

 **FREE eBook**

LEARNING twitter-bootstrap

Free unaffiliated eBook created from
Stack Overflow contributors.

#twitter-
bootstrap

www.dbooks.org

Table of Contents

About.....	1
Chapter 1: Getting started with twitter-bootstrap.....	2
Remarks.....	2
Versions.....	2
Examples.....	3
Installation/Setup.....	3
Basic Template.....	5
When to use Bootstrap.....	6
Basic webpage using bootstrap components.....	6
Chapter 2: Alert.....	9
Remarks.....	9
Examples.....	9
Alert Types.....	9
Alert basic example.....	9
Animated Alerts.....	10
Dismissible Alerts.....	10
Link color in Alerts.....	10
Chapter 3: Bootstrap Affix.....	12
Examples.....	12
On Navbar.....	12
Affix Example 2.....	12
Chapter 4: Bootstrap Badges and Labels.....	14
Examples.....	14
Badges.....	14
Labels.....	14
Chapter 5: Bootstrap Components.....	15
Remarks.....	15
Examples.....	15
Examples of Bootstrap Components.....	15
Chapter 6: Bootstrap Containers.....	16

Introduction.....	16
Examples.....	16
Containers.....	16
Chapter 7: Bootstrap Dropdowns.....	17
Parameters.....	17
Remarks.....	17
Examples.....	17
How to Use.....	17
Basic Example.....	17
Chapter 8: Bootstrap Navbar.....	19
Examples.....	19
Bootstrap Navbar.....	19
Bootstrap Brand Image.....	19
Chapter 9: Bootstrap Themes.....	21
Examples.....	21
Bootstrap themes versus rule overrides.....	21
Chapter 10: Bootstrap Validation.....	22
Remarks.....	22
Examples.....	22
Using ASP.NET MVC and Data Annotations.....	22
Example input that requires validation.....	23
Optional.....	23
Chapter 11: Buttons.....	25
Syntax.....	25
Examples.....	25
Button Classes.....	25
Chapter 12: Carousels.....	27
Remarks.....	27
Examples.....	27
Basic HTML usage.....	27
Basic Javascript usage and initialization.....	28

Chapter 13: Columns	29
Examples	29
Responsive columns same height (CSS or SASS only)	29
Chapter 14: Dropdowns	33
Remarks	33
Examples	33
Basic HTML usage	33
Chapter 15: Forms	34
Examples	34
Basic form	34
Read-only and disabled inputs	34
Chapter 16: Glyphicons	35
Remarks	35
Examples	35
How to Use Glyphicons	35
Chapter 17: Grid Nesting	37
Introduction	37
Remarks	37
Examples	37
Nesting columns	37
Chapter 18: Grid system	38
Introduction	38
Remarks	38
Examples	38
Media Queries	38
Bootstrap Grid Tiers (Breakpoints)	38
Bootstrap Rows & Columns	40
Containers	41
Offsetting columns	42
Column order manipulation using push and pull	42
Chapter 19: Jumbotron	43

Introduction.....	43
Remarks.....	43
Examples.....	43
Basic jumbotron with two lines of text and a button.....	43
Chapter 20: List group.....	44
Remarks.....	44
Examples.....	44
Basic example.....	44
Badges.....	44
Linked Items.....	44
Button items.....	44
Disabled Items.....	45
Contextual classes.....	45
Custom content.....	45
Chapter 21: Migrating to Bootstrap 4.....	46
Introduction.....	46
Remarks.....	46
Examples.....	46
Column layout changes of grid system in Bootstrap 4.....	46
Grid Layout Bootstrap 4.....	46
Browser support changes.....	47
Affix class removal.....	47
Bootstrap 4 Navbar.....	50
Bootstrap 3 to Bootstrap 4 CSS Changes.....	52
Bootstrap 4 Vertical Align.....	55
Bootstrap 4 Centering.....	57
Horizontal Center.....	57
Vertical Center.....	58
Bootstrap 4 Column Order.....	58
Chapter 22: Modal Dialogs.....	60
Remarks.....	60
Examples.....	60

Basic HTML usage.....	60
Basic Javascript usage and initialization.....	60
Chapter 23: Modals.....	62
Remarks.....	62
Examples.....	62
Basic HTML Modal.....	62
Chapter 24: Navbar.....	63
Examples.....	63
Basic Navbar (fixed at the top of page).....	63
Submenu in navbar.....	63
Navbar divider.....	64
Keep current navigation link "active".....	64
Change Navbar breakpoint (mobile vs normal).....	64
Close collapsed navbar when clicking outside of the navbar.....	65
Chapter 25: Navigation Menus.....	66
Examples.....	66
Horizontal Pill Menu.....	66
Vertical Pill Menu.....	66
Full Width Responsive Horizontal Pill.....	66
Chapter 26: Navs.....	67
Examples.....	67
Bootstrap Navs.....	67
Chapter 27: Pagination.....	68
Introduction.....	68
Examples.....	68
A simple Pagination example.....	68
Chapter 28: Panels.....	69
Remarks.....	69
Examples.....	69
Basic example.....	69
Panel with heading.....	69
Panel with footer.....	69

