CSS

Global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system.

Overview

Get the lowdown on the key pieces of Bootstrap's infrastructure, including our approach to better, faster, stronger web development.

HTML5 doctype

Bootstrap makes use of certain HTML elements and CSS properties that require the use of the HTML5 doctype. Include it at the beginning of all your projects.

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

Mobile first

With Bootstrap 2, we added optional mobile friendly styles for key aspects of the framework. With Bootstrap 3, we've rewritten the project to be mobile friendly from the start. Instead of adding on optional mobile styles, they're baked right into the core. In fact, **Bootstrap is mobile first**. Mobile first styles can be found throughout the entire library instead of in separate files.

To ensure proper rendering and touch zooming, **add the viewport meta tag** to your `<head>`.

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

You can disable zooming capabilities on mobile devices by adding `user-scalable=no` to the viewport meta tag. This disables zooming, meaning users are only able to scroll, and results in your site feeling a bit more like a native application. Overall, we don't recommend this on every site, so use caution!

```
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
```

Typography and links

Bootstrap sets basic global display, typography, and link styles. Specifically, we:

- Set `background-color: #fff;` on the `body`
- Use the `@font-family-base`, `@font-size-base`, and `@line-height-base` attributes as our typographic base
- Set the global link color via `@link-color` and apply link underlines only on `:hover`

These styles can be found within `scaffolding.less`.

Normalize.css

For improved cross-browser rendering, we use Normalize.css, a project by Nicolas Gallagher and Jonathan Neal.

Containers

Bootstrap requires a containing element to wrap site contents and house our grid system. You may choose one of two containers to use in your projects. Note that, due to `padding` and more, neither container is nestable.

Use `.container` for a responsive fixed width container.

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

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

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

Grid system

Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases. It includes predefined classes for easy layout options, as well as powerful mixins for generating more semantic layouts.

Introduction

Grid systems are used for creating page layouts through a series of rows and columns that house your content. Here's how the Bootstrap grid system works:

- Rows must be placed within a `.container` (fixed-width) or `.container-fluid` (full-width) for proper alignment and padding.
- Use rows to create horizontal groups of columns.
- Content should be placed within columns, and only columns may be immediate children of rows.
- Predefined grid classes like `.row` and `.col-xs-4` are available for quickly making grid layouts. Less mixins can also be used for more semantic layouts.
- Columns create gutters (gaps between column content) via `padding`. That padding is offset in rows for the first and last column via negative margin on `.row`s.
- The negative margin is why the examples below are outdented. It's so that content within grid columns is lined up with non-grid content.
- Grid columns are created by specifying the number of twelve available columns you wish to span. For example, three equal columns would use three `.col-xs-4`.
- If more than 12 columns are placed within a single row, each group of extra columns will, as one unit, wrap onto a new line.
- Grid classes apply to devices with screen widths greater than or equal to the breakpoint sizes, and override grid classes targeted at smaller devices. Therefore, e.g. applying any `.col-md-*` class to an element will not only affect its styling on medium devices but also on large devices if a `.col-lg-*` class is not present.

Look to the examples for applying these principles to your code.

Media queries

We use the following media queries in our Less files to create the key breakpoints in our grid system.

```
/* Extra small devices (phones, less than 768px) */
/* No media query since this is the default in Bootstrap */

/* Small devices (tablets, 768px and up) */
@media (min-width: @screen-sm-min) { ... }

/* Medium devices (desktops, 992px and up) */
@media (min-width: @screen-md-min) { ... }

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

We occasionally expand on these media queries to include a `max-width` to limit CSS to a narrower set of devices.

```
@media (max-width: @screen-xs-max) { ... }
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }
@media (min-width: @screen-md-min) and (max-width: @screen-md-max) { ... }
@media (min-width: @screen-lg-min) { ... }
```

Grid options

See how aspects of the Bootstrap grid system work across multiple devices with a handy table.

	Extra small devices Phones (<768px)	**Small devices** Tablets (≥768px)	**Medium devices** Desktops (≥992px)	**Large devices** Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
Container width	None (auto)	750px	970px	1170px
Class prefix	`.col-xs-`	`.col-sm-`	`.col-md-`	`.col-lg-`
# of columns	12			
Column width	Auto	~62px	~81px	~97px
Gutter width	30px (15px on each side of a column)			
Nestable	Yes			
Offsets	Yes			
Column ordering	Yes			

Example: Stacked-to-horizontal

Using a single set of `.col-md-*` grid classes, you can create a basic grid system that starts out stacked on mobile devices and tablet devices (the extra small to small range) before becoming horizontal on desktop (medium) devices. Place grid columns in any `.row`.

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1

.col-md-8	.col-md-4

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

.col-md-6	.col-md-6

```
<div class="row">
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
</div>
<div class="row">
  <div class="col-md-8">.col-md-8</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-6">.col-md-6</div>
  <div class="col-md-6">.col-md-6</div>
</div>
```

Example: Fluid container

Turn any fixed-width grid layout into a full-width layout by changing your outermost `.container` to `.container-fluid`.

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

Example: Mobile and desktop

Don't want your columns to simply stack in smaller devices? Use the extra small and medium device grid classes by adding `.col-xs-*` `.col-md-*` to your columns. See the example below for a better idea of how it all works.

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

.col-xs-6 .col-md-4	.col-xs-6 .col-md-4	.col-xs-6 .col-md-4

.col-xs-6	.col-xs-6

```
<!-- Stack the columns on mobile by making one full-width and the other half-width -->
<div class="row">
  <div class="col-xs-12 col-md-8">.col-xs-12 .col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>

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

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

Example: Mobile, tablet, desktop

Build on the previous example by creating even more dynamic and powerful layouts with tablet `.col-sm-*` classes.

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

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

```
<div class="row">
  <div class="col-xs-12 col-sm-6 col-md-8">.col-xs-12 .col-sm-6 .col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>
<div class="row">
  <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
  <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
  <!-- Optional: clear the XS cols if their content doesn't match in height -->
  <div class="clearfix visible-xs-block"></div>
```

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

Example: Column wrapping

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

.col-xs-9

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

.col-xs-6
Subsequent columns continue along the new line.

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

Responsive column resets

With the four tiers of grids available you're bound to run into issues where, at certain breakpoints, your columns don't clear quite right as one is taller than the other. To fix that, use a combination of a `.clearfix` and our responsive utility classes.

.col-xs-6 .col-sm-3
Resize your viewport or check it out on your phone for an example.

.col-xs-6 .col-sm-3

.col-xs-6 .col-sm-3

.col-xs-6 .col-sm-3

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

  <!-- Add the extra clearfix for only the required viewport -->
  <div class="clearfix visible-xs-block"></div>

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

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

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

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

Offsetting columns

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

.col-md-4

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

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

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

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

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

You can also override offsets from lower grid tiers with `.col-*-offset-0` classes.

```
<div class="row">
  <div class="col-xs-6 col-sm-4">
  </div>
  <div class="col-xs-6 col-sm-4">
  </div>
  <div class="col-xs-6 col-xs-offset-3 col-sm-4 col-sm-offset-0">
```

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

Nesting columns

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

Level 1: .col-sm-9	
Level 2: .col-xs-8 .col-sm-6	Level 2: .col-xs-4 .col-sm-6

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

Column ordering

Easily change the order of our built-in grid columns with `.col-md-push-*` and `.col-md-pull-*` modifier classes.

.col-md-3 .col-md-pull-9	.col-md-9 .col-md-push-3

```
<div class="row">
  <div class="col-md-9 col-md-push-3">.col-md-9 .col-md-push-3</div>
  <div class="col-md-3 col-md-pull-9">.col-md-3 .col-md-pull-9</div>
</div>
```

Less mixins and variables

In addition to prebuilt grid classes for fast layouts, Bootstrap includes Less variables and mixins for quickly generating your own simple, semantic layouts.

