="100" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Multiple bars

Include multiple progress bars in a progress component if you need.

```
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 15%" aria-valuenow="15" aria-valuemin="0" aria-valuemax="100"></div>
  <div class="progress-bar bg-success" role="progressbar" style="width: 30%" aria-valuenow="30" aria-valuemin="0" aria-valuemax="100"></div>
  <div class="progress-bar bg-info" role="progressbar" style="width: 20%" aria-valuenow="20" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Striped

Add `.progress-bar-striped` to any `.progress-bar` to apply a stripe via CSS gradient over the progress bar's background color.

```
<div class="progress">
  <div class="progress-bar progress-bar-striped" role="progressbar" style="width: 10%" aria-valuenow="10" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-striped bg-success" role="progressbar" style="width: 25%" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-striped bg-info" role="progressbar" style="width: 50%" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-striped bg-warning" role="progressbar" style="width: 75%" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-striped bg-danger" role="progressbar" style="width: 100%" aria-valuenow="100" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Animated stripes

The striped gradient can also be animated. Add .progress-bar-animated to .progress-bar to animate the stripes right to left via CSS3 animations.

Toggle animation

```
<div class="progress">
  <div class="progress-bar progress-bar-striped progress-bar-animated" role="progressbar" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100" style="width: 75%"></div>
</div>
```

Sass

Variables

```
$progress-height:                   1rem;
$progress-font-size:                $font-size-base * .75;
$progress-bg:                       $gray-200;
$progress-border-radius:            $border-radius;
$progress-box-shadow:               $box-shadow-inset;
$progress-bar-color:                $white;
$progress-bar-bg:                   $primary;
$progress-bar-animation-timing:     1s linear infinite;
$progress-bar-transition:           width .6s ease;
```

Keyframes

Used for creating the CSS animations for .progress-bar-animated. Included in scss/_progress-bar.scss.

```
@if $enable-transitions {
  @keyframes progress-bar-stripes {
    0% { background-position-x: $progress-height; }
  }
}
```

Scrollspy

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

How it works

Scrollspy has a few requirements to function properly:

- 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>`.
- 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.

Scrollable containers and keyboard access

If you're making a scrollable container (other than the `<body>`), be sure to have a `height` set and `overflow-y: scroll;` applied to it—alongside a `tabindex="0"` to ensure keyboard access.

Example in navbar

Scroll the area below the navbar and watch the active class change. The dropdown items will be highlighted as well.

Navbar

- First
- Second
- Dropdown

First heading

This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Second heading
This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Third heading
This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Fourth heading
This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Fifth heading
This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

```
<nav id="navbar-example2" class="navbar navbar-light bg-light px-3">
  <a class="navbar-brand" href="#">Navbar</a>
  <ul class="nav nav-pills">
    <li class="nav-item">
      <a class="nav-link" href="#scrollspyHeading1">First</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#scrollspyHeading2">Second</a>
    </li>
    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown" href="#" role="button" aria-
expanded="false">Dropdown</a>
      <ul class="dropdown-menu">
        <li><a class="dropdown-item" href="#scrollspyHeading3">Third</a></li>
        <li><a class="dropdown-item" href="#scrollspyHeading4">Fourth</a></li>
        <li><hr class="dropdown-divider"></li>
        <li><a class="dropdown-item" href="#scrollspyHeading5">Fifth</a></li>
      </ul>
    </li>
  </ul>
</nav>
<div data-bs-spy="scroll" data-bs-target="#navbar-example2" data-bs-offset="0" class="scrollspy-example"
tabindex="0">
  <h4 id="scrollspyHeading1">First heading</h4>
  <p>...</p>
  <h4 id="scrollspyHeading2">Second heading</h4>
  <p>...</p>
  <h4 id="scrollspyHeading3">Third heading</h4>
  <p>...</p>
  <h4 id="scrollspyHeading4">Fourth heading</h4>
  <p>...</p>
  <h4 id="scrollspyHeading5">Fifth heading</h4>
  <p>...</p>
</div>
```

Example with nested nav

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.

NavbarItem 1Item 1-1Item 1-2Item 2Item 3Item 3-1Item 3-2

Item 1

This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Item 1-1

This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Item 1-2

This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Item 2

This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Item 3

This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Item 3-1

This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Item 3-2

This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

```
<nav id="navbar-example3" class="navbar navbar-light bg-light flex-column align-items-stretch p-3">
  <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 ms-3 my-1" href="#item-1-1">Item 1-1</a>
      <a class="nav-link ms-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 ms-3 my-1" href="#item-3-1">Item 3-1</a>
      <a class="nav-link ms-3 my-1" href="#item-3-2">Item 3-2</a>
    </nav>
  </nav>
</nav>

<div data-bs-spy="scroll" data-bs-target="#navbar-example3" data-bs-offset="0" tabindex="0">
  <h4 id="item-1">Item 1</h4>
  <p>...</p>
  <h5 id="item-1-1">Item 1-1</h5>
  <p>...</p>
  <h5 id="item-1-2">Item 1-2</h5>
  <p>...</p>
  <h4 id="item-2">Item 2</h4>
  <p>...</p>
  <h4 id="item-3">Item 3</h4>
  <p>...</p>
  <h5 id="item-3-1">Item 3-1</h5>
  <p>...</p>
  <h5 id="item-3-2">Item 3-2</h5>
  <p>...</p>
</div>
```