Chapter 29: Printing in Bootstrap.....	71
Examples.....	71
Basic HTML usage.....	71
Chapter 30: Tables.....	72
Examples.....	72
Simple Table.....	72
Chapter 31: Tables.....	73
Remarks.....	73
Examples.....	73
Basic table.....	73
Table with advanced styling.....	73
Striped rows.....	73
Bordered table.....	74
Hover on rows.....	74
Condensed table.....	74
Contextual classes.....	74
Responsive tables.....	75
Table Reflow - Vertical headers.....	75
Chapter 32: Tabs.....	77
Examples.....	77
Basic HTML.....	77
Animated Tabs.....	77
Chapter 33: Tooltip.....	79
Remarks.....	79
Examples.....	79
Positioning Tooltips.....	79
Basic Example.....	79
Chapter 34: Twitter Bootstrap Style Customization.....	81
Remarks.....	81
Examples.....	81
Overriding Default CSS.....	81

Chapter 35: Using Clearfix in Rows and Cols	83
Introduction	83
Remarks	83
Examples	83
The Naive First Attempt	83
The Height Problem	84
Clearfix to the Rescue	86
A Dashboard	89
2,4,6 Layout with Clearfixes	94
Why Would Bootstrap Columns Exceed 12 in a Row?	96
Chapter 36: Utility Classes	99
Examples	99
Generate .hidden-* classes for all breakpoints - SCSS	99
Credits	100

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [twitter-bootstrap](#)

It is an unofficial and free twitter-bootstrap ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official twitter-bootstrap.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with twitter-bootstrap

Remarks

Bootstrap is a HTML, CSS, and JavaScript framework used to create websites that are created using a mobile-first paradigm as well as responsive web design (RWD). It uses a combination of premade CSS classes and JavaScript to make a variety of things on the web. It includes things such as a custom, responsive grid that allows websites to be viewed in on any screen, a dropdown navbar that is capable of being responsive and even simple things that can be time intensive such as premade buttons, forms and accordions to name a few.

Bootstrap can be useful for the following reasons:

1. **Time savings:** Bootstrap features many things that are pre-built, and simply need to be called upon, when writing code. This saves a considerable amount of time, and can greatly reduce the time needed to code a website.
2. **Built with responsive web design in mind:** Bootstrap allows web developers to not be concerned about creating things that will scale with the size of their screen, as Bootstrap uses mobile-first, responsive design that allows them to build things that will work on any screen size.
3. **Simplifies design process:** Bootstrap comes prebuilt with elements that have good design practices. This can be useful for those whose web design skills are not that great, or for those who view design as a tedious task, as many of Bootstrap classes are aesthetically pleasing and great to look at.

This section should mention any large subjects within twitter-bootstrap, and link out to the related topics. Since the Documentation for twitter-bootstrap is new, you may need to create initial versions of those related topics.

Versions

Version	Release Date
4.0	2999-01-01
3.3.7	2016-07-25
3.3.6	2015-11-24
3.3.5	2015-06-15

Version	Release Date
3.3.4	2015-03-16
3.3.1	2014-11-12
3.3.0	2014-10-29
3.2.0	2014-06-26
3.1.0	2014-01-30
3.0	2013-08-19
2.3	2013-02-07
2.2	2012-10-29
2.1	2012-08-20
2.0	2012-02-01
1.0	2011-08-18

Examples

Installation/Setup

Downloading:

- Download Bootstrap [directly](#) or clone, etc. from the [GitHub](#) repository
- Download your customized version of Bootstrap from official [docs](#)
- Install with bower: `bower install bootstrap`
- Install with npm: `npm install bootstrap`
- Install with composer: `composer require twbs/bootstrap`

The File Structure

```
bootstrap/
|— css/
|   |— bootstrap.css
|   |— bootstrap.min.css
|   |— bootstrap-theme.css
|   |— bootstrap-theme.min.css
|— js/
|   |— bootstrap.js
|   |— bootstrap.min.js
|— fonts/
|   |— glyphsicons-halflings-regular.eot
|   |— glyphsicons-halflings-regular.svg
|   |— glyphsicons-halflings-regular.ttf
|   |— glyphsicons-halflings-regular.woff
```

Installing:

Within your HTML page, include Bootstrap's CSS, JS, and the dependency of jQuery (pre version 3, at least as of the latest Bootstrap version). Please note that if you plan to utilize Bootstrap's JavaScript features, your jQuery reference must come *before* your bootstrap.js reference within your HTML.