Variables

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

```
@grid-columns:              12;
@grid-gutter-width:         30px;
@grid-float-breakpoint:     768px;
```

Mixins

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

```
// Creates a wrapper for a series of columns
.make-row(@gutter: @grid-gutter-width) {
  // Then clear the floated columns
  .clearfix();

  @media (min-width: @screen-sm-min) {
    margin-left:  (@gutter / -2);
    margin-right: (@gutter / -2);
  }

  // Negative margin nested rows out to align the content of columns
  .row {
    margin-left:  (@gutter / -2);
    margin-right: (@gutter / -2);
  }
}

// Generate the extra small columns
.make-xs-column(@columns; @gutter: @grid-gutter-width) {
  position: relative;
  // Prevent columns from collapsing when empty
  min-height: 1px;
  // Inner gutter via padding
  padding-left:  (@gutter / 2);
  padding-right: (@gutter / 2);

  // Calculate width based on number of columns available
  @media (min-width: @grid-float-breakpoint) {
    float: left;
    width: percentage((@columns / @grid-columns));
  }
}

// Generate the small columns
.make-sm-column(@columns; @gutter: @grid-gutter-width) {
  position: relative;
```

```
  // Prevent columns from collapsing when empty
  min-height: 1px;
  // Inner gutter via padding
  padding-left:  (@gutter / 2);
  padding-right: (@gutter / 2);

  // Calculate width based on number of columns available
  @media (min-width: @screen-sm-min) {
    float: left;
    width: percentage((@columns / @grid-columns));
  }
}

// Generate the small column offsets
.make-sm-column-offset(@columns) {
  @media (min-width: @screen-sm-min) {
    margin-left: percentage((@columns / @grid-columns));
  }
}
.make-sm-column-push(@columns) {
  @media (min-width: @screen-sm-min) {
    left: percentage((@columns / @grid-columns));
  }
}
.make-sm-column-pull(@columns) {
  @media (min-width: @screen-sm-min) {
    right: percentage((@columns / @grid-columns));
  }
}

// Generate the medium columns
.make-md-column(@columns; @gutter: @grid-gutter-width) {
  position: relative;
  // Prevent columns from collapsing when empty
  min-height: 1px;
  // Inner gutter via padding
  padding-left:  (@gutter / 2);
  padding-right: (@gutter / 2);

  // Calculate width based on number of columns available
  @media (min-width: @screen-md-min) {
    float: left;
    width: percentage((@columns / @grid-columns));
  }
}

// Generate the medium column offsets
.make-md-column-offset(@columns) {
  @media (min-width: @screen-md-min) {
    margin-left: percentage((@columns / @grid-columns));
  }
}
.make-md-column-push(@columns) {
  @media (min-width: @screen-md-min) {
    left: percentage((@columns / @grid-columns));
  }
}
.make-md-column-pull(@columns) {
  @media (min-width: @screen-md-min) {
    right: percentage((@columns / @grid-columns));
  }
}

// Generate the large columns
.make-lg-column(@columns; @gutter: @grid-gutter-width) {
  position: relative;
  // Prevent columns from collapsing when empty
  min-height: 1px;
  // Inner gutter via padding
  padding-left:  (@gutter / 2);
  padding-right: (@gutter / 2);

  // Calculate width based on number of columns available
  @media (min-width: @screen-lg-min) {
    float: left;
    width: percentage((@columns / @grid-columns));
  }
}

// Generate the large column offsets
.make-lg-column-offset(@columns) {
  @media (min-width: @screen-lg-min) {
    margin-left: percentage((@columns / @grid-columns));
  }
}
.make-lg-column-push(@columns) {
  @media (min-width: @screen-lg-min) {
    left: percentage((@columns / @grid-columns));
  }
}
.make-lg-column-pull(@columns) {
  @media (min-width: @screen-lg-min) {
    right: percentage((@columns / @grid-columns));
  }
}
```

Example usage

You can modify the variables to your own custom values, or just use the mixins with their default values. Here's an example of using the

default settings to create a two-column layout with a gap between.

```
.wrapper {
  .make-row();
}
.content-main {
  .make-lg-column(8);
}
.content-secondary {
  .make-lg-column(3);
  .make-lg-column-offset(1);
}
```

```
<div class="wrapper">
  <div class="content-main">...</div>
  <div class="content-secondary">...</div>
</div>
```

Typography

Headings

All HTML headings, `<h1>` through `<h6>`, are available. `.h1` through `.h6` classes are also available, for when you want to match the font styling of a heading but still want your text to be displayed inline.

EXAMPLE

h1. Bootstrap heading	Semibold 36px
h2. Bootstrap heading	Semibold 30px
h3. Bootstrap heading	Semibold 24px
h4. Bootstrap heading	Semibold 18px
h5. Bootstrap heading	Semibold 14px
h6. Bootstrap heading	Semibold 12px

```
<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>
```

Create lighter, secondary text in any heading with a generic `<small>` tag or the `.small` class.

EXAMPLE

h1. Bootstrap heading Secondary text

h2. Bootstrap heading Secondary text

h3. Bootstrap heading Secondary text

h4. Bootstrap heading Secondary text

h5. Bootstrap heading Secondary text

h6. Bootstrap heading Secondary text

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

Body copy

Bootstrap's global default `font-size` is **14px**, with a `line-height` of **1.428**. This is applied to the `<body>` and all paragraphs. In addition, `<p>` (paragraphs) receive a bottom margin of half their computed line-height (10px by default).

EXAMPLE

Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec ullamcorper nulla non metus auctor fringilla. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla non metus auctor fringilla.

Maecenas sed diam eget risus varius blandit sit amet non magna. Donec id elit non mi porta gravida at eget metus. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit.

```
<p>...</p>
```

Lead body copy

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

EXAMPLE

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

```
<p class="lead">...</p>
```

Built with Less

The typographic scale is based on two Less variables in **variables.less**: `@font-size-base` and `@line-height-base`. The first is the base font-size used throughout and the second is the base line-height. We use those variables and some simple math to create the margins, paddings, and line-heights of all our type and more. Customize them and Bootstrap adapts.

Inline text elements

Marked text

For highlighting a run of text due to its relevance in another context, use the `<mark>` tag.

EXAMPLE

You can use the mark tag to highlight text.

```
You can use the mark tag to <mark>highlight</mark> text.
```

Deleted text

For indicating blocks of text that have been deleted use the `<del>` tag.

EXAMPLE

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

```
<del>This line of text is meant to be treated as deleted text.</del>
```

Strikethrough text

For indicating blocks of text that are no longer relevant use the `<s>` tag.

EXAMPLE

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

```
<s>This line of text is meant to be treated as no longer accurate.</s>
```

Inserted text

For indicating additions to the document use the `<ins>` tag.

EXAMPLE

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

```
<ins>This line of text is meant to be treated as an addition to the document.</ins>
```

Underlined text

To underline text use the `<u>` tag.

EXAMPLE

This line of text will render as underlined

```
<u>This line of text will render as underlined</u>
```

Make use of HTML's default emphasis tags with lightweight styles.

Small text

For de-emphasizing inline or blocks of text, use the `<small>` tag to set text at 85% the size of the parent. Heading elements receive their

For de-emphasizing inline or blocks of text, use the `<small>` tag to set text at 85% the size of the parent. Headings elements receive their own `font-size` for nested `<small>` elements.

You may alternatively use an inline element with `.small` in place of any `<small>`.

EXAMPLE

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

```
<small>This line of text is meant to be treated as fine print.</small>
```

Bold

For emphasizing a snippet of text with a heavier font-weight.

EXAMPLE

The following snippet of text is **rendered as bold text**.

```
<strong>rendered as bold text</strong>
```

Italics

For emphasizing a snippet of text with italics.

EXAMPLE

The following snippet of text is *rendered as italicized text*.

```
<em>rendered as italicized text</em>
```

> **Alternate elements**
>
> 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.

Alignment classes

Easily realign text to components with text alignment classes.

EXAMPLE

Left aligned text.

Center aligned text.

Right aligned text.

Justified text.

No wrap text.

```
<p class="text-left">Left aligned text.</p>
<p class="text-center">Center aligned text.</p>
<p class="text-right">Right aligned text.</p>
<p class="text-justify">Justified text.</p>
<p class="text-nowrap">No wrap text.</p>
```

Transformation classes

Transform text in components with text capitalization classes.

EXAMPLE

lowercased text.

UPPERCASED TEXT.

Capitalized Text.

```
<p class="text-lowercase">Lowercased text.</p>
<p class="text-uppercase">Uppercased text.</p>
<p class="text-capitalize">Capitalized text.</p>
```

Abbreviations

Stylized implementation of HTML's `<abbr>` element for abbreviations and acronyms to show the expanded version on hover. Abbreviations with a `title` attribute have a light dotted bottom border and a help cursor on hover, providing additional context on hover and to users of assistive technologies.

Basic abbreviation

EXAMPLE

An abbreviation of the word attribute is attr.

```
<abbr title="attribute">attr</abbr>
```

Initialism

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

EXAMPLE

HTML is the best thing since sliced bread.

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

Addresses

Present contact information for the nearest ancestor or the entire body of work. Preserve formatting by ending all lines with `<br>`.

EXAMPLE

Twitter, Inc.
1355 Market Street, Suite 900
San Francisco, CA 94103
P: (123) 456-7890

Full Name
first.last@example.com

```
<address>
  <strong>Twitter, Inc.</strong><br>
  1355 Market Street, Suite 900<br>
  San Francisco, CA 94103<br>
  <abbr title="Phone">P:</abbr> (123) 456-7890
</address>

<address>
  <strong>Full Name</strong><br>
  <a href="mailto:#">first.last@example.com</a>
</address>
```

Blockquotes

For quoting blocks of content from another source within your document.

Default blockquote

Wrap `<blockquote>` around any HTML as the quote. For straight quotes, we recommend a `<p>`.

EXAMPLE

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

```
<blockquote>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.</p>
</blockquote>
```

Blockquote options

Style and content changes for simple variations on a standard `<blockquote>`.

Naming a source

Add a `<footer>` for identifying the source. Wrap the name of the source work in `<cite>`.

EXAMPLE

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

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

Alternate displays

Add `.blockquote-reverse` for a blockquote with right-aligned content.

EXAMPLE

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

```
<blockquote class="blockquote-reverse">
  ...
</blockquote>
```

Lists

Unordered

A list of items in which the order does *not* explicitly matter.

EXAMPLE

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

```
<ul>
  <li>...</li>
</ul>
```

Ordered

A list of items in which the order *does* explicitly matter.