Example with list-group

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

Item 1Item 2Item 3Item 4

Item 1
This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Item 2
This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Item 3
This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

Item 4
This is some placeholder content for the scrollspy page. Note that as you scroll down the page, the appropriate navigation link is highlighted. It's repeated throughout the component example. We keep adding some more example here to emphasize the scrolling and highlighting.

```
<div id="list-example" class="list-group">
  <a class="list-group-item list-group-item-action" href="#list-item-1">Item 1</a>
  <a class="list-group-item list-group-item-action" href="#list-item-2">Item 2</a>
  <a class="list-group-item list-group-item-action" href="#list-item-3">Item 3</a>
  <a class="list-group-item list-group-item-action" href="#list-item-4">Item 4</a>
</div>
<div data-bs-spy="scroll" data-bs-target="#list-example" data-bs-offset="0" class="scrollspy-example"
tabindex="0">
  <h4 id="list-item-1">Item 1</h4>
  <p>...</p>
  <h4 id="list-item-2">Item 2</h4>
  <p>...</p>
  <h4 id="list-item-3">Item 3</h4>
  <p>...</p>
  <h4 id="list-item-4">Item 4</h4>
  <p>...</p>
</div>
```

Usage

Via data attributes

To easily add scrollspy behavior to your topbar navigation, add `data-bs-spy="scroll"` to the element you want to spy on (most typically this would be the `<body>`). Then add the `data-bs-target` attribute with the ID or class of the parent element of any Bootstrap `.nav` component.

```
body {
  position: relative;
}
```

```
<body data-bs-spy="scroll" data-bs-target="#navbar-example">
  ...
  <div id="navbar-example">
    <ul class="nav nav-tabs" role="tablist">
      ...
    </ul>
  </div>
  ...
</body>
```

Via JavaScript

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

```
var scrollSpy = new bootstrap.ScrollSpy(document.body, {
  target: '#navbar-example'
})
```

Resolvable ID targets required

Navbar links must have resolvable id targets. For example, a `<a href="#home">home</a>` must correspond to something in the DOM like `<div id="home"></div>`.

Non-visible target elements ignored

Target elements that are not visible will be ignored and their corresponding nav items will never be highlighted.

Methods

refresh

When using scrollspy in conjunction with adding or removing of elements from the DOM, you'll need to call the refresh method like so:

```
var dataSpyList = [].slice.call(document.querySelectorAll('[data-bs-spy="scroll"]'))
dataSpyList.forEach(function (dataSpyEl) {
  bootstrap.ScrollSpy.getInstance(dataSpyEl)
    .refresh()
})
```

dispose

Destroys an element's scrollspy. (Removes stored data on the DOM element)

getInstance

Static method which allows you to get the scrollspy instance associated with a DOM element

```
var scrollSpyContentEl = document.getElementById('content')
var scrollSpy = bootstrap.ScrollSpy.getInstance(scrollSpyContentEl) // Returns a Bootstrap scrollspy instance
```

getOrCreateInstance

Static method which allows you to get the scrollspy instance associated with a DOM element, or create a new one in case it wasn't initialised

```
var scrollSpyContentEl = document.getElementById('content')
var scrollSpy = bootstrap.ScrollSpy.getOrCreateInstance(scrollSpyContentEl) // Returns a Bootstrap scrollspy
instance
```

Options

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

Name	Type	Default	Description
offset	number	10	Pixels to offset from top when calculating position of scroll.
method	string	auto	Finds which section the spied element is in. auto will choose the best method to get scroll coordinates. offset will use the [Element.getBoundingClientRect()]() method to get scroll coordinates. position will use the [HTMLElement.offsetTop]() and [HTMLElement.offsetLeft]() properties to get scroll coordinates.
target	string | jQuery object | DOM element		Specifies element to apply Scrollspy plugin.

Events

Event type	Description
activate.bs.scrollspy	This event fires on the scroll element whenever a new item becomes activated by the scrollspy.

```
var firstScrollSpyEl = document.querySelector('[data-bs-spy="scroll"]')
firstScrollSpyEl.addEventListener('activate.bs.scrollspy', function () {
  // do something...
})
```

Spinners

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

About

Bootstrap “spinners” can be used to show the loading state in your projects. They’re built only with HTML and CSS, meaning you don’t need any JavaScript to create them. You will, however, need some custom JavaScript to toggle their visibility. Their appearance, alignment, and sizing can be easily customized with our amazing utility classes.