You can utilize your installed Bootstrap files from the above section, or reference a CDN provided by the makers of Bootstrap (links taken from [Getting Started with Bootstrap](#)):

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiisSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">

<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-
theme.min.css" integrity="sha384-
rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXw1/Sp" crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA712mCWNIPg9mGCD8wGNICPD7Txa"
crossorigin="anonymous"></script>
```

A very basic Bootstrap webpage:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other head content must come
    *after* these tags -->
```

```

<title>Bootstrap 101 Template</title>

<!-- Bootstrap -->
<link href="css/bootstrap.min.css" rel="stylesheet">

<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
<!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
    <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
<![endif]-->
</head>
<body>
    <h1>Hello, world!</h1>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="js/bootstrap.min.js"></script>
</body>
</html>

```

Basic Template

```

<!DOCTYPE html>
<html lang="en">

    <head>

        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <!-- The above 3 meta tags *must* come first in the head; any other head content must
come *after* these tags -->

        <title>Bootstrap 101 Template</title>    <!-- The title of the Website -->

        <!-- Reference to Bootstrap's CSS file -->
        <!-- This is the line to reference the bootstrap's Stylesheet -->
        <link href="css/bootstrap.min.css" rel="stylesheet">

        <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
        <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
        <!-- [if lt IE 9] -->
        <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
        <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
        <!--[endif]-->

    </head>

    <body>

        <h1>Hello, world!</h1>

        <!-- Referencing jQuery (necessary for Bootstrap JavaScript plugins (bootstrap.min.js)

```

```

to work) -->
    <script src =
"https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

    <!-- Referencing Javascript Bootstrap Plugin to Facilitate Bootstrap Animations and
functionalities. -->
    <!-- (Necessary to run Bootstrap) -->
    <script src="js/bootstrap.min.js"></script>

</body>
</html>

```

When to use Bootstrap

Bootstrap is an opinionated framework for HTML, CSS and Javascript. It contains basic styling and functionality for what have become accepted [User Interface] elements, such as form elements, buttons, modal windows and navigation elements.

Bootstrap is a responsive web framework, meaning it is designed to adapt layout and design for screen sizes large and small, such as mobile devices, tablets and desktop computers, all in a single code base.

One of the fundamental concepts of Bootstrap is the grid framework. By applying classes to HTML elements, it is possible to create intricate layouts using a basic grid of twelve columns. For example, a four column layout might adapt to two columns on tablet devices and one column on mobile devices. The grid uses `media queries`, a CSS method for targeting specific screen sizes, to achieve this.

Bootstrap performs particularly well if:

- Custom design is not a top priority
- You are more comfortable editing HTML and adding classes than you are creating custom CSS
- You are comfortable using a framework that will have many visual similarities to many other websites

Bootstrap can be used by those who are new to HTML, CSS and Javascript, since the [documentation](#) is excellent. However, there is a learning curve for those not entirely comfortable with the three basic technologies used by Bootstrap (HTML, CSS and Javascript).

It is possible to purchase or download Bootstrap themes in order to alter the style or functionality of Bootstrap. It is also possible to use Bootstrap as a starting point, with customization of CSS and Javascript.

Basic webpage using bootstrap components

```

<!DOCTYPE html>
<html lang="en">
<head>

```

```

<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<!-- The above 3 meta tags *must* come first in the head; any other head content must come
*after* these tags -->
<title>Bootstrap 101 Template</title>

<!-- Bootstrap -->
<link href="css/bootstrap.min.css" rel="stylesheet">

<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
<!--[if lt IE 9]>
  <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
  <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
<![endif]-->
</head>
<body>
  <!-- Fixed navbar -->
  <nav class="navbar navbar-default navbar-fixed-top">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
data-target="#navbar" aria-expanded="false" aria-controls="navbar">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <a class="navbar-brand" href="#">Project name</a>
      </div>
      <div id="navbar" class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
          <li class="active"><a href="#">Home</a></li>
          <li><a href="#about">About</a></li>
          <li><a href="#contact">Contact</a></li>
          <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button"
aria-haspopup="true" aria-expanded="false">Dropdown <span class="caret"></span></a>
            <ul class="dropdown-menu">
              <li><a href="#">Action</a></li>
              <li><a href="#">Another action</a></li>
              <li><a href="#">Something else here</a></li>
              <li role="separator" class="divider"></li>
              <li class="dropdown-header">Nav header</li>
              <li><a href="#">Separated link</a></li>
              <li><a href="#">One more separated link</a></li>
            </ul>
          </li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
          <li><a href="../navbar/">Default</a></li>
          <li><a href="../navbar-static-top/">Static top</a></li>
          <li class="active"><a href="#">Fixed top <span class="sr-
only">(current)</span></a></li>
        </ul>
      </div><!--/.nav-collapse -->
    </div>
  </nav>

  <div class="container">

```

```

    <div class="jumbotron">
      <h1>Navbar example</h1>
      <p>This example is a quick exercise to illustrate how the default, static and
fixed to top navbar work. It includes the responsive CSS and HTML, so it also adapts to your
viewport and device.</p>
      <p>To see the difference between static and fixed top navbars, just scroll.</p>
      <p>
        <a class="btn btn-lg btn-primary" href="../../components/#navbar"
role="button">View navbar docs &raquo;</a>
      </p>
    </div>
  </div> <!-- /container -->

  <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
  <!-- Include all compiled plugins (below), or include individual files as needed -->
  <script src="js/bootstrap.min.js"></script>
</body>
</html>

```

Read [Getting started with twitter-bootstrap](https://riptutorial.com/twitter-bootstrap/topic/818/getting-started-with-twitter-bootstrap) online: <https://riptutorial.com/twitter-bootstrap/topic/818/getting-started-with-twitter-bootstrap>