EXAMPLE

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

```
<ol>
  <li>...</li>
</ol>
```

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.

EXAMPLE

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

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

```
<ul class="list-unstyled">
  <li>...</li>
</ul>
```

Inline

Place all list items on a single line with `display: inline-block;` and some light padding.

EXAMPLE

Lorem ipsum Phasellus iaculis Nulla volutpat

```
<ul class="list-inline">
  <li>...</li>
</ul>
```

Description

A list of terms with their associated descriptions.

EXAMPLE

Description lists
A description list is perfect for defining terms.
Euismod
Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.
Donec id elit non mi porta gravida at eget metus.
Malesuada porta
Etiam porta sem malesuada magna mollis euismod.

```
<dl>
  <dt>...</dt>
  <dd>...</dd>
</dl>
```

Horizontal description

Make terms and descriptions in `<dl>` line up side-by-side. Starts off stacked like default `<dl>`s, but when the navbar expands, so do these.

EXAMPLE

Description lists	A description list is perfect for defining terms.
Euismod	Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit. Donec id elit non mi porta gravida at eget metus.
Malesuada porta	Etiam porta sem malesuada magna mollis euismod.
Felis euismod sempe...	Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.

```
<dl class="dl-horizontal">
  <dt>...</dt>
  <dd>...</dd>
</dl>
```

> **Auto-truncating**
>
> Horizontal description lists will truncate terms that are too long to fit in the left column with `text-overflow`. In narrower viewports, they will change to the default stacked layout.

Code

Inline

Wrap inline snippets of code with `<code>`.

EXAMPLE

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

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

User input

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

EXAMPLE

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

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

Basic block

Use `<pre>` for multiple lines of code. Be sure to escape any angle brackets in the code for proper rendering.

EXAMPLE

```
<p>Sample text here...</p>
```

```
<pre>&lt;p&gt;Sample text here...&lt;/p&gt;</pre>
```

You may optionally add the `.pre-scrollable` class, which will set a max-height of 350px and provide a y-axis scrollbar.

Variables

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

EXAMPLE

y = *mx* + *b*

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

Sample output

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

EXAMPLE

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

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

Tables

Basic example

For basic styling—light padding and only horizontal dividers—add the base class `.table` to any `<table>`. It may seem super redundant, but given the widespread use of tables for other plugins like calendars and date pickers, we've opted to isolate our custom table styles.

EXAMPLE

Optional table caption.

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

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

Striped rows

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

> **Cross-browser compatibility**
> Striped tables are styled via the `:nth-child` CSS selector, which is not available in Internet Explorer 8.

EXAMPLE

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

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

Bordered table

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

EXAMPLE

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

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

Hover rows

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

EXAMPLE

#	First Name	Last Name	Username
1	Mark	Otto	@mdo

2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

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

Condensed table

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

EXAMPLE

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

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

Contextual classes

Use contextual classes to color table rows or individual cells.

Class	Description
`.active`	Applies the hover color to a particular row or cell
`.success`	Indicates a successful or positive action
`.info`	Indicates a neutral informative change or action
`.warning`	Indicates a warning that might need attention
`.danger`	Indicates a dangerous or potentially negative action

EXAMPLE

#	Column heading	Column heading	Column heading
1	Column content	Column content	Column content
2	Column content	Column content	Column content
3	Column content	Column content	Column content
4	Column content	Column content	Column content
5	Column content	Column content	Column content
6	Column content	Column content	Column content
7	Column content	Column content	Column content
8	Column content	Column content	Column content
9	Column content	Column content	Column content

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

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

Conveying meaning to assistive technologies

Using color to add meaning to a table row or individual cell 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 (the visible text in the relevant table row/cell), or is included through alternative means, such as additional text hidden with the `.sr-only` class.

Responsive tables

Create responsive tables by wrapping any `.table` in `.table-responsive` to make them scroll horizontally on small devices (under 768px). When viewing on anything larger than 768px wide, you will not see any difference in these tables.

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.

Firefox and fieldsets

Firefox has some awkward fieldset styling involving `width` that interferes with the responsive table. This cannot be overridden without a Firefox-specific hack that we **don't** provide in Bootstrap:

```
@-moz-document url-prefix() {
  fieldset { display: table-cell; }
}
```

For more information, read this Stack Overflow answer.

EXAMPLE

#	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading
1	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
2	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
3	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell

#	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading
1	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
2	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
3	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell

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

Forms

Basic example

Individual form controls automatically receive some global styling. All textual `<input>`, `<textarea>`, and `<select>` elements with `.form-control` are set to `width: 100%;` by default. Wrap labels and controls in `.form-group` for optimum spacing.

EXAMPLE

Email address

Email

Password

Password

File input

Choose file No file chosen

Example block-level help text here.

Check me out

Submit

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" placeholder="Email">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">
  </div>
  <div class="form-group">
    <label for="exampleInputFile">File input</label>
    <input type="file" id="exampleInputFile">
    <p class="help-block">Example block-level help text here.</p>
  </div>
  <div class="checkbox">
```

```
    <label>
      <input type="checkbox"> Check me out
    </label>
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

Don't mix form groups with input groups

Do not mix form groups directly with input groups. Instead, nest the input group inside of the form group.

Inline form

Add `.form-inline` to your form (which doesn't have to be a `<form>`) for left-aligned and inline-block controls. **This only applies to forms within viewports that are at least 768px wide.**

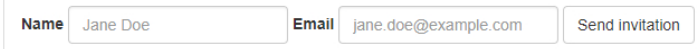
May require custom widths

Inputs and selects have `width: 100%;` applied by default in Bootstrap. Within inline forms, we reset that to `width: auto;` so multiple controls can reside on the same line. Depending on your layout, additional custom widths may be required.

Always add labels

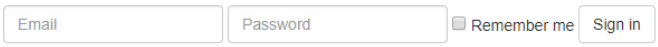
Screen readers will have trouble with your forms if you don't include a label for every input. For these inline forms, you can hide the labels using the `.sr-only` class. There are further alternative methods of providing a label for assistive technologies, such as the `aria-label`, `aria-labelledby`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

EXAMPLE




```
<form class="form-inline">
  <div class="form-group">
    <label for="exampleInputName2">Name</label>
    <input type="text" class="form-control" id="exampleInputName2" placeholder="Jane Doe">
  </div>
  <div class="form-group">
    <label for="exampleInputEmail2">Email</label>
    <input type="email" class="form-control" id="exampleInputEmail2" placeholder="jane.doe@example.com">
  </div>
  <button type="submit" class="btn btn-default">Send invitation</button>
</form>
```

EXAMPLE




```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only" for="exampleInputEmail3">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail3" placeholder="Email">
  </div>
  <div class="form-group">
    <label class="sr-only" for="exampleInputPassword3">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword3" placeholder="Password">
  </div>
  <div class="checkbox">
    <label>
      <input type="checkbox"> Remember me
    </label>
  </div>
  <button type="submit" class="btn btn-default">Sign in</button>
</form>
```

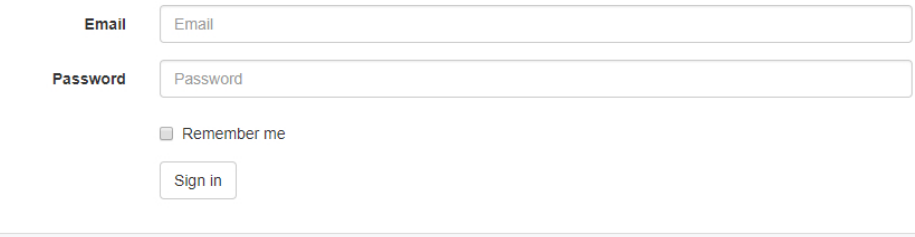
EXAMPLE


```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only" for="exampleInputAmount">Amount (in dollars)</label>
    <div class="input-group">
      <div class="input-group-addon">$</div>
      <input type="text" class="form-control" id="exampleInputAmount" placeholder="Amount">
      <div class="input-group-addon">.00</div>
    </div>
  </div>
  <button type="submit" class="btn btn-primary">Transfer cash</button>
</form>
```

Horizontal form

Use Bootstrap's predefined grid classes to align labels and groups of form controls in a horizontal layout by adding `.form-horizontal` to the form (which doesn't have to be a `<form>`). Doing so changes `.form-group`s to behave as grid rows, so no need for `.row`.

EXAMPLE




```
<form class="form-horizontal">
  <div class="form-group">
    <label for="inputEmail3" class="col-sm-2 control-label">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="inputEmail3" placeholder="Email">
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword3" class="col-sm-2 control-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword3" placeholder="Password">
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <div class="checkbox">
        <label>
          <input type="checkbox"> Remember me
        </label>
      </div>
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <button type="submit" class="btn btn-default">Sign in</button>
    </div>
  </div>
</form>
```

Supported controls

Examples of standard form controls supported in an example form layout.

Inputs

Most common form control, text-based input fields. Includes support for all HTML5 types: `text`, `password`, `datetime`, `datetime-local`, `date`, `month`, `time`, `week`, `number`, `email`, `url`, `search`, `tel`, and `color`.