For accessibility purposes, each loader here includes `role="status"` and a nested `<span class="visually-hidden">Loading...</span>`.

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

Border spinner

Use the border spinners for a lightweight loading indicator.

Loading...

```
<div class="spinner-border" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
```

Colors

The border spinner uses `currentColor` for its `border-color`, meaning you can customize the color with text color utilities. You can use any of our text color utilities on the standard spinner.

Loading...

Loading...

Loading...

Loading...

Loading...

Loading...

Loading...

Loading...

```
<div class="spinner-border text-primary" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-secondary" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-success" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-danger" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-warning" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-info" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-light" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-dark" role="status">
  <span class="visually-hidden">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!

Loading...

```
<div class="spinner-grow" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
```
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.

Loading...

Loading...

Loading...

Loading...

Loading...

Loading...

Loading...

Loading...

```
<div class="spinner-grow text-primary" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-grow text-secondary" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-grow text-success" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-grow text-danger" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-grow text-warning" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-grow text-info" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-grow text-light" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-grow text-dark" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
```

Alignment

Spinners in Bootstrap are built with `rem`s, `currentColor`, and `display: inline-flex`. This means they can easily be resized, recolored, and quickly aligned.

Margin

Use margin utilities like `.m-5` for easy spacing.

Loading...

```
<div class="spinner-border m-5" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
```

Placement

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

Flex

Loading...

```
<div class="d-flex justify-content-center">
  <div class="spinner-border" role="status">
    <span class="visually-hidden">Loading...</span>
  </div>
</div>
```

Loading...

```
<div class="d-flex align-items-center">
  <strong>Loading...</strong>
  <div class="spinner-border ms-auto" role="status" aria-hidden="true"></div>
</div>
```

Floats

Loading...

```
<div class="clearfix">
  <div class="spinner-border float-end" role="status">
    <span class="visually-hidden">Loading...</span>
  </div>
</div>
```

Text align

Loading...

```
<div class="text-center">
  <div class="spinner-border" role="status">
    <span class="visually-hidden">Loading...</span>
  </div>
</div>
```

Size

Add .spinner-border-sm and .spinner-grow-sm to make a smaller spinner that can quickly be used within other components.

Loading...

Loading...

```
<div class="spinner-border spinner-border-sm" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
```

```
<div class="spinner-grow spinner-grow-sm" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
```

Or, use custom CSS or inline styles to change the dimensions as needed.

Loading...

Loading...

```
<div class="spinner-border" style="width: 3rem; height: 3rem;" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-grow" style="width: 3rem; height: 3rem;" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
```

Buttons

Use spinners within buttons to indicate an action is currently processing or taking place. You may also swap the text out of the spinner element and utilize button text as needed.

Loading... Loading...

```
<button class="btn btn-primary" type="button" disabled>
  <span class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span>
  <span class="visually-hidden">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>
```

Loading... Loading...

```
<button class="btn btn-primary" type="button" disabled>
  <span class="spinner-grow spinner-grow-sm" role="status" aria-hidden="true"></span>
  <span class="visually-hidden">Loading...</span>
</button>
<button class="btn btn-primary" type="button" disabled>
  <span class="spinner-grow spinner-grow-sm" role="status" aria-hidden="true"></span>
  Loading...
</button>
```

Sass

Variables

```
$spinner-width:           2rem;
$spinner-height:          $spinner-width;
$spinner-vertical-align:  -.125em;
$spinner-border-width:    .25em;
$spinner-animation-speed: .75s;

$spinner-width-sm:        1rem;
$spinner-height-sm:       $spinner-width-sm;
$spinner-border-width-sm: .2em;
```

Keyframes

Used for creating the CSS animations for our spinners. Included in scss/_spinners.scss.

```
@keyframes spinner-border {
  to { transform: rotate(360deg) #{"/* rtl:ignore */"}; }
}


@keyframes spinner-grow {
  0% {
    transform: scale(0);
  }
  50% {
    opacity: 1;
    transform: none;
  }
}
```

Toasts

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.

Overview

Things to know when using the toast plugin:

- Toasts are opt-in for performance reasons, so **you must initialize them yourself**.
- Toasts will automatically hide if you do not specify `autohide: false`.

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

Examples

Basic

To encourage extensible and predictable toasts, we recommend a header and body. Toast headers use `display: flex`, allowing easy alignment of content thanks to our margin and flexbox utilities.

Toasts are as flexible as you need and have very little required markup. At a minimum, we require a single element to contain your "toasted" content and strongly encourage a dismiss button.

Bootstrap11 mins ago

Hello, world! This is a toast message.

