CSS

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

Looking for Bootstrap 2.3.2 docs? (../2.3.2/) We've moved it to a new home while we push forward with Bootstrap 3. Read the

Overview

Get the lowdown on the key pieces of Bootstrap's infrastructure, including our approach t 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=1.0">
```

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=1.0, maximum-scale=1.0, user-scalable
o">
```

Responsive images

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

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

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

For improved cross-browser rendering, we use Normalize (http://necolas.github.io/normalize.css/), a project by Nicolas Gallagher (http://twitter.com/necolas) and Jonathan Neal (http://twitter.com/jon_neal).

Containers

Easily center a page's contents by wrapping its contents in a `.container`. Containers set `width` at various media query breakp to match our grid system.

Note that, due to `padding` and fixed widths, containers are not nestable by default.

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

Grid system

Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up 12 columns as the device or viewport size increases. It includes predefined classes for eas 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` 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 u for more semantic layouts.
- Columns create gutters (gaps between column content) via `padding`. That padding is offset in rows for the first and last co via negative margin on `.row`s.
- 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`.

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

> **Grids and full-width layouts**
> Folks looking to create fully fluid layouts (meaning your site stretches the entire width of the viewport) must wrap their grid content in a containing element with `padding: 0 15px;` to offset the `margin: 0 -15px;` used on `.row`s.

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** Deskto (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
Max container width	None (auto)	750px	970px	1170px
Class prefix	`.col-xs-`	`.col-sm-`	`.col-md-`	`.col-lg-`
# of columns	12			
Max column width	Auto	60px	78px	95px
Gutter width	30px (15px on each side of a column)			
Nestable	Yes			
Offsets	Yes			
Column ordering	Yes			

Grid classes apply to devices with screen widths greater than or equal to the breakpoint sizes, and override grid classes targeted a smaller devices. Therefore, applying any `.col-md-` class to an element will not only affect its styling on medium devices but also large devices if a `.col-lg-` class is not present.

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 t 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	.c m

.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: 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, desktops

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"></div>
  <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</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 r 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"></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**. Those resets are availa for medium and large grid tiers only, since they start only at the (second) small grid tier. See this in action in the grid example (../examples/grid/).

```
<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
ol-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-
fset-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. 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

.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>
```

Nesting columns

To nest your content with the default grid, add a new `.row` and set of `.col-md-*` columns within an existing `.col-md-*` colum Nested rows should include a set of columns that add up to 12.

Level 1: .col-md-9

Level 2: .col-md-6 Level 2: .col-md-6

```
<div class="row">
  <div class="col-md-9">
    Level 1: .col-md-9
    <div class="row">
      <div class="col-md-6">
        Level 2: .col-md-6
      </div>
      <div class="col-md-6">
        Level 2: .col-md-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 u 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 usi 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 matc 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. 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 auc fringilla. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla n 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 no 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 the base font-size used throughout and the second is the base line-height. We use those variables and some simple math to creat margins, paddings, and line-heights of all our type and more. Customize them and Bootstrap adapts.

Emphasis

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 rece 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 importanc 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 te

```
<p class="text-left">Left aligned text.</p>
<p class="text-center">Center aligned text.</p>
<p class="text-right">Right aligned text.</p>
```

Emphasis classes

Convey meaning through color with a handful of emphasis utility classes. These may also be applied to links and will darken on ho 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.

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 o hover.

Basic abbreviation

For expanded text on long hover of an abbreviation, include the `title` attribute with the `<abbr>` element.

EXAMPLE

An abbreviation of the word attribute is attr (attribute).

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

Initialism

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

EXAMPLE

HTML (HYPERTEXT MARKUP LANGUAGE) 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.
795 Folsom Ave, Suite 600
San Francisco, CA 94107
P: (Phone) (123) 456-7890

Full Name
first.last@example.com (mailto:#)

```
<address>
  <strong>Twitter, Inc.</strong><br>
  795 Folsom Ave, Suite 600<br>
  San Francisco, CA 94107<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 (HyperText Markup Language) 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 `<small>` tag or `.small` class 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>
  <small>Someone famous in <cite title="Source Title">Source Title</cite></small>
</blockquote>
```

Alternate displays

Use `.pull-right` for a floated, right-aligned blockquote.

EXAMPLE

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

```
<blockquote class="pull-right">
  ...
</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 childre 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, s
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.
```

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.

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 cus table styles.

EXAMPLE

#	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
	Mark	Otto	@TwBootstrap
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
`.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

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

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

Responsive tables

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

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 w `.form-control` are set to `width: 100%;` by default. Wrap labels and controls in `.form-group` for optimum spacing.

EXAMPLE

Email address

Enter email

Password

Password

File input

Choose File no file selected

Example block-level help text here.

Check me out

Submit