Type declaration required

Inputs will only be fully styled if their `type` is properly declared.

EXAMPLE


```
<input type="text" class="form-control" placeholder="Text input">
```

Input groups

> To add integrated text or buttons before and/or after any text-based `<input>`, check out the input group component.

Textarea

Form control which supports multiple lines of text. Change `rows` attribute as necessary.

EXAMPLE

Textarea

```
<textarea class="form-control" rows="3"></textarea>
```

Checkboxes and radios

Checkboxes are for selecting one or several options in a list, while radios are for selecting one option from many.

Disabled checkboxes and radios are supported, but to provide a "not-allowed" cursor on hover of the parent `<label>`, you'll need to add the `.disabled` class to the parent `.radio`, `.radio-inline`, `.checkbox`, or `.checkbox-inline`.

Default (stacked)

EXAMPLE

- [ ] Option one is this and that—be sure to include why it's great
- [ ] Option two is disabled

- (•) Option one is this and that—be sure to include why it's great
- ( ) Option two can be something else and selecting it will deselect option one
- ( ) Option three is disabled

```
<div class="checkbox">
  <label>
    <input type="checkbox" value="">
    Option one is this and that&mdash;be sure to include why it's great
  </label>
</div>
<div class="checkbox disabled">
  <label>
    <input type="checkbox" value="" disabled>
    Option two is disabled
  </label>
</div>

<div class="radio">
  <label>
    <input type="radio" name="optionsRadios" id="optionsRadios1" value="option1" checked>
    Option one is this and that&mdash;be sure to include why it's great
  </label>
</div>
<div class="radio">
  <label>
    <input type="radio" name="optionsRadios" id="optionsRadios2" value="option2">
    Option two can be something else and selecting it will deselect option one
  </label>
</div>
<div class="radio disabled">
  <label>
    <input type="radio" name="optionsRadios" id="optionsRadios3" value="option3" disabled>
    Option three is disabled
  </label>
</div>
```

Inline checkboxes and radios

Use the `.checkbox-inline` or `.radio-inline` classes on a series of checkboxes or radios for controls that appear on the same line.

EXAMPLE

☐ 1 ☐ 2 ☐ 3

( ) 1 ( ) 2 ( ) 3

```
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox1" value="option1"> 1
</label>
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox2" value="option2"> 2
</label>
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox3" value="option3"> 3
</label>

<label class="radio-inline">
  <input type="radio" name="inlineRadioOptions" id="inlineRadio1" value="option1"> 1
</label>
<label class="radio-inline">
  <input type="radio" name="inlineRadioOptions" id="inlineRadio2" value="option2"> 2
</label>
<label class="radio-inline">
```

```
  <input type="radio" name="inlineRadioOptions" id="inlineRadio3" value="option3"> 3
</label>
```

Checkboxes and radios without label text

Should you have no text within the `<label>`, the input is positioned as you'd expect. **Currently only works on non-inline checkboxes and radios.** Remember to still provide some form of label for assistive technologies (for instance, using `aria-label`).

EXAMPLE


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

Selects

Note that many native select menus—namely in Safari and Chrome—have rounded corners that cannot be modified via `border-radius` properties.

EXAMPLE

1

```
<select class="form-control">
  <option>1</option>
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
</select>
```

For `<select>` controls with the `multiple` attribute, multiple options are shown by default.

EXAMPLE

1
2
3
4

```
<select multiple class="form-control">
  <option>1</option>
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
</select>
```

Static control

When you need to place plain text next to a form label within a form, use the `.form-control-static` class on a `<p>`.

EXAMPLE

Email email@example.com

Password Password

```
<form class="form-horizontal">
  <div class="form-group">
    <label class="col-sm-2 control-label">Email</label>
    <div class="col-sm-10">
      <p class="form-control-static">email@example.com</p>
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword" class="col-sm-2 control-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword" placeholder="Password">
    </div>
  </div>
</form>
```

EXAMPLE

email@example.com Password Confirm identity

```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only">Email</label>
    <p class="form-control-static">email@example.com</p>
  </div>
  <div class="form-group">
    <label for="inputPassword2" class="sr-only">Password</label>
    <input type="password" class="form-control" id="inputPassword2" placeholder="Password">
  </div>
  <button type="submit" class="btn btn-default">Confirm identity</button>
</form>
```

Focus state

We remove the default `outline` styles on some form controls and apply a `box-shadow` in its place for `:focus`.

EXAMPLE

Demonstrative focus state

Demo `:focus` state

The above example input uses custom styles in our documentation to demonstrate the `:focus` state on a `.form-control`.

Disabled state

Add the `disabled` boolean attribute on an input to prevent user interactions. Disabled inputs appear lighter and add a `not-allowed` cursor.

EXAMPLE

Disabled input here...

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

Disabled fieldsets

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

Caveat about link functionality of `<a>`

By default, browsers will treat all native form controls (`<input>`, `<select>` and `<button>` elements) inside a `<fieldset disabled>` as disabled, preventing both keyboard and mouse interactions on them. However, if your form also includes `<a ... class="btn btn-*">` elements, these will only be given a style of `pointer-events: none`. As noted in the section about disabled state for buttons (and specifically in the sub-section for anchor elements), this CSS property is not yet standardized and isn't fully supported in Opera 18 and below, or in Internet Explorer 11, and won't prevent keyboard users from being able to focus or activate these links. So to be safe, use custom JavaScript to disable such links.

Cross-browser compatibility

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

EXAMPLE

Disabled input

Disabled input

Disabled select menu

Disabled select

Can't check this

Submit

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

```
  </fieldset>
</form>
```

Readonly state

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

EXAMPLE

Readonly input here...

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

Help text

Block level help text for form controls.

Associating help text with form controls

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

EXAMPLE

Input with help text

A block of help text that breaks onto a new line and may extend beyond one line.

```
<label class="sr-only" for="inputHelpBlock">Input with help text</label>
<input type="text" id="inputHelpBlock" class="form-control" aria-describedby="helpBlock">
...
<span id="helpBlock" class="help-block">A block of help text that breaks onto a new line and may extend
beyond one line.</span>
```

Validation states

Bootstrap includes validation styles for error, warning, and success states on form controls. To use, add `.has-warning`, `.has-error`, or `.has-success` to the parent element. Any `.control-label`, `.form-control`, and `.help-block` within that element will receive the validation styles.

Conveying validation state to assistive technologies and colorblind users

Using these validation styles to denote the state of a form control only provides a visual, color-based indication, which will not be conveyed to users of assistive technologies - such as screen readers - or to colorblind users.

Ensure that an alternative indication of state is also provided. For instance, you can include a hint about state in the form control's `<label>` text itself (as is the case in the following code example), include a Glyphicon (with appropriate alternative text using the `.sr-only` class - see the Glyphicon examples), or by providing an additional help text block. Specifically for assistive technologies, invalid form controls can also be assigned an `aria-invalid="true"` attribute.

EXAMPLE

Input with success

A block of help text that breaks onto a new line and may extend beyond one line.

Input with warning

Input with error

- Checkbox with success
- Checkbox with warning
- Checkbox with error

```
<div class="form-group has-success">
  <label class="control-label" for="inputSuccess1">Input with success</label>
  <input type="text" class="form-control" id="inputSuccess1" aria-describedby="helpBlock2">
  <span id="helpBlock2" class="help-block">A block of help text that breaks onto a new line and may extend
beyond one line.</span>
</div>
<div class="form-group has-warning">
  <label class="control-label" for="inputWarning1">Input with warning</label>
  <input type="text" class="form-control" id="inputWarning1">
</div>
<div class="form-group has-error">
```

```
  <label class="control-label" for="inputError1">Input with error</label>
  <input type="text" class="form-control" id="inputError1">
</div>
<div class="has-success">
  <div class="checkbox">
    <label>
      <input type="checkbox" id="checkboxSuccess" value="option1">
      Checkbox with success
    </label>
  </div>
</div>
<div class="has-warning">
  <div class="checkbox">
    <label>
      <input type="checkbox" id="checkboxWarning" value="option1">
      Checkbox with warning
    </label>
  </div>
</div>
<div class="has-error">
  <div class="checkbox">
    <label>
      <input type="checkbox" id="checkboxError" value="option1">
      Checkbox with error
    </label>
  </div>
</div>
```

With optional icons

You can also add optional feedback icons with the addition of `.has-feedback` and the right icon.

Feedback icons only work with textual `<input class="form-control">` elements.

Icons, labels, and input groups

Manual positioning of feedback icons is required for inputs without a label and for input groups with an add-on on the right. You are strongly encouraged to provide labels for all inputs for accessibility reasons. If you wish to prevent labels from being displayed, hide them with the `.sr-only` class. If you must do without labels, adjust the `top` value of the feedback icon. For input groups, adjust the `right` value to an appropriate pixel value depending on the width of your addon.

Conveying the icon's meaning to assistive technologies

To ensure that assistive technologies – such as screen readers – correctly convey the meaning of an icon, additional hidden text should be included with the `.sr-only` class and explicitly associated with the form control it relates to using `aria-describedby`. Alternatively, ensure that the meaning (for instance, the fact that there is a warning for a particular text entry field) is conveyed in some other form, such as changing the text of the actual `<label>` associated with the form control.