```html
<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    <img src="..." class="rounded me-2" alt="...">
    <strong class="me-auto">Bootstrap</strong>
    <small>11 mins ago</small>
    <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
  </div>
  <div class="toast-body">
    Hello, world! This is a toast message.
  </div>
</div>
```

Live

Click the button below to show a toast (positioned with our utilities in the lower right corner) that has been hidden by default with `.hide`.

Show live toast

```html
<button type="button" class="btn btn-primary" id="liveToastBtn">Show live toast</button>

<div class="position-fixed bottom-0 end-0 p-3" style="z-index: 11">
  <div id="liveToast" class="toast hide" role="alert" aria-live="assertive" aria-atomic="true">
    <div class="toast-header">
      <img src="..." class="rounded me-2" alt="...">
      <strong class="me-auto">Bootstrap</strong>
      <small>11 mins ago</small>
      <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
    </div>
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
  </div>
</div>
```

Translucent

Toasts are slightly translucent to blend in with what's below them.

Bootstrap11 mins ago

Hello, world! This is a toast message.

```html
<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    <img src="..." class="rounded me-2" alt="...">
    <strong class="me-auto">Bootstrap</strong>
    <small class="text-muted">11 mins ago</small>
    <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
  </div>
  <div class="toast-body">
    Hello, world! This is a toast message.
  </div>
</div>
```

Stacking

You can stack toasts by wrapping them in a toast container, which will vertically add some spacing.

Bootstrapjust now

See? Just like this.

Bootstrap2 seconds ago

Heads up, toasts will stack automatically

```
<div class="toast-container">
  <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
    <div class="toast-header">
      <img src="..." class="rounded me-2" alt="...">
      <strong class="me-auto">Bootstrap</strong>
      <small class="text-muted">just now</small>
      <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
    </div>
    <div class="toast-body">
      See? Just like this.
    </div>
  </div>

  <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
    <div class="toast-header">
      <img src="..." class="rounded me-2" alt="...">
      <strong class="me-auto">Bootstrap</strong>
      <small class="text-muted">2 seconds ago</small>
      <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
    </div>
    <div class="toast-body">
      Heads up, toasts will stack automatically
    </div>
  </div>
</div>
```

Custom content

Customize your toasts by removing sub-components, tweaking them with utilities, or by adding your own markup. Here we've created a simpler toast by removing the default .toast-header, adding a custom hide icon from Bootstrap Icons, and using some flexbox utilities to adjust the layout.

Hello, world! This is a toast message.

```
<div class="toast align-items-center" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="d-flex">
    <div class="toast-body">
    Hello, world! This is a toast message.
   </div>
    <button type="button" class="btn-close me-2 m-auto" data-bs-dismiss="toast" aria-label="Close"></button>
  </div>
</div>
```

Alternatively, you can also add additional controls and components to toasts.

Hello, world! This is a toast message.

Take action Close

```
<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-body">
    Hello, world! This is a toast message.
    <div class="mt-2 pt-2 border-top">
      <button type="button" class="btn btn-primary btn-sm">Take action</button>
      <button type="button" class="btn btn-secondary btn-sm" data-bs-dismiss="toast">Close</button>
    </div>
  </div>
</div>
```

Color schemes

Building on the above example, you can create different toast color schemes with our color and background utilities. Here we've added .bg-primary and .text-white to the .toast, and then added .btn-close-white to our close button. For a crisp edge, we remove the default border with .border-0.

Hello, world! This is a toast message.

```
<div class="toast align-items-center text-white bg-primary border-0" role="alert" aria-live="assertive" aria-
atomic="true">
  <div class="d-flex">
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
    <button type="button" class="btn-close btn-close-white me-2 m-auto" data-bs-dismiss="toast" aria-
label="Close"></button>
  </div>
</div>
```

Placement

Place toasts with custom CSS as you need them. The top right is often used for notifications, as is the top middle. If you're only ever going to show one toast at a time, put the positioning styles right on the `.toast`.

Toast placement

Select a position...

Bootstrap11 mins ago

Hello, world! This is a toast message.

```
<form>
  <div class="mb-3">
    <label for="selectToastPlacement">Toast placement</label>
    <select class="form-select mt-2" id="selectToastPlacement">
      <option value="" selected>Select a position...</option>
      <option value="top-0 start-0">Top left</option>
      <option value="top-0 start-50 translate-middle-x">Top center</option>
      <option value="top-0 end-0">Top right</option>
      <option value="top-50 start-0 translate-middle-y">Middle left</option>
      <option value="top-50 start-50 translate-middle">Middle center</option>
      <option value="top-50 end-0 translate-middle-y">Middle right</option>
      <option value="bottom-0 start-0">Bottom left</option>
      <option value="bottom-0 start-50 translate-middle-x">Bottom center</option>
      <option value="bottom-0 end-0">Bottom right</option>
    </select>
  </div>
</form>
<div aria-live="polite" aria-atomic="true" class="bg-dark position-relative bd-example-toasts">
  <div class="toast-container position-absolute p-3" id="toastPlacement">
    <div class="toast">
      <div class="toast-header">
        <img src="..." class="rounded me-2" alt="...">
        <strong class="me-auto">Bootstrap</strong>
        <small>11 mins ago</small>
      </div>
      <div class="toast-body">
        Hello, world! This is a toast message.
      </div>
    </div>
  </div>
</div>
```

For systems that generate more notifications, consider using a wrapping element so they can easily stack.

Bootstrapjust now

See? Just like this.

Bootstrap2 seconds ago

Heads up, toasts will stack automatically