Chapter 2: Alert

Remarks

See more: <http://getbootstrap.com/components/#alerts>

Examples

Alert Types

Unlike some other Bootstrap components like [Buttons](#), the [Alerts](#) do **not** come with a default or primary styling, because they are meant to alert the user in a specific way.

```
<div class="alert alert-success" role="alert">
  Some action was completed successfully
</div>
<div class="alert alert-info" role="alert">
  Here is some information. Just FYI.
</div>
<div class="alert alert-warning" role="alert">
  Careful! You're about to do something dangerous.
</div>
<div class="alert alert-danger" role="alert">
  An error (or something dangerous) happened!
</div>
```

Alert basic example

```
<div class="container">
  <h2>Alerts</h2>
  <div class="alert alert-success">
    <strong>Success!</strong>
  </div>
  <div class="alert alert-info">
    <strong>Info!</strong>
  </div>
  <div class="alert alert-warning">
    <strong>Warning!</strong> All foelds are required
  </div>
  <div class="alert alert-danger">
    The username is required and can't be empty
  </div>
</div>
```

Alerts

Success!

Info!

Warning! This alert box could indicate a warning that might need attention.

The username is required and can't be empty

Animated Alerts

The `.fade` and `.in` classes adds a fading effect when closing the alert message.

```
<div class="alert alert-success fade in">
  <a href="#" class="close" data-dismiss="alert" aria-label="close">&times;</a>
  <strong>Success!</strong> This is a good example!
</div>
```

Dismissible Alerts

To give an alert close functionality, all we need is to add `data-dismiss="alert"` to our close button.

```
<div class="alert alert-info alert-dismissible" role="alert">
  <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
  Sphinx of black quartz, judge my vow
</div>
```

Sphinx of black quartz, judge my vow



`.alert-dismissible` and `.close` classes are optional, only useful for styling.

Link color in Alerts

To quickly provide a matching color for links inside any alert, we can use the `.alert-link` utility class.

```
<div class="alert alert-success">
  You have won! Click <a href="#" class="alert-link">here</a> to claim your prize ...
</div>

<div class="alert alert-info">
  You might want to check <a href="#" class="alert-link">this</a> instead.
</div>

<div class="alert alert-warning">
  You are running out of coins. Buy more <a href="#" class="alert-link">here</a>.
</div>

<div class="alert alert-danger">
  Something went wrong. You can try <a href="#" class="alert-link">again</a> or ...
</div>
```

Congratulations! You have **WON!** Click **here** to claim your prize ...

You might want to check **this** instead.

You are running out of coins. Buy more **here**.

Something went wrong. You can try **again** or ...

Read Alert online: <https://riptutorial.com/twitter-bootstrap/topic/6434/alert>

Chapter 3: Bootstrap Affix

Examples

On Navbar

Html:

```
<nav class="navbar navbar-default" data-offset-top="120" data-spy="affix" >
    ...
</nav>
```

Css:

```
<style>
.navbar {
    background-color: red;
}
.navbar.affix {
    background-color: green;
}

</style>
```

Affix Example 2

```
<div class="container" id="con">
  <div class="row">
    <div class="span12">
      <div class="well">
        <h1> Header </h1>
      </div>
    </div>
  </div>
</div>
<div class="container" data-spy="affix" data-offset-top="400" id="nav">
  <div class="navbar">
    <div class="navbar-inner">
      <div class="container">
        <div class="span12">
          <a class="brand" href="#">Home</a>
          <ul class="nav">
            <li class="active"><a href="#">Home</a></li>
            <li><a href="#">Link</a></li>
            <li><a href="#">Link</a></li>
          </ul>
        </div>
      </div>
    </div>
  </div>
</div>
<div class="container">
  <div class="span3">
```

```
<p style="height:1000px;padding:10px;">
```

Long scrolling text here... Ovi lispmd idr. Blah goo bar forr foo. Ovi lispmd idr. Blah goo bar forr foo. Ovi lispmd idr. Blah goo bar forr foo. Ovi lispmd idr. Blah goo bar forr foo. Ovi lispmd idr. Blah goo bar forr foo. Ovi lispmd idr. Blah goo bar forr foo. Ovi lispmd idr. Blah goo bar forr foo. Ovi lispmd idr. Blah goo bar forr foo. Ovi lispmd idr. Blah goo bar forr foo. Ovi lispmd idr. Blah goo bar forr foo.

```
</p>
```

```
</div>
```

```
</div>
```

```
<style>
```

```
#con .well {  
  height:400px;  
}
```

```
#nav.affix {  
  position: fixed;  
  top: 0;  
  width: 100%  
}
```

```
</style>
```

```
<script>
```

```
  $('#nav').affix();  
</script>
```

Read Bootstrap Affix online: <https://riptutorial.com/twitter-bootstrap/topic/6639/bootstrap-affix>

Chapter 4: Bootstrap Badges and Labels

Examples

Badges

Badges are numerical indicators of how many items are associated with a link:

Use the `.badge` class within `<span>` elements to create badges:

```
<a href="#">News <span class="badge">5</span></a><br>
<a href="#">Comments <span class="badge">10</span></a><br>
<a href="#">Updates <span class="badge">2</span></a>
```

Badge in Button

```
<button type="button" class="btn btn-primary">Primary <span class="badge">7</span></button>
```