Although the following examples already mention the validation state of their respective form controls in the `<label>` text itself, the above technique (using `.sr-only` text and `aria-describedby`) has been included for illustrative purposes.

EXAMPLE

Input with success

Input with warning

Input with error

Input group with success

@

```
<div class="form-group has-success has-feedback">
  <label class="control-label" for="inputSuccess2">Input with success</label>
  <input type="text" class="form-control" id="inputSuccess2" aria-describedby="inputSuccess2Status">
  <span class="glyphicon glyphicon-ok form-control-feedback" aria-hidden="true"></span>
  <span id="inputSuccess2Status" class="sr-only">(success)</span>
</div>
<div class="form-group has-warning has-feedback">
  <label class="control-label" for="inputWarning2">Input with warning</label>
  <input type="text" class="form-control" id="inputWarning2" aria-describedby="inputWarning2Status">
  <span class="glyphicon glyphicon-warning-sign form-control-feedback" aria-hidden="true"></span>
  <span id="inputWarning2Status" class="sr-only">(warning)</span>
</div>
<div class="form-group has-error has-feedback">
  <label class="control-label" for="inputError2">Input with error</label>
  <input type="text" class="form-control" id="inputError2" aria-describedby="inputError2Status">
  <span class="glyphicon glyphicon-remove form-control-feedback" aria-hidden="true"></span>
  <span id="inputError2Status" class="sr-only">(error)</span>
</div>
<div class="form-group has-success has-feedback">
  <label class="control-label" for="inputGroupSuccess1">Input group with success</label>
  <div class="input-group">
    <span class="input-group-addon">@</span>
    <input type="text" class="form-control" id="inputGroupSuccess1" aria-
describedby="inputGroupSuccess1Status">
```

```
  </div>
  <span class="glyphicon glyphicon-ok form-control-feedback" aria-hidden="true"></span>
  <span id="inputGroupSuccess1Status" class="sr-only">(success)</span>
</div>
```

Optional icons in horizontal and inline forms

EXAMPLE

Input with success

Input group with success @

```
<form class="form-horizontal">
  <div class="form-group has-success has-feedback">
    <label class="control-label col-sm-3" for="inputSuccess3">Input with success</label>
    <div class="col-sm-9">
      <input type="text" class="form-control" id="inputSuccess3" aria-describedby="inputSuccess3Status">
      <span class="glyphicon glyphicon-ok form-control-feedback" aria-hidden="true"></span>
      <span id="inputSuccess3Status" class="sr-only">(success)</span>
    </div>
  </div>
  <div class="form-group has-success has-feedback">
    <label class="control-label col-sm-3" for="inputGroupSuccess2">Input group with success</label>
    <div class="col-sm-9">
      <div class="input-group">
        <span class="input-group-addon">@</span>
        <input type="text" class="form-control" id="inputGroupSuccess2" aria-
describedby="inputGroupSuccess2Status">
      </div>
      <span class="glyphicon glyphicon-ok form-control-feedback" aria-hidden="true"></span>
      <span id="inputGroupSuccess2Status" class="sr-only">(success)</span>
    </div>
  </div>
</form>
```

EXAMPLE

Input with success

Input group with success @

```
<form class="form-inline">
  <div class="form-group has-success has-feedback">
    <label class="control-label" for="inputSuccess4">Input with success</label>
    <input type="text" class="form-control" id="inputSuccess4" aria-describedby="inputSuccess4Status">
    <span class="glyphicon glyphicon-ok form-control-feedback" aria-hidden="true"></span>
    <span id="inputSuccess4Status" class="sr-only">(success)</span>
  </div>
</form>
<form class="form-inline">
  <div class="form-group has-success has-feedback">
    <label class="control-label" for="inputGroupSuccess3">Input group with success</label>
    <div class="input-group">
      <span class="input-group-addon">@</span>
      <input type="text" class="form-control" id="inputGroupSuccess3" aria-
describedby="inputGroupSuccess3Status">
    </div>
    <span class="glyphicon glyphicon-ok form-control-feedback" aria-hidden="true"></span>
    <span id="inputGroupSuccess3Status" class="sr-only">(success)</span>
  </div>
</form>
```

Optional icons with hidden `.sr-only` labels

If you use the `.sr-only` class to hide a form control's `<label>` (rather than using other labelling options, such as the `aria-label` attribute), Bootstrap will automatically adjust the position of the icon once it's been added.

EXAMPLE

@

```
<div class="form-group has-success has-feedback">
  <label class="control-label sr-only" for="inputSuccess5">Hidden label</label>
  <input type="text" class="form-control" id="inputSuccess5" aria-describedby="inputSuccess5Status">
  <span class="glyphicon glyphicon-ok form-control-feedback" aria-hidden="true"></span>
  <span id="inputSuccess5Status" class="sr-only">(success)</span>
</div>
<div class="form-group has-success has-feedback">
  <label class="control-label sr-only" for="inputGroupSuccess4">Input group with success</label>
  <div class="input-group">
    <span class="input-group-addon">@</span>
    <input type="text" class="form-control" id="inputGroupSuccess4" aria-
describedby="inputGroupSuccess4Status">
  </div>
  <span class="glyphicon glyphicon-ok form-control-feedback" aria-hidden="true"></span>
  <span id="inputGroupSuccess4Status" class="sr-only">(success)</span>
</div>
```

Control sizing

Set heights using classes like `.input-lg`, and set widths using grid column classes like `.col-lg-*`.

Height sizing

Create taller or shorter form controls that match button sizes.

EXAMPLE


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

<select class="form-control input-lg">...</select>
<select class="form-control">...</select>
<select class="form-control input-sm">...</select>
```

Horizontal form group sizes

Quickly size labels and form controls within `.form-horizontal` by adding `.form-group-lg` or `.form-group-sm`.

EXAMPLE


```
<form class="form-horizontal">
  <div class="form-group form-group-lg">
    <label class="col-sm-2 control-label" for="formGroupInputLarge">Large label</label>
    <div class="col-sm-10">
      <input class="form-control" type="text" id="formGroupInputLarge" placeholder="Large input">
    </div>
  </div>
  <div class="form-group form-group-sm">
    <label class="col-sm-2 control-label" for="formGroupInputSmall">Small label</label>
    <div class="col-sm-10">
      <input class="form-control" type="text" id="formGroupInputSmall" placeholder="Small input">
    </div>
  </div>
</form>
```

Column sizing

Wrap inputs in grid columns, or any custom parent element, to easily enforce desired widths.

EXAMPLE


```
<div class="row">
  <div class="col-xs-2">
    <input type="text" class="form-control" placeholder=".col-xs-2">
  </div>
  <div class="col-xs-3">
    <input type="text" class="form-control" placeholder=".col-xs-3">
  </div>
  <div class="col-xs-4">
    <input type="text" class="form-control" placeholder=".col-xs-4">
  </div>
</div>
```

Buttons

Button tags

Use the button classes on an `<a>`, `<button>`, or `<input>` element.

EXAMPLE

Link Button Input Submit

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

Context-specific usage

While button classes can be used on `<a>` and `<button>` elements, only `<button>` elements are supported within our nav and navbar components.

Links acting as buttons

If the `<a>` elements are used to act as buttons – triggering in-page functionality, rather than navigating to another document or section within the current page – they should also be given an appropriate `role="button"`.

Cross-browser rendering

As a best practice, **we highly recommend using the `<button>` element whenever possible** to ensure matching cross-browser rendering.

Among other things, there's a bug in Firefox <30 that prevents us from setting the `line-height` of `<input>`-based buttons, causing them to not exactly match the height of other buttons on Firefox.

Options

Use any of the available button classes to quickly create a styled button.

EXAMPLE

Default Primary Success Info Warning Danger Link

```
<!-- Standard button -->
<button type="button" class="btn btn-default">Default</button>

<!-- Provides extra visual weight and identifies the primary action in a set of buttons -->
<button type="button" class="btn btn-primary">Primary</button>

<!-- Indicates a successful or positive action -->
<button type="button" class="btn btn-success">Success</button>

<!-- Contextual button for informational alert messages -->
<button type="button" class="btn btn-info">Info</button>

<!-- Indicates caution should be taken with this action -->
<button type="button" class="btn btn-warning">Warning</button>

<!-- Indicates a dangerous or potentially negative action -->
<button type="button" class="btn btn-danger">Danger</button>

<!-- Deemphasize a button by making it look like a link while maintaining button behavior -->
<button type="button" class="btn btn-link">Link</button>
```

Conveying meaning to assistive technologies

Using color to add meaning to a button 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 (the visible text of the button), or is included through alternative means, such as additional text hidden with the `.sr-only` class.

Sizes

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

EXAMPLE

Large button Large button

Default button Default button

Small button Small button

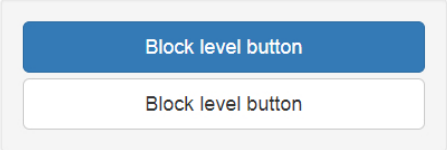
Extra small button Extra small button