```
<div aria-live="polite" aria-atomic="true" class="position-relative">
  <!-- Position it: -->
  <!-- - `.toast-container` for spacing between toasts -->
  <!-- - `.position-absolute`, `top-0` & `end-0` to position the toasts in the upper right corner -->
  <!-- - `.p-3` to prevent the toasts from sticking to the edge of the container  -->
  <div class="toast-container position-absolute top-0 end-0 p-3">

    <!-- Then put toasts within -->
    <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
      <div class="toast-header">
        <img src="..." class="rounded me-2" alt="...">
        <strong class="me-auto">Bootstrap</strong>
        <small class="text-muted">just now</small>
        <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
      </div>
      <div class="toast-body">
        See? Just like this.
      </div>
    </div>

    <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
      <div class="toast-header">
        <img src="..." class="rounded me-2" alt="...">
        <strong class="me-auto">Bootstrap</strong>
        <small class="text-muted">2 seconds ago</small>
        <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
      </div>
      <div class="toast-body">
        Heads up, toasts will stack automatically
      </div>
    </div>
  </div>
</div>
```

You can also get fancy with flexbox utilities to align toasts horizontally and/or vertically.

Bootstrap11 mins ago

Hello, world! This is a toast message.

```html
<!-- Flexbox container for aligning the toasts -->
<div aria-live="polite" aria-atomic="true" class="d-flex justify-content-center align-items-center w-100">

  <!-- Then put toasts within -->
  <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
    <div class="toast-header">
      <img src="..." class="rounded me-2" alt="...">
      <strong class="me-auto">Bootstrap</strong>
      <small>11 mins ago</small>
      <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
    </div>
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
  </div>
</div>
```

Accessibility

Toasts are intended to be small interruptions to your visitors or users, so to help those with screen readers and similar assistive technologies, you should wrap your toasts in an `aria-live` region. Changes to live regions (such as injecting/updating a toast component) are automatically announced by screen readers without needing to move the user's focus or otherwise interrupt the user. Additionally, include `aria-atomic="true"` to ensure that the entire toast is always announced as a single (atomic) unit, rather than just announcing what was changed (which could lead to problems if you only update part of the toast's content, or if displaying the same toast content at a later point in time). If the information needed is important for the process, e.g. for a list of errors in a form, then use the alert component instead of toast.

Note that the live region needs to be present in the markup *before* the toast is generated or updated. If you dynamically generate both at the same time and inject them into the page, they will generally not be announced by assistive technologies.

You also need to adapt the `role` and `aria-live` level depending on the content. If it's an important message like an error, use `role="alert" aria-live="assertive"`, otherwise use `role="status" aria-live="polite"` attributes.

As the content you're displaying changes, be sure to update the `delay` timeout so that users have enough time to read the toast.

```html
<div class="toast" role="alert" aria-live="polite" aria-atomic="true" data-bs-delay="10000">
  <div role="alert" aria-live="assertive" aria-atomic="true">...</div>
</div>
```

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

Bootstrap11 mins ago

Hello, world! This is a toast message.

```
<div role="alert" aria-live="assertive" aria-atomic="true" class="toast" data-bs-autohide="false">
  <div class="toast-header">
    <img src="..." class="rounded me-2" alt="...">
    <strong class="me-auto">Bootstrap</strong>
    <small>11 mins ago</small>
    <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
  </div>
  <div class="toast-body">
    Hello, world! This is a toast message.
  </div>
</div>
```

While technically it's possible to add focusable/actionable controls (such as additional buttons or links) in your toast, you should avoid doing this for autohiding toasts. Even if you give the toast a long delay timeout, keyboard and assistive technology users may find it difficult to reach the toast in time to take action (since toasts don't receive focus when they are displayed). If you absolutely must have further controls, we recommend using a toast with autohide: false.

Sass

Variables

```
$toast-max-width:                   350px;
$toast-padding-x:                   .75rem;
$toast-padding-y:                   .5rem;
$toast-font-size:                   .875rem;
$toast-color:                       null;
$toast-background-color:            rgba($white, .85);
$toast-border-width:                1px;
$toast-border-color:                rgba(0, 0, 0, .1);
$toast-border-radius:               $border-radius;
$toast-box-shadow:                  $box-shadow;
$toast-spacing:                     $container-padding-x;