```
<form role="form">
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" placeholder="Enter 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>
```

Inline form

Add `.form-inline` to your `<form>` for left-aligned and inline-block controls. **This only applies to forms within viewports that at least 768px wide.**

Requires custom widths

Inputs, selects, and textareas are 100% wide by default in Bootstrap. To use the inline form, you'll have to set a width on the form controls used within.

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 hid the labels using the `.sr-only` class.

EXAMPLE

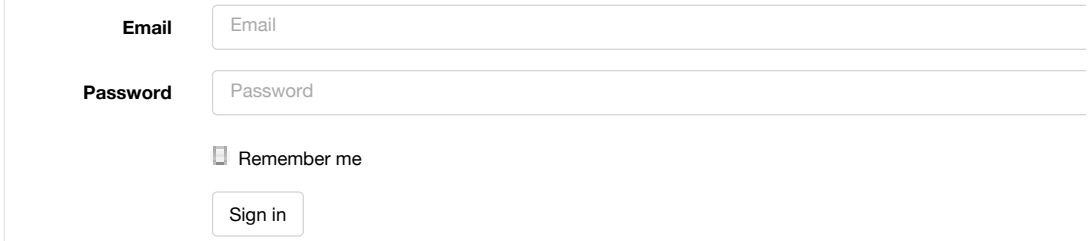
Enter email Password Remember me Sign in

```
<form class="form-inline" role="form">
  <div class="form-group">
    <label class="sr-only" for="exampleInputEmail2">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail2" placeholder="Enter email">
  </div>
  <div class="form-group">
    <label class="sr-only" for="exampleInputPassword2">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword2" 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>
```

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. Doing so changes `.form-group`s to behave as grid rows, so no need for `.row`.

EXAMPLE




```
<form class="form-horizontal" role="form">
  <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

Text input

```
<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 (../components/#input-groups).

Textarea

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

EXAMPLE

```
<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.

Default (stacked)

EXAMPLE

Option one is this and that—be sure to include why it's great

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

```
<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="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>
```

Inline checkboxes

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

EXAMPLE

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>
```

Selects

Use the default option, or add `multiple` to show multiple options at once.

EXAMPLE

1

1
2
3
4
5

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

<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 horizontal form, use the `.form-control-static` class on a `<p>`

EXAMPLE

Email email@example.com

Password Password

```
<form class="form-horizontal" role="form">
  <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>
```

Form states

Provide feedback to users or visitors with basic feedback states on form controls and labels.

Input focus

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

EXAMPLE

This is focused...

```
<input class="form-control" id="focusedInput" type="text" value="This is focused...">
```

Disabled inputs

Add the `disabled` attribute on an input to prevent user input and trigger a slightly different look.

EXAMPLE

Disabled input here...

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

Disabled fieldsets

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

Link functionality of `<a>` not impacted

This class will only change the appearance of `<a class="btn btn-default">` buttons, not their functionality. Use custom JavaScript to disable links here.

Cross-browser compatibility

While Bootstrap will apply these styles in all browsers, Internet Explorer 9 and below don't actually 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 role="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>
```

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 tha element will receive the validation styles.

EXAMPLE

Input with success

Input with warning

Input with error

```
<div class="form-group has-success">
  <label class="control-label" for="inputSuccess">Input with success</label>
  <input type="text" class="form-control" id="inputSuccess">
</div>
<div class="form-group has-warning">
  <label class="control-label" for="inputWarning">Input with warning</label>
  <input type="text" class="form-control" id="inputWarning">
</div>
<div class="form-group has-error">
  <label class="control-label" for="inputError">Input with error</label>
  <input type="text" class="form-control" id="inputError">
</div>
```

Control sizing

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

Height sizing

Create larger or smaller form controls that match button sizes.

EXAMPLE

.input-lg

Default input

.input-sm

.input-lg

Default select

.input-sm

```
<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>
```

Column sizing

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

EXAMPLE

.col-xs-2 .col-xs-3 .col-xs-4

```
<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>
```

Help text

Block level help text for form controls.

EXAMPLE

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

```
<span class="help-block">A block of help text that breaks onto a new line and may extend beyond one li
.</span>
```

Buttons

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>
```

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, t done via `:active`. For `<a>` elements, it's done with `.active`. However, you may use `.active` `<button>` s should you need replicate the active state progammatically.

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 50%.

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-shado that we cannot fix.

Anchor element

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

EXAMPLE

Primary link Link

```
<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 not impacted**
> This class will only change the `<a>`'s appearance, not its functionality. Use custom JavaScript to disable links here.

> **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.

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">
```

> **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 Firefox bug (https://bugzilla.mozilla.org/show_bug.cgi?id=697451) 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.

Images

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">
```

Responsive images

Looking for how to make images more responsive? Check out the responsive images section up top.

Helper classes

Close icon

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

EXAMPLE

×

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

Carets

Use carets to indicate dropdown functionality and direction. Note that the default caret will reverse automatically in dropup menus (../components/#btn-dropdowns-dropup).

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 (../components/#navbar-component-alignment) 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>
```

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

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

Clearfix

Clear the `float` on any element with the `.clearfix` class. Utilizes the micro clearfix (http://nicolasgallagher.com/micro-clearfix hack/) 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 cla use `!important` to avoid specificity conflicts, just like the quick floats. They are only available for block level toggling. They can a 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` ins

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;
  visibility: hidden !important;
}
```

```
.invisible {
  visibility: hidden;
}

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

Screen reader content

Hide an element to all devices **except screen readers** with `.sr-only`. Necessary for following accessibility best practices (../get started#accessibility). Can also be used as a mixin.

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

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

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 wh printed.

Try to use these on a limited basis and avoid creating entirely different versions of the same site. Instead, use them to complemen each device's presentation. **Responsive utilities are currently only available for block and table toggling.** Use with inline and t elements is currently not supported.

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

Print classes

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

Class	Browser	Print
`.visible-print`	Hidden	Visible
`.hidden-print`	Visible	Hidden

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**	