```
<p>
  <button type="button" class="btn btn-primary btn-lg">Large button</button>
  <button type="button" class="btn btn-default btn-lg">Large button</button>
</p>
<p>
  <button type="button" class="btn btn-primary">Default button</button>
  <button type="button" class="btn btn-default">Default button</button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-sm">Small button</button>
```

```
  <button type="button" class="btn btn-default btn-sm">Small button</button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-xs">Extra small button</button>
  <button type="button" class="btn btn-default btn-xs">Extra small button</button>
</p>
```

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

EXAMPLE




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

Active state

Buttons will appear pressed (with a darker background, darker border, and inset shadow) when active. For `<button>` elements, this is done via `:active`. For `<a>` elements, it's done with `.active`. However, you may use `.active` on `<button>`s (and include the `aria-pressed="true"` attribute) should you need to replicate the active state programmatically.

Button element

No need to add `:active` as it's a pseudo-class, but if you need to force the same appearance, go ahead and add `.active`.

EXAMPLE


```
<button type="button" class="btn btn-primary btn-lg active">Primary button</button>
<button type="button" class="btn btn-default btn-lg active">Button</button>
```

Anchor element

Add the `.active` class to `<a>` buttons.

EXAMPLE


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

Disabled state

Make buttons look unclickable by fading them back with `opacity`.

Button element

Add the `disabled` attribute to `<button>` buttons.

EXAMPLE


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

Cross-browser compatibility

If you add the `disabled` attribute to a `<button>`, Internet Explorer 9 and below will render text gray with a nasty text-shadow that we cannot fix.

Anchor element

Add the `.disabled` class to `<a>` buttons.

EXAMPLE


```
<a href="#" class="btn btn-primary btn-lg disabled" role="button">Primary link</a>
<a href="#" class="btn btn-default btn-lg disabled" role="button">Link</a>
```

We use `.disabled` as a utility class here, similar to the common `.active` class, so no prefix is required.

> **Link functionality caveat**
>
> This class uses `pointer-events: none` to try to disable the link functionality of `<a>`s, but that CSS property is not yet standardized and isn't fully supported in Opera 18 and below, or in Internet Explorer 11. 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, use custom JavaScript to disable such links.

Images

Responsive images

Images in Bootstrap 3 can be made responsive-friendly via the addition of the `.img-responsive` class. This applies `max-width: 100%;`, `height: auto;` and `display: block;` to the image so that it scales nicely to the parent element.

To center images which use the `.img-responsive` class, use `.center-block` instead of `.text-center`. See the helper classes section for more details about `.center-block` usage.

> **SVG images and IE 8-10**
>
> In Internet Explorer 8-10, SVG images with `.img-responsive` are disproportionately sized. To fix this, add `width: 100% \9;` where necessary. Bootstrap doesn't apply this automatically as it causes complications to other image formats.

```
<img src="..." class="img-responsive" alt="Responsive image">
```

Image shapes

Add classes to an `<img>` element to easily style images in any project.

> **Cross-browser compatibility**
>
> Keep in mind that Internet Explorer 8 lacks support for rounded corners.

EXAMPLE


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

Helper classes

Contextual colors

Convey meaning through color with a handful of emphasis utility classes. These may also be applied to links and will darken on hover just like our default link styles.

EXAMPLE

Fusce dapibus, tellus ac cursus commodo, tortor mauris nibh.

Nullam id dolor id nibh ultricies vehicula ut id elit.

Duis mollis, est non commodo luctus, nisi erat porttitor ligula.

Maecenas sed diam eget risus varius blandit sit amet non magna.

Etiam porta sem malesuada magna mollis euismod.

Donec ullamcorper nulla non metus auctor fringilla.

```
<p class="text-muted">...</p>
<p class="text-primary">...</p>
<p class="text-success">...</p>
<p class="text-info">...</p>
<p class="text-warning">...</p>
<p class="text-danger">...</p>
```

> **Dealing with specificity**

Sometimes emphasis classes cannot be applied due to the specificity of another selector. In most cases, a sufficient workaround is to wrap your text in a `<span>` with the class.

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (the contextual colors are only used to reinforce meaning that is already present in the text/markup), or is included through alternative means, such as additional text hidden with the `.sr-only` class.

Contextual backgrounds

Similar to the contextual text color classes, easily set the background of an element to any contextual class. Anchor components will darken on hover, just like the text classes.

EXAMPLE

Nullam id dolor id nibh ultricies vehicula ut id elit.

Duis mollis, est non commodo luctus, nisi erat porttitor ligula.

Maecenas sed diam eget risus varius blandit sit amet non magna.

Etiam porta sem malesuada magna mollis euismod.

Donec ullamcorper nulla non metus auctor fringilla.

```
<p class="bg-primary">...</p>
<p class="bg-success">...</p>
<p class="bg-info">...</p>
<p class="bg-warning">...</p>
<p class="bg-danger">...</p>
```

Dealing with specificity

Sometimes contextual background classes cannot be applied due to the specificity of another selector. In some cases, a sufficient workaround is to wrap your element's content in a `<div>` with the class.

Conveying meaning to assistive technologies

As with contextual colors, ensure that any meaning conveyed through color is also conveyed in a format that is not purely presentational.

Close icon

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

EXAMPLE

×

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

Carets

Use carets to indicate dropdown functionality and direction. Note that the default caret will reverse automatically in dropup menus.

EXAMPLE

▾

```
<span class="caret"></span>
```

Quick floats

Float an element to the left or right with a class. `!important` is included to avoid specificity issues. Classes can also be used as mixins.

```
<div class="pull-left">...</div>
<div class="pull-right">...</div>
```

```
// Classes
.pull-left {
  float: left !important;
}
.pull-right {
  float: right !important;
}

// Usage as mixins
```

```
.element {
  .pull-left();
}
.another-element {
  .pull-right();
}
```

> **Not for use in navbars**
>
> To align components in navbars with utility classes, use `.navbar-left` or `.navbar-right` instead. See the navbar docs for details.

Center content blocks

Set an element to `display: block` and center via `margin`. Available as a mixin and class.

```
<div class="center-block">...</div>
```

```
// Class
.center-block {
  display: block;
  margin-left: auto;
  margin-right: auto;
}

// Usage as a mixin
.element {
  .center-block();
}
```

Clearfix

Easily clear `float`s by adding `.clearfix` **to the parent element**. Utilizes the micro clearfix as popularized by Nicolas Gallagher. Can also be used as a mixin.

```
<!-- Usage as a class -->
<div class="clearfix">...</div>
```

```
// Mixin itself
.clearfix() {
  &:before,
  &:after {
    content: " ";
    display: table;
  }
  &:after {
    clear: both;
  }
}

// Usage as a mixin
.element {
  .clearfix();
}
```

Showing and hiding content

Force an element to be shown or hidden (**including for screen readers**) with the use of `.show` and `.hidden` classes. These classes use `!important` to avoid specificity conflicts, just like the quick floats. They are only available for block level toggling. They can also be used as mixins.

`.hide` is available, but it does not always affect screen readers and is **deprecated** as of v3.0.1. Use `.hidden` or `.sr-only` instead.

Furthermore, `.invisible` can be used to toggle only the visibility of an element, meaning its `display` is not modified and the element can still affect the flow of the document.

```
<div class="show">...</div>
<div class="hidden">...</div>
```

```
// Classes
.show {
  display: block !important;
}
.hidden {
  display: none !important;
}
.invisible {
  visibility: hidden;
}

// Usage as mixins
.element {
  .show();
}
.another-element {
  .hidden();
}
```

Screen reader and keyboard navigation content

Hide an element to all devices **except screen readers** with `.sr-only`. Combine `.sr-only` with `.sr-only-focusable` to show the element again when it's focused (e.g. by a keyboard-only user). Necessary for following accessibility best practices. Can also be used as

mixins.

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

```
// Usage as a mixin
.skip-navigation {
  .sr-only();
  .sr-only-focusable();
}
```

Image replacement

Utilize the `.text-hide` class or mixin to help replace an element's text content with a background image.

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