$toast-header-color:                $gray-600;
$toast-header-background-color:     rgba($white, .85);
$toast-header-border-color:         rgba(0, 0, 0, .05);
```

Usage

Initialize toasts via JavaScript:

```
var toastElList = [].slice.call(document.querySelectorAll('.toast'))
var toastList = toastElList.map(function (toastEl) {
  return new bootstrap.Toast(toastEl, option)
})
```

Options

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

Name	Type	Default	Description
`animation`	boolean	`true`	Apply a CSS fade transition to the toast
`autohide`	boolean	`true`	Auto hide the toast
`delay`	number	`5000`	Delay hiding the toast (ms)

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.

show

Reveals an element's toast. **Returns to the caller before the toast has actually been shown** (i.e. before the `shown.bs.toast` event occurs). You have to manually call this method, instead your toast won't show.

```
toast.show()
```

hide

Hides an element's toast. **Returns to the caller before the toast has actually been hidden** (i.e. before the `hidden.bs.toast` event occurs). You have to manually call this method if you made `autohide` to `false`.

```
toast.hide()
```

dispose

Hides an element's toast. Your toast will remain on the DOM but won't show anymore.

```
toast.dispose()
```

getInstance

Static method which allows you to get the scrollspy instance associated with a DOM element

```
var myToastEl = document.getElementById('myToastEl')
var myToast = bootstrap.Toast.getInstance(myToastEl) // Returns a Bootstrap toast instance
```

getOrCreateInstance

Static method which allows you to get the scrollspy instance associated with a DOM element, or create a new one in case it wasn't initialised

```
var myToastEl = document.getElementById('myToastEl')
var myToast = bootstrap.Toast.getOrCreateInstance(myToastEl) // Returns a Bootstrap toast instance
```

Events

Event type	Description
show.bs.toast	This event fires immediately when the show instance method is called.
shown.bs.toast	This event is fired when the toast has been made visible to the user.
hide.bs.toast	This event is fired immediately when the hide instance method has been called.
hidden.bs.toast	This event is fired when the toast has finished being hidden from the user.

```
var myToastEl = document.getElementById('myToast')
myToastEl.addEventListener('hidden.bs.toast', function () {
  // do something...
})
```

Tooltips

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

Overview

Things to know when using the tooltip plugin:

- Tooltips rely on the 3rd party library Popper for positioning. You must include popper.min.js before bootstrap.js or use `bootstrap.bundle.min.js` / `bootstrap.bundle.js` which contains Popper in order for tooltips to work!
- 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.

By default, this component uses the built-in content sanitizer, which strips out any HTML elements that are not explicitly allowed. See the sanitizer section in our JavaScript documentation for more details.
The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the reduced motion section of our accessibility documentation.

Got all that? Great, let's see how they work with some examples.

Example: Enable tooltips everywhere

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

```
var tooltipTriggerList = [].slice.call(document.querySelectorAll('[data-bs-toggle="tooltip"]'))
var tooltipList = tooltipTriggerList.map(function (tooltipTriggerEl) {
  return new bootstrap.Tooltip(tooltipTriggerEl)
})
```

Examples

Hover over the links below to see tooltips:

Placeholder text to demonstrate some inline links with tooltips. This is now just filler, no killer. Content placed here just to mimic the presence of real text. And all that just to give you an idea of how tooltips would look when used in real-world situations. So hopefully you've now seen how these tooltips on links can work in practice, once you use them on your own site or project.

Hover over the buttons below to see the four tooltips directions: top, right, bottom, and left. Directions are mirrored when using Bootstrap in RTL.

Tooltip on top Tooltip on right Tooltip on bottom Tooltip on left Tooltip with HTML

```
<button type="button" class="btn btn-secondary" data-bs-toggle="tooltip" data-bs-placement="top" title="Tooltip on top">
  Tooltip on top
</button>
<button type="button" class="btn btn-secondary" data-bs-toggle="tooltip" data-bs-placement="right" title="Tooltip on right">
  Tooltip on right
</button>
<button type="button" class="btn btn-secondary" data-bs-toggle="tooltip" data-bs-placement="bottom" title="Tooltip on bottom">
  Tooltip on bottom
</button>
<button type="button" class="btn btn-secondary" data-bs-toggle="tooltip" data-bs-placement="left" title="Tooltip on left">
  Tooltip on left
</button>
```

And with custom HTML added:

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

```
  Tooltip with HTML
</button>
```
With an SVG:

Sass

Variables

```
$tooltip-font-size:               $font-size-sm;
$tooltip-max-width:                 200px;
$tooltip-color:                   $white;
$tooltip-bg:                     $black;
$tooltip-border-radius:             $border-radius;
$tooltip-opacity:                 .9;
$tooltip-padding-y:                 $spacer * .25;
$tooltip-padding-x:                 $spacer * .5;
$tooltip-margin:                  0;