Labels

Labels are used to provide additional information about something:

Use the `.label` class, followed by one of the six contextual classes `.label-default`, `.label-primary`, `.label-success`, `.label-info`, `.label-warning` or `.label-danger`, within a `<span>` element to create a label:

```
<h1>Example <span class="label label-default">New</span></h1>
<h2>Example <span class="label label-default">New</span></h2>
<h3>Example <span class="label label-default">New</span></h3>
<h4>Example <span class="label label-default">New</span></h4>
<h5>Example <span class="label label-default">New</span></h5>
<h6>Example <span class="label label-default">New</span></h6>
```

Read Bootstrap Badges and Labels online: <https://riptutorial.com/twitter-bootstrap/topic/7867/bootstrap-badges-and-labels>

Chapter 5: Bootstrap Components

Remarks

For more information, visit the official documentation located at <http://getbootstrap.com/javascript/>, where the component list is derived from.

Examples

Examples of Bootstrap Components

Bootstrap components are a collection of optional jQuery plugins which bundled with Bootstrap.

The purpose of Bootstrap components is to provide extended features and capabilities which would be difficult (or impossible) to accomplish without the use of Javascript. Some components provide are purely functional, whereas some components are used to define functionality for some of Bootstrap's special front-end widgets.

Examples include [transition effects](#), [modal dialogs](#), [dropdowns](#), [scrollspy](#), [tabs](#), [tooltips](#), [alerts](#), [popovers](#), [buttons](#), [collapse](#), [carousels](#), and [affixes](#).

Read Bootstrap Components online: <https://riptutorial.com/twitter-bootstrap/topic/6054/bootstrap-components>

Chapter 6: Bootstrap Containers

Introduction

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

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

Examples

Containers

`.container` has one fixed width for each screen size in bootstrap (xs,sm,md,lg);

`.container-fluid` expands to fill the available width.

```
@media (min-width: 568px) {  
  .container {  
    width: 550px;  
  }  
}  
  
@media (min-width: 992px) {  
  .container {  
    width: 970px;  
  }  
}  
  
@media (min-width: 1200px) {  
  .container {  
    width: 1170px;  
  }  
}
```

Depending on the width of the viewport that the webpage is being viewed on, the container class gives its div a specific fixed width.

Your `.container-fluid` element, on the other hand, will constantly resize as you make even the smallest changes to your browser width.

Read Bootstrap Containers online: <https://riptutorial.com/twitter-bootstrap/topic/10908/bootstrap-containers>

Chapter 7: Bootstrap Dropdowns

Parameters

Methods	Example
Call Via Javascript	<code>\$('.dropdown-toggle').dropdown();</code>
Toggles the dropdown	<code>\$('.dropdown-toggle').dropdown('toggle')</code>
Event Type	Description
show.bs.dropdown	This event fires immediately when the show instance method is called.
shown.bs.dropdown	This event is fired when the dropdown has been made visible to the user (will wait for CSS transitions, to complete).
hide.bs.dropdown	This event is fired immediately when the hide instance method has been called.
hidden.bs.dropdown	This event is fired when the dropdown has finished being hidden from the user (will wait for CSS transitions, to complete).
Event Handler Example	<code>\$(element).on('show.bs.dropdown', function () { // do something... })</code>

Remarks

When calling Dropdown Via Javascript `$('.dropdown-toggle').dropdown()`, the data-api i.e `data-toggle="dropdown"` still required. [Read More](#)

Examples

How to Use

Use `.dropdown` class on parent element of dropdown menu.

Add the `.dropdown-menu` class to a element to initialize the dropdown menu plugin.

Call the plugin by Using class `.dropdown-toggle` and the `data-toggle="dropdown"` attribute on a button or a Hyperlink.

Basic Example

```
<div class="dropdown">
  <button class="btn btn-primary dropdown-toggle" type="button" data-
toggle="dropdown">Dropdown Example
  <span class="caret"></span></button>
  <ul class="dropdown-menu">
    <li><a href="#">Option One</a></li>
    <li><a href="#">Option two</a></li>
    <li><a href="#">More Options</a></li>
  </ul>
</div>
```

Read Bootstrap Dropdowns online: <https://riptutorial.com/twitter-bootstrap/topic/6361/bootstrap-dropdowns>

Chapter 8: Bootstrap Navbar

Examples

Bootstrap Navbar

This is example of Bootstrap version 3 Navbar:

```
<nav class="navbar navbar-default" role="navigation">
<div class="container-fluid">
  <!-- Brand and toggle get grouped for better mobile display -->
  <div class="navbar-header">
    <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-ex1-collapse">
      <span class="sr-only">Toggle navigation</span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </button>
    <a class="navbar-brand" href="#">Title</a>
  </div>

  <!-- Collect the nav links, forms, and other content for toggling -->
  <div class="collapse navbar-collapse navbar-ex1-collapse">
    <ul class="nav navbar-nav">
      <li class="active"><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
    </ul>
    <form class="navbar-form navbar-left" role="search">
      <div class="form-group">
        <input type="text" class="form-control" placeholder="Search">
      </div>
      <button type="submit" class="btn btn-default">Submit</button>
    </form>
    <ul class="nav navbar-nav navbar-right">
      <li><a href="#">Link</a></li>
      <li class="dropdown">
        <a href="#" class="dropdown-toggle" data-toggle="dropdown">Dropdown <b
class="caret"></b></a>
        <ul class="dropdown-menu">
          <li><a href="#">Action</a></li>
          <li><a href="#">Another action</a></li>
          <li><a href="#">Something else here</a></li>
          <li><a href="#">Separated link</a></li>
        </ul>
      </li>
    </ul>
  </div><!-- /.navbar-collapse -->
</div>
```