```
// Usage as a mixin
.heading {
  .text-hide();
}
```

Responsive utilities

For faster mobile-friendly development, use these utility classes for showing and hiding content by device via media query. Also included are utility classes for toggling content when printed.

Try to use these on a limited basis and avoid creating entirely different versions of the same site. Instead, use them to complement each device's presentation.

Available classes

Use a single or combination of the available classes for toggling content across viewport breakpoints.

	Extra small devices Phones (<768px)	**Small devices** Tablets (≥768px)	**Medium devices** Desktops (≥992px)	**Large devices** Desktops (≥1200px)
`.visible-xs-*`	Visible	Hidden	Hidden	Hidden
`.visible-sm-*`	Hidden	Visible	Hidden	Hidden
`.visible-md-*`	Hidden	Hidden	Visible	Hidden
`.visible-lg-*`	Hidden	Hidden	Hidden	Visible
`.hidden-xs`	Hidden	Visible	Visible	Visible
`.hidden-sm`	Visible	Hidden	Visible	Visible
`.hidden-md`	Visible	Visible	Hidden	Visible
`.hidden-lg`	Visible	Visible	Visible	Hidden

As of v3.2.0, the `.visible-*-*` classes for each breakpoint come in three variations, one for each CSS `display` property value listed below.

Group of classes	**CSS `display`**
`.visible-*-block`	`display: block;`
`.visible-*-inline`	`display: inline;`
`.visible-*-inline-block`	`display: inline-block;`

So, for extra small (`xs`) screens for example, the available `.visible-*-*` classes are: `.visible-xs-block`, `.visible-xs-inline`, and `.visible-xs-inline-block`.

The classes `.visible-xs`, `.visible-sm`, `.visible-md`, and `.visible-lg` also exist, but are **deprecated as of v3.2.0**. They are approximately equivalent to `.visible-*-block`, except with additional special cases for toggling `<table>`-related elements.

Print classes

Similar to the regular responsive classes, use these for toggling content for print.

Classes	**Browser**	**Print**

`.visible-print-block`
`.visible-print-inline`
`.visible-print-inline-block`
	Hidden	Visible

`.hidden-print`	Visible	Hidden

The class `.visible-print` also exists but is **deprecated** as of v3.2.0. It is approximately equivalent to `.visible-print-block`, except with additional special cases for `<table>`-related elements.

Test cases

Resize your browser or load on different devices to test the responsive utility classes.

Visible on...

Green checkmarks indicate the element **is visible** in your current viewport.

Extra small	Small	Medium	✓ Visible on large
Extra small and small			✓ Visible on medium and large
Extra small and medium			✓ Visible on small and large
✓ Visible on x-small and large			Small and medium

Hidden on...

Here, green checkmarks also indicate the element **is hidden** in your current viewport.

Extra small	Small	Medium	✓ Hidden on large
Extra small and small			✓ Hidden on medium and large
Extra small and medium			✓ Hidden on small and large
✓ Hidden on x-small and large			Small and medium

Using Less

Bootstrap's CSS is built on Less, a preprocessor with additional functionality like variables, mixins, and functions for compiling CSS. Those looking to use the source Less files instead of our compiled CSS files can make use of the numerous variables and mixins we use throughout the framework.

Grid variables and mixins are covered within the Grid system section.

Compiling Bootstrap

Bootstrap can be used in at least two ways: with the compiled CSS or with the source Less files. To compile the Less files, consult the Getting Started section for how to setup your development environment to run the necessary commands.

Third party compilation tools may work with Bootstrap, but they are not supported by our core team.

Variables

Variables are used throughout the entire project as a way to centralize and share commonly used values like colors, spacing, or font stacks. For a complete breakdown, please see the Customizer.

Colors

Easily make use of two color schemes: grayscale and semantic. Grayscale colors provide quick access to commonly used shades of black while semantic include various colors assigned to meaningful contextual values.

EXAMPLE

```
@gray-darker:  lighten(#000, 13.5%); // #222
@gray-dark:    lighten(#000, 20%);   // #333
@gray:         lighten(#000, 33.5%); // #555
@gray-light:   lighten(#000, 46.7%); // #777
@gray-lighter: lighten(#000, 93.5%); // #eee
```

EXAMPLE

```
@brand-primary: darken(#428bca, 6.5%); // #337ab7
@brand-success: #5cb85c;
@brand-info:    #5bc0de;
@brand-warning: #f0ad4e;
@brand-danger:  #d9534f;
```

Use any of these color variables as they are or reassign them to more meaningful variables for your project.

```
// Use as-is
.masthead {
  background-color: @brand-primary;
}

// Reassigned variables in Less
@alert-message-background: @brand-info;
.alert {
  background-color: @alert-message-background;
}
```

Scaffolding

A handful of variables for quickly customizing key elements of your site's skeleton.

```
// Scaffolding
@body-bg:    #fff;
@text-color: @black-50;
```

Links

Easily style your links with the right color with only one value.

```
// Variables
@link-color:       @brand-primary;
@link-hover-color: darken(@link-color, 15%);

// Usage
a {
  color: @link-color;
  text-decoration: none;

  &:hover {
    color: @link-hover-color;
    text-decoration: underline;
  }
}
```

Note that the `@link-hover-color` uses a function, another awesome tool from Less, to automagically create the right hover color. You can use `darken`, `lighten`, `saturate`, and `desaturate`.

Typography

Easily set your typeface, text size, leading, and more with a few quick variables. Bootstrap makes use of these as well to provide easy typographic mixins.

```
@font-family-sans-serif:  "Helvetica Neue", Helvetica, Arial, sans-serif;
@font-family-serif:       Georgia, "Times New Roman", Times, serif;
@font-family-monospace:   Menlo, Monaco, Consolas, "Courier New", monospace;
@font-family-base:        @font-family-sans-serif;

@font-size-base:          14px;
@font-size-large:         ceil((@font-size-base * 1.25)); // ~18px
@font-size-small:         ceil((@font-size-base * 0.85)); // ~12px

@font-size-h1:            floor((@font-size-base * 2.6)); // ~36px
@font-size-h2:            floor((@font-size-base * 2.15)); // ~30px
@font-size-h3:            ceil((@font-size-base * 1.7)); // ~24px
@font-size-h4:            ceil((@font-size-base * 1.25)); // ~18px
@font-size-h5:            @font-size-base;
@font-size-h6:            ceil((@font-size-base * 0.85)); // ~12px

@line-height-base:        1.428571429; // 20/14
@line-height-computed:    floor((@font-size-base * @line-height-base)); // ~20px

@headings-font-family:    inherit;
@headings-font-weight:    500;
@headings-line-height:    1.1;
@headings-color:          inherit;
```

Icons

Two quick variables for customizing the location and filename of your icons.

```
@icon-font-path:          "../fonts/";
@icon-font-name:          "glyphicons-halflings-regular";
```

Components

Components throughout Bootstrap make use of some default variables for setting common values. Here are the most commonly used.

```
@padding-base-vertical:          6px;
@padding-base-horizontal:        12px;

@padding-large-vertical:         10px;
```

```
@padding-large-horizontal:     16px;

@padding-small-vertical:       5px;
@padding-small-horizontal:     10px;

@padding-xs-vertical:          1px;
@padding-xs-horizontal:        5px;

@line-height-large:            1.33;
@line-height-small:            1.5;

@border-radius-base:           4px;
@border-radius-large:          6px;
@border-radius-small:          3px;

@component-active-color:       #fff;
@component-active-bg:          @brand-primary;

@caret-width-base:             4px;
@caret-width-large:            5px;
```

Vendor mixins

Vendor mixins are mixins to help support multiple browsers by including all relevant vendor prefixes in your compiled CSS.

Box-sizing

Reset your components' box model with a single mixin. For context, see this helpful article from Mozilla.

The mixin is **deprecated** as of v3.2.0, with the introduction of Autoprefixer. To preserve backwards-compatibility, Bootstrap will continue to use the mixin internally until Bootstrap v4.

```
.box-sizing(@box-model) {
  -webkit-box-sizing: @box-model; // Safari <= 5
     -moz-box-sizing: @box-model; // Firefox <= 19
          box-sizing: @box-model;
}
```

Rounded corners

Today all modern browsers support the non-prefixed `border-radius` property. As such, there is no `.border-radius()` mixin, but Bootstrap does include shortcuts for quickly rounding two corners on a particular side of an object.

```
.border-top-radius(@radius) {
  border-top-right-radius: @radius;
   border-top-left-radius: @radius;
}
.border-right-radius(@radius) {
  border-bottom-right-radius: @radius;
     border-top-right-radius: @radius;
}
.border-bottom-radius(@radius) {
  border-bottom-right-radius: @radius;
   border-bottom-left-radius: @radius;
}
.border-left-radius(@radius) {
  border-bottom-left-radius: @radius;
     border-top-left-radius: @radius;
}
```

Box (Drop) shadows

If your target audience is using the latest and greatest browsers and devices, be sure to just use the `box-shadow` property on its own. If you need support for older Android (pre-v4) and iOS devices (pre-iOS 5), use the **deprecated** mixin to pick up the required `-webkit` prefix.

The mixin is **deprecated** as of v3.1.0, since Bootstrap doesn't officially support the outdated platforms that don't support the standard property. To preserve backwards-compatibility, Bootstrap will continue to use the mixin internally until Bootstrap v4.

Be sure to use `rgba()` colors in your box shadows so they blend as seamlessly as possible with backgrounds.

```
.box-shadow(@shadow: 0 1px 3px rgba(0,0,0,.25)) {
  -webkit-box-shadow: @shadow; // iOS <4.3 & Android <4.1
          box-shadow: @shadow;
}
```

Transitions

Multiple mixins for flexibility. Set all transition information with one, or specify a separate delay and duration as needed.

The mixins are **deprecated** as of v3.2.0, with the introduction of Autoprefixer. To preserve backwards-compatibility, Bootstrap will continue to use the mixins internally until Bootstrap v4.