$tooltip-arrow-width:               .8rem;
$tooltip-arrow-height:              .4rem;
$tooltip-arrow-color:               $tooltip-bg;
```

Usage

The tooltip plugin generates content and markup on demand, and by default places tooltips after their trigger element.

Trigger the tooltip via JavaScript:

```
var exampleEl = document.getElementById('example')
var tooltip = new bootstrap.Tooltip(exampleEl, 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 this, set the `boundary` option (for the flip modifier using the `popperConfig` option) to any HTMLElement to override the default value, `'clippingParents'`, such as `document.body`:

```
var exampleEl = document.getElementById('example')
var tooltip = new bootstrap.Tooltip(exampleEl, {
  boundary: document.body // or document.querySelector('#boundary')
```

})

Markup

The required markup for a tooltip is only a data attribute and title on the HTML element you wish to have a tooltip. The generated markup of a tooltip is rather simple, though it does require a position (by default, set to top by the plugin).

Making tooltips work for keyboard and assistive technology users
You should only add tooltips to HTML elements that are traditionally keyboard-focusable and interactive (such as links or form controls). Although arbitrary HTML elements (such as <span>s) can be made focusable by adding the tabindex="0" attribute, this will add potentially annoying and confusing tab stops on non-interactive elements for keyboard users, and most assistive technologies currently do not announce the tooltip in this situation. Additionally, do not rely solely on hover as the trigger for your tooltip, as this will make your tooltips impossible to trigger for keyboard users.

```
<!-- HTML to write -->
<a href="#" data-bs-toggle="tooltip" title="Some tooltip text!">Hover over me</a>

<!-- Generated markup by the plugin -->
<div class="tooltip bs-tooltip-top" role="tooltip">
  <div class="tooltip-arrow"></div>
  <div class="tooltip-inner">
    Some tooltip text!
  </div>
</div>
```

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 <span>, ideally made keyboard-focusable using tabindex="0".

Disabled button

```
<span class="d-inline-block" tabindex="0" data-bs-toggle="tooltip" title="Disabled tooltip">
  <button class="btn btn-primary" type="button" disabled>Disabled button</button>
</span>
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to data-bs-, as in data-bs-animation="". Make sure to change the case type of the option name from camelCase to kebab-case when passing the options via data attributes. For example, instead of using data-bs-customClass="beautifier", use data-bs-custom-class="beautifier".

Note that for security reasons the sanitize, sanitizeFn, and allowList options cannot be supplied using data attributes.

Name	Type	Default	Description
animation	boolean	true	Apply a CSS fade transition to the tooltip
container	string | element | false	false	Appends the tooltip to a specific element. Example: container: 'body'. This option is particularly useful in that it allows you to position the tooltip in the flow of the document near the triggering element - which will prevent the tooltip from floating away from the triggering element during a window resize.
delay	number | object	0	Delay showing and hiding the tooltip (ms) - does not apply to manual trigger type If a number is supplied, delay is applied to both hide/show Object structure is: delay: { "show": 500, "hide": 100 }
html	boolean	false	Allow HTML in the tooltip. If true, HTML tags in the tooltip's title will be rendered in the tooltip. If false, innerText property will be used to insert content into the DOM. Use text if you're worried about XSS attacks.
placement	string | function	'top'	How to position the tooltip - auto | top | bottom | left | right. When auto is specified, it will dynamically reorient the tooltip.

Name	Type	Default	Description
			When a function is used to determine the placement, it is called with the tooltip DOM node as its first argument and the triggering element DOM node as its second. The this context is set to the tooltip instance.
selector	string | false	false	If a selector is provided, tooltip objects will be delegated to the specified targets. In practice, this is used to also apply tooltips to dynamically added DOM elements (jQuery.on support). See this and an informative example.
template	string	'<div class="tooltip" role="tooltip"><div class="tooltip-arrow"></div><div class="tooltip-inner"></div></div>'	Base HTML to use when creating the tooltip. The tooltip's title will be injected into the .tooltip-inner. .tooltip-arrow will become the tooltip's arrow. The outermost wrapper element should have the .tooltip class and role="tooltip".
title	string | element | function	''	Default title value if title attribute isn't present. If a function is given, it will be called with its this reference set to the element that the tooltip is attached to.

Name	Type	Default	Description
trigger	string	'hover focus'	How tooltip is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space. 'manual' indicates that the tooltip will be triggered programmatically via the .show(), .hide() and .toggle() methods; this value cannot be combined with any other trigger. 'hover' on its own will result in tooltips that cannot be triggered via the keyboard, and should only be used if alternative methods for conveying the same information for keyboard users is present.
fallbackPlacements	array	['top', 'right', 'bottom', 'left']	Define fallback placements by providing a list of placements in array (in order of preference). For more information refer to Popper's behavior docs
boundary	string | element	'clippingParents'	Overflow constraint boundary of the tooltip (applies only to Popper's preventOverflow modifier). By default it's 'clippingParents' and can accept an HTMLElement reference (via JavaScript only). For more information refer to Popper's detectOverflow docs.
customClass	string | function	''	Add classes to the tooltip when it is shown. Note that these classes will be added in addition to any classes specified in the template. To add multiple classes, separate them with spaces: 'class-1 class-2'. You can also pass a function that should return a single string containing additional class names.

Name	Type	Default	Description
sanitize	boolean	true	Enable or disable the sanitization. If activated 'template' and 'title' options will be sanitized. See the sanitizer section in our JavaScript documentation.
allowList	object	Default value	Object which contains allowed attributes and tags
sanitizeFn	null | function	null	Here you can supply your own sanitize function. This can be useful if you prefer to use a dedicated library to perform sanitization.
offset	array | string | function	[0, 0]	Offset of the tooltip relative to its target. You can pass a string in data attributes with comma separated values like: data-bs-offset="10,20" When a function is used to determine the offset, it is called with an object containing the popper placement, the reference, and popper rects as its first argument. The triggering element DOM node is passed as the second argument. The function must return an array with two numbers: [skidding, distance]. For more information refer to Popper's offset docs.
popperConfig	null | object | function	null	To change Bootstrap's default Popper config, see Popper's configuration. When a function is used to create the Popper configuration, it's called with an object that contains the Bootstrap's default Popper configuration. It helps you use and merge the default with your own configuration. The function must return a configuration object for Popper.

Data attributes for individual tooltips

Options for individual tooltips can alternatively be specified through the use of data attributes, as explained above.

Using function with popperConfig

```
var tooltip = new bootstrap.Tooltip(element, {
  popperConfig: function (defaultBsPopperConfig) {
    // var newPopperConfig = {...}
    // use defaultBsPopperConfig if needed...
    // return newPopperConfig
  }
})
```

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.

show

Reveals an element's tooltip. **Returns to the caller before the tooltip has actually been shown** (i.e. before the `shown.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip. Tooltips with zero-length titles are never displayed.

```
tooltip.show()
```

hide

Hides an element's tooltip. **Returns to the caller before the tooltip has actually been hidden** (i.e. before the `hidden.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip.

```
tooltip.hide()
```

toggle

Toggles an element's tooltip. **Returns to the caller before the tooltip has actually been shown or hidden** (i.e. before the `shown.bs.tooltip` or `hidden.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip.

```
tooltip.toggle()
```

dispose

Hides and destroys an element's tooltip (Removes stored data on the DOM element). Tooltips that use delegation (which are created using the selector option) cannot be individually destroyed on descendant trigger elements.

```
tooltip.dispose()
```

enable

Gives an element's tooltip the ability to be shown. **Tooltips are enabled by default.**

```
tooltip.enable()
```

disable

Removes the ability for an element's tooltip to be shown. The tooltip will only be able to be shown if it is re-enabled.

```
tooltip.disable()
```

toggleEnabled

Toggles the ability for an element's tooltip to be shown or hidden.

```
tooltip.toggleEnabled()
```

update

Updates the position of an element's tooltip.

```
tooltip.update()
```

getInstance

Static method which allows you to get the tooltip instance associated with a DOM element

```
var exampleTriggerEl = document.getElementById('example')
var tooltip = bootstrap.Tooltip.getInstance(exampleTriggerEl) // Returns a Bootstrap tooltip instance
```

getOrCreateInstance

Static method which allows you to get the tooltip instance associated with a DOM element, or create a new one in case it wasn't initialised

```
var exampleTriggerEl = document.getElementById('example')
var tooltip = bootstrap.Tooltip.getOrCreateInstance(exampleTriggerEl) // Returns a Bootstrap tooltip instance
```

Events

Event type	Description
show.bs.tooltip	This event fires immediately when the show instance method is called.
shown.bs.tooltip	This event is fired when the tooltip has been made visible to the user (will wait for CSS transitions to complete).
hide.bs.tooltip	This event is fired immediately when the hide instance method has been called.
hidden.bs.tooltip	This event is fired when the tooltip has finished being hidden from the user (will wait for CSS transitions to complete).
inserted.bs.tooltip	This event is fired after the show.bs.tooltip event when the tooltip template has been added to the DOM.

```
var myTooltipEl = document.getElementById('myTooltip')
var tooltip = new bootstrap.Tooltip(myTooltipEl)

myTooltipEl.addEventListener('hidden.bs.tooltip', function () {
  // do something...
})

tooltip.hide()
```