```
.transition(@transition) {
  -webkit-transition: @transition;
          transition: @transition;
}
.transition-property(@transition-property) {
  -webkit-transition-property: @transition-property;
          transition-property: @transition-property;
}
.transition-delay(@transition-delay) {
  -webkit-transition-delay: @transition-delay;
          transition-delay: @transition-delay;
}
.transition-duration(@transition-duration) {
```

```
  -webkit-transition-duration: @transition-duration;
          transition-duration: @transition-duration;
}
.transition-timing-function(@timing-function) {
  -webkit-transition-timing-function: @timing-function;
          transition-timing-function: @timing-function;
}
.transition-transform(@transition) {
  -webkit-transition: -webkit-transform @transition;
     -moz-transition: -moz-transform @transition;
       -o-transition: -o-transform @transition;
          transition: transform @transition;
}
```

Transformations

Rotate, scale, translate (move), or skew any object.

The mixins are **deprecated** as of v3.2.0, with the introduction of Autoprefixer. To preserve backwards-compatibility, Bootstrap will continue to use the mixins internally until Bootstrap v4.

```
.rotate(@degrees) {
  -webkit-transform: rotate(@degrees);
      -ms-transform: rotate(@degrees); // IE9 only
          transform: rotate(@degrees);
}
.scale(@ratio; @ratio-y...) {
  -webkit-transform: scale(@ratio, @ratio-y);
      -ms-transform: scale(@ratio, @ratio-y); // IE9 only
          transform: scale(@ratio, @ratio-y);
}
.translate(@x; @y) {
  -webkit-transform: translate(@x, @y);
      -ms-transform: translate(@x, @y); // IE9 only
          transform: translate(@x, @y);
}
.skew(@x; @y) {
  -webkit-transform: skew(@x, @y);
      -ms-transform: skewX(@x) skewY(@y); // See https://github.com/twbs/bootstrap/issues/4885; IE9+
          transform: skew(@x, @y);
}
.translate3d(@x; @y; @z) {
  -webkit-transform: translate3d(@x, @y, @z);
          transform: translate3d(@x, @y, @z);
}

.rotateX(@degrees) {
  -webkit-transform: rotateX(@degrees);
      -ms-transform: rotateX(@degrees); // IE9 only
          transform: rotateX(@degrees);
}
.rotateY(@degrees) {
  -webkit-transform: rotateY(@degrees);
      -ms-transform: rotateY(@degrees); // IE9 only
          transform: rotateY(@degrees);
}
.perspective(@perspective) {
  -webkit-perspective: @perspective;
     -moz-perspective: @perspective;
          perspective: @perspective;
}
.perspective-origin(@perspective) {
  -webkit-perspective-origin: @perspective;
     -moz-perspective-origin: @perspective;
          perspective-origin: @perspective;
}
.transform-origin(@origin) {
  -webkit-transform-origin: @origin;
     -moz-transform-origin: @origin;
      -ms-transform-origin: @origin; // IE9 only
```

```
          transform-origin: @origin;
}
```

Animations

A single mixin for using all of CSS3's animation properties in one declaration and other mixins for individual properties.

The mixins are **deprecated** as of v3.2.0, with the introduction of Autoprefixer. To preserve backwards-compatibility, Bootstrap will continue to use the mixins internally until Bootstrap v4.

```
.animation(@animation) {
  -webkit-animation: @animation;
          animation: @animation;
}
.animation-name(@name) {
  -webkit-animation-name: @name;
          animation-name: @name;
}
.animation-duration(@duration) {
  -webkit-animation-duration: @duration;
          animation-duration: @duration;
}
.animation-timing-function(@timing-function) {
  -webkit-animation-timing-function: @timing-function;
          animation-timing-function: @timing-function;
}
.animation-delay(@delay) {
  -webkit-animation-delay: @delay;
          animation-delay: @delay;
}
.animation-iteration-count(@iteration-count) {
  -webkit-animation-iteration-count: @iteration-count;
          animation-iteration-count: @iteration-count;
}
.animation-direction(@direction) {
  -webkit-animation-direction: @direction;
          animation-direction: @direction;
}
```

Opacity

Set the opacity for all browsers and provide a `filter` fallback for IE8.

```
.opacity(@opacity) {
  opacity: @opacity;
  // IE8 filter
  @opacity-ie: (@opacity * 100);
  filter: ~"alpha(opacity=@{opacity-ie})";
}
```

Placeholder text

Provide context for form controls within each field.

```
.placeholder(@color: @input-color-placeholder) {
  &::-moz-placeholder           { color: @color; } // Firefox
  &:-ms-input-placeholder       { color: @color; } // Internet Explorer 10+
  &::-webkit-input-placeholder  { color: @color; } // Safari and Chrome
}
```

Columns

Generate columns via CSS within a single element.

```
.content-columns(@width; @count; @gap) {
  -webkit-column-width: @width;
     -moz-column-width: @width;
          column-width: @width;
  -webkit-column-count: @count;
     -moz-column-count: @count;
          column-count: @count;
  -webkit-column-gap: @gap;
     -moz-column-gap: @gap;
          column-gap: @gap;
}
```

Gradients

Easily turn any two colors into a background gradient. Get more advanced and set a direction, use three colors, or use a radial gradient. With a single mixin you get all the prefixed syntaxes you'll need.

```
#gradient > .vertical(#333; #000);
#gradient > .horizontal(#333; #000);
#gradient > .radial(#333; #000);
```

You can also specify the angle of a standard two-color, linear gradient:

```
#gradient > .directional(#333; #000; 45deg);
```

If you need a barber-stripe style gradient, that's easy, too. Just specify a single color and we'll overlay a translucent white stripe.

```
#gradient > .striped(#333; 45deg);
```

Up the ante and use three colors instead. Set the first color, the second color, the second color's color stop (a percentage value like

25%), and the third color with these mixins:

```
#gradient > .vertical-three-colors(#777; #333; 25%; #000);
#gradient > .horizontal-three-colors(#777; #333; 25%; #000);
```

Heads up! Should you ever need to remove a gradient, be sure to remove any IE-specific `filter` you may have added. You can do that by using the `.reset-filter()` mixin alongside `background-image: none;`.

Utility mixins

Utility mixins are mixins that combine otherwise unrelated CSS properties to achieve a specific goal or task.

Clearfix

Forget adding `class="clearfix"` to any element and instead add the `.clearfix()` mixin where appropriate. Uses the micro clearfix from Nicolas Gallagher.

```
// Mixin
.clearfix() {
  &:before,
  &:after {
    content: " ";
    display: table;
  }
  &:after {
    clear: both;
  }
}

// Usage
.container {
  .clearfix();
}
```

Horizontal centering

Quickly center any element within its parent. **Requires `width` or `max-width` to be set.**

```
// Mixin
.center-block() {
  display: block;
  margin-left: auto;
  margin-right: auto;
}

// Usage
.container {
  width: 940px;
  .center-block();
}
```

Sizing helpers

Specify the dimensions of an object more easily.

```
// Mixins
.size(@width; @height) {
  width: @width;
  height: @height;
}
.square(@size) {
  .size(@size; @size);
}

// Usage
.image { .size(400px; 300px); }
.avatar { .square(48px); }
```

Resizable textareas

Easily configure the resize options for any textarea, or any other element. Defaults to normal browser behavior (`both`).

```
.resizable(@direction: both) {
  // Options: horizontal, vertical, both
  resize: @direction;
  // Safari fix
  overflow: auto;
}
```

Truncating text

Easily truncate text with an ellipsis with a single mixin. **Requires element to be `block` or `inline-block` level.**

```
// Mixin
.text-overflow() {
  overflow: hidden;
  text-overflow: ellipsis;
  white-space: nowrap;
}

// Usage
.branch-name {
  display: inline-block;
  max-width: 200px;
  .text-overflow();
```

```
}
```

Retina images

Specify two image paths and the @1x image dimensions, and Bootstrap will provide an @2x media query. **If you have many images to serve, consider writing your retina image CSS manually in a single media query.**

```
.img-retina(@file-1x; @file-2x; @width-1x; @height-1x) {
  background-image: url("@{file-1x}");

  @media
  only screen and (-webkit-min-device-pixel-ratio: 2),
  only screen and (   min--moz-device-pixel-ratio: 2),
  only screen and (     -o-min-device-pixel-ratio: 2/1),
  only screen and (        min-device-pixel-ratio: 2),
  only screen and (                min-resolution: 192dpi),
  only screen and (                min-resolution: 2dppx) {
    background-image: url("@{file-2x}");
    background-size: @width-1x @height-1x;
  }
}

// Usage
.jumbotron {
  .img-retina("/img/bg-1x.png", "/img/bg-2x.png", 100px, 100px);
}
```

Using Sass

While Bootstrap is built on Less, it also has an official Sass port. We maintain it in a separate GitHub repository and handle updates with a conversion script.

What's included

Since the Sass port has a separate repo and serves a slightly different audience, the contents of the project differ greatly from the main Bootstrap project. This ensures the Sass port is as compatible with as many Sass-based systems as possible.

Path	Description
`lib/`	Ruby gem code (Sass configuration, Rails and Compass integrations)
`tasks/`	Converter scripts (turning upstream Less to Sass)
`test/`	Compilation tests
`templates/`	Compass package manifest
`vendor/assets/`	Sass, JavaScript, and font files
`Rakefile`	Internal tasks, such as rake and convert

Visit the Sass port's GitHub repository to see these files in action.

Installation

For information on how to install and use Bootstrap for Sass, consult the GitHub repository readme. It's the most up to date source and includes information for use with Rails, Compass, and standard Sass projects.

Bootstrap for Sass