Bootstrap Brand Image

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
```

```
<div class="navbar-header">
  <a class="navbar-brand" href="#">
    
  </a>
</div>
</div>
</nav>
```

Read Bootstrap Navbar online: <https://riptutorial.com/twitter-bootstrap/topic/6233/bootstrap-navbar>

Chapter 9: Bootstrap Themes

Examples

Bootstrap themes versus rule overrides

What are themes?

There are several visual appearance out there for Bootstrap, which can be found from sources, such as [Bootswatch](#), which are modifying the *bootstrap.min.css* file. You can also create your own theme this way.

When to modify themes and when to add new rules to a *site.css* file?

When to modify the

bootstrap.min.css

file, and when to add your own *.css* file, such as

site.css

?

Sometimes there are style requirements, which must be done, no matter what theme you are using. These rules should go into your own *.css* file, such as *site.css*, so the main theme can be changed, the rules from *site.css* will apply anyhow. In order to do that, you just have to link the bootstrap theme, and your own rules, to override the existing ones:

```
<link href="../../../Content/bootstrap.min.css" rel="stylesheet">
<link href="../../../Content/site.css" rel="stylesheet">
```

This way, Bootstrap themes can be changed anytime without losing the mandatory rules, applied from *site.css*.

Read Bootstrap Themes online: <https://riptutorial.com/twitter-bootstrap/topic/6381/bootstrap-themes>

Chapter 10: Bootstrap Validation

Remarks

- This validation technique can only be used on inputs that are within a form.
- Properties must have at least one validation requirement to show highlighting on a failed `onSubmit()` validation. Data types (other than string) have a hidden data type requirement, so do not require an explicit data annotation. Strings do not have this, so to force a validation check along with the other fields, add the data annotation `[MinLengthAttribute(0)]`.

Examples

Using ASP.NET MVC and Data Annotations

Add the following to Web.config (in Views folder), within `<appSettings>`:

```
<add key="ClientValidationEnabled" value="true"/>
<add key="UnobtrusiveJavaScriptEnabled" value="true"/>
```

Add the jqueryval bundle to BundleConfig.cs:

```
bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
    "~/Scripts/jqueryval/jquery.validate*"));
```

Add the following to all pages that need validation (or `_Layout.cshml`):

```
<!-- Reference to the jqueryval bundle -->
@Scripts.Render("~/bundles/jqueryval")

<!-- jQuery to apply bootstrap validation classes and glyphs to inputs -->
<script type="text/javascript">
    $.validator.defaults({
        highlight: function (element) {
            $(element).closest('.form-group').removeClass('has-success has-
feedback').addClass('has-error has-feedback'); // red highlighting
            $(element).closest('.form-group').find('.form-control-
feedback').removeClass('glyphicon-ok').addClass('glyphicon-remove'); // red cross glyphicon
        },
        unhighlight: function (element) {
            $(element).closest('.form-group').removeClass('has-error has-
feedback').addClass('has-success has-feedback'); // green highlighting
            $(element).closest('.form-group').find('.form-control-
feedback').removeClass('glyphicon-remove').addClass('glyphicon-ok'); // green tick glyphicon
        }
    });
</script>
```

Add data annotations to the relevant fields in the model:

```
using System.ComponentModel.DataAnnotations;

[Required(ErrorMessage = "This field is required.")]
```

In the view, add the following to each input that needs validating:

```
<!-- Validation messages -->
<div class="text-danger">@Html.ValidationMessageFor(m => m.SomeField)</div>

<!-- Bootstrap feedback span: -->
<span class="glyphicon form-control-feedback"></span>
```

Add the following to the relevant controller action to add server-side validation:

```
if (!ModelState.IsValid)
{
    return View(model);
}
else
{
    // continue with action
}
```

Example input that requires validation

Model:

```
[Required(ErrorMessage = "This field is required.")
[StringLength(maximumLength: 10, ErrorMessage = "This field must be 10 characters or less.")]
public string SomeRequiredField { get; set; }
```

View:

```
<div class="form-group has-feedback">
  <div class="col-md-4">
    @Html.LabelFor(m => m.SomeRequiredField, new { @class = "control-label" })
  </div>
  <div class="col-md-8">
    @Html.TextBoxFor(m => m.SomeRequiredField, new { @class = "form-control" })
    <div class="text-danger">@Html.ValidationMessageFor(m => m.SomeRequiredField)</div>
    <span class="glyphicon form-control-feedback"></span>
  </div>
</div>
```

Optional

Add the following jQuery to validate inputs on blur, as well as on submit:

```
$('input').on('blur', function () {
    $(this).valid();
});
```

```
});
```

Read Bootstrap Validation online: <https://riptutorial.com/twitter-bootstrap/topic/6388/bootstrap-validation>

Chapter 11: Buttons

Syntax

- Classes: `.btn-default` | `.btn-primary` | `.btn-success` | `.btn-info` | `.btn-warning` | `.btn-danger` | `.btn-link`;
- Sizes: `.btn-lg` | `.btn-md` | `.btn-sm` | `.btn-xs`;
- State: `active` | `disabled`.

Examples

Button Classes

Bootstrap provides multiple classes for styling buttons and making them stand out.

Bootstrap buttons can be created by adding the `.btn` class to an element.

Bootstrap Class	Role (color)
<code>.btn-default</code>	Standard button (white)
<code>.btn-primary</code>	Provides extra visual weight and identifies the primary action (blue)
<code>.btn-success</code>	Used to indicate a successful action (green)
<code>.btn-info</code>	Contextual button for providing information (light blue)
<code>.btn-warning</code>	Indicates caution should be applied by the user (yellow)
<code>.btn-danger</code>	Indicates a dangerous or negative action (red)
<code>.btn-link</code>	Make you button look like an anchor tag.

Button Sizes

You can also create different sizes of buttons with the `.btn-size` classes

Bootstrap Class	Result
<code>.btn-lg</code>	Creates a larger sized button
<code>.btn-sm</code>	Creates a smaller sized button
<code>.btn-xs</code>	Creates an extra-small button
<code>.btn-block</code>	Buttons become block-level elements and span the full width of their

Bootstrap Class	Result
	parent

Make button active

The `active` class will make a button appear pressed.

```
<button type="button" class="btn btn-primary active">Active Primary</button>
```

Disable a button

Adding the `disabled` class to a button will render the button un-clickable and show a forbidden cursor when hovering over it.

```
<button type="button" class="btn btn-primary disabled">Disabled Primary</button>
```

Render buttons horizontally together

Multiple buttons can be rendered horizontally with the `.btn-group` class. Simply wrap your buttons inside a container element and give that element the `btn-group` class.

```
<div class="btn-group">
  <button type="button" class="btn btn-primary">Apples</button>
  <button type="button" class="btn btn-primary">Oranges</button>
  <button type="button" class="btn btn-primary">Pineapples</button>
</div>
```

Render buttons vertically

Apply the `.btn-group-vertical` class to the container element

```
<div class="btn-group-vertical">
  <button type="button" class="btn btn-primary">Apples</button>
  <button type="button" class="btn btn-primary">Oranges</button>
  <button type="button" class="btn btn-primary">Pineapples</button>
</div>
```

Make button group take up full width

Buttons wrapped inside a `.btn-group` element only take up as much width as needed. To make the group span the entire width of the screen, use `.btn-group-justified` instead.

```
<div class="btn-group btn-group-justified">
  <a href="#" class="btn btn-primary">Apples</a>
  <a href="#" class="btn btn-primary">Oranges</a>
  <a href="#" class="btn btn-primary">Pineapples</a>
</div>
```

Read Buttons online: <https://riptutorial.com/twitter-bootstrap/topic/4757/buttons>

Chapter 12: Carousels

Remarks

For more information, visit the official documentation at <http://getbootstrap.com/javascript/#carousel>, where the basic HTML and Javascript usage examples and information are derived from.

It should be noted that carousels do not function correctly in IE 9 and earlier due to the use of CSS3 transitions/animations.

Examples

Basic HTML usage

A Bootstrap carousel is a Bootstrap component that creates a slideshow which cycles through elements within the carousel.

Here is a basic HTML usage example:

```
<div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#carousel-example-generic" data-slide-to="0" class="active"></li>
    <li data-target="#carousel-example-generic" data-slide-to="1"></li>
    <li data-target="#carousel-example-generic" data-slide-to="2"></li>
  </ol>

  <!-- Wrapper for slides -->
  <div class="carousel-inner" role="listbox">
    <div class="item active">
      
      <div class="carousel-caption">
        ...
      </div>
    </div>
    <div class="item">
      
      <div class="carousel-caption">
        ...
      </div>
    </div>
    ...
  </div>

  <!-- Controls -->
  <a class="left carousel-control" href="#carousel-example-generic" role="button" data-
slide="prev">
    <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="right carousel-control" href="#carousel-example-generic" role="button" data-
```

```
slide="next">
  <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
  <span class="sr-only">Next</span>
</a>
</div>
```

Basic Javascript usage and initialization

Carousel components can be instantiated via jQuery with the function

`$('.carousel').carousel(options)`, where `$('.carousel')` is a top-level reference to the specific carousel and `options` is a Javascript object specifying the carousel's default attributes.

The `options` object allows for multiple properties to be defined which will affect how the carousel behaves. These properties are defined as such:

- The `interval` property accepts a Javascript `number` type which allows a user to define the amount of time the carousel displays a given carousel slide for. If the boolean value `false` is specified, the carousel will not cycle automatically.
- The `pause` property accepts a Javascript `string` type which toggles behavior where the carousel's automatic cycling is paused when the user's mouse enters the carousel. The default (and only) value accepted is "hover".
- The `wrap` property accepts a Javascript `boolean` type which allows a user to define whether or not they want the carousel to continuously cycle without stopping on a given slide.
- The `keyboard` property accepts a Javascript `boolean` type which allows a user to define whether or not they want the carousel to respond to keyboard events.

Here is an example of the basic Javascript usage:

```
$('#carCarousel').carousel({ interval: 2500, pause: "hover", wrap: false, keyboard: true });
```

As with other Bootstrap components, the carousel's options can also be specified in HTML via data attributes.

Read Carousels online: <https://riptutorial.com/twitter-bootstrap/topic/1568/carousels>

Chapter 13: Columns

Examples

Responsive columns same height (CSS or SASS only)

You have to add a div with the class `.row-height` inside the row, and also add `.col-height` to the columns. If you want to restrict the effect to a certain media query, just use the responsive `.row-height` and `.col-height` classes: for example `.row-sm-height` with `.col-sm-height`.

CSS version:

```
.row-height {
  display: table;
  table-layout: fixed;
  height: 100%;
  width: calc(100% + 30px);
}
.col-height {
  display: table-cell;
  float: none;
  height: 100%;
}
.col-top {
  vertical-align: top;
}
.col-middle {
  vertical-align: middle;
}
.col-bottom {
  vertical-align: bottom;
}

@media (min-width: 480px) {
  .row-xs-height {
    display: table;
    table-layout: fixed;
    height: 100%;
    width: 100%;
  }
  .col-xs-height {
    display: table-cell;
    float: none;
    height: 100%;
  }
  .col-xs-top {
    vertical-align: top;
  }
  .col-xs-middle {
    vertical-align: middle;
  }
  .col-xs-bottom {
    vertical-align: bottom;
  }
}
```

```

@media (min-width: 768px) {
  .row-sm-height {
    display: table;
    table-layout: fixed;
    height: 100%;
    width: 100%;
  }
  .col-sm-height {
    display: table-cell;
    float: none;
    height: 100%;
  }
  .col-sm-top {
    vertical-align: top;
  }
  .col-sm-middle {
    vertical-align: middle;
  }
  .col-sm-bottom {
    vertical-align: bottom;
  }
}

@media (min-width: 992px) {
  .row-md-height {
    display: table;
    table-layout: fixed;
    height: 100%;
    width: calc(100% + 30px);
  }
  .col-md-height {
    display: table-cell;
    float: none;
    height: 100%;
  }
  .col-md-top {
    vertical-align: top;
  }
  .col-md-middle {
    vertical-align: middle;
  }
  .col-md-bottom {
    vertical-align: bottom;
  }
  .row-md-height .col-md-3 {
    width: 25%;
    min-width: 25%;
    max-width: 25%;
  }
}

@media (min-width: 1200px) {
  .row-lg-height {
    display: table;
    table-layout: fixed;
    height: 100%;
    width: 100%;
  }
  .col-lg-height {
    display: table-cell;

```

```

    float: none;
    height: 100%;
}
.col-lg-top {
    vertical-align: top;
}
.col-lg-middle {
    vertical-align: middle;
}
.col-lg-bottom {
    vertical-align: bottom;
}
}

```

SASS version (needed bootstrap _variables.scss):

```

@import "../bootstrap/variables.scss";
$sizes: xs sm md lg;
$screens: $screen-xs-min $screen-sm-min $screen-md-min $screen-lg-min;

//general
.row-height {
    display: table;
    table-layout: fixed;
    height: 100%;
    width: calc(100% + $grid-gutter-width);
}
.col-height {
    display: table-cell;
    float: none;
    height: 100%;
}
.col-top {
    vertical-align: top;
}
.col-middle {
    vertical-align: middle;
}
.col-bottom {
    vertical-align: bottom;
}

//different sizes
@for $i from 1 through length($sizes) {
    $size: nth($sizes, $i);
    $screen: nth($screens, $i);

    @media (min-width: #{$screen}) {
        .row-#{$size}-height {
            display: table;
            table-layout: fixed;
            height: 100%;
            width: 100%;
        }
        .col-#{$size}-height {
            display: table-cell;
            float: none;
            height: 100%;
        }
        .col-#{$size}-top {

```

```
        vertical-align: top;
    }
    .col-#{$size}-middle {
        vertical-align: middle;
    }
    .col-#{$size}-bottom {
        vertical-align: bottom;
    }
}

}
```

Read Columns online: <https://riptutorial.com/twitter-bootstrap/topic/6469/columns>

Chapter 14: Dropdowns

Remarks

For more information, visit the official Bootstrap documentation located at <http://getbootstrap.com/javascript/#dropdowns>, where the basic HTML usage example is derived from.

Examples

Basic HTML usage

A Bootstrap dropdown is a Bootstrap component that allows an HTML element trigger the display of a sub-menu dropdown upon the element being clicked.

Here is a basic HTML usage example:

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

Dropdown sub-menu items can be specified by inserting `li` elemented within the `ul` element with the `.dropdown-menu` class.

Read Dropdowns online: <https://riptutorial.com/twitter-bootstrap/topic/6033/dropdowns>

Chapter 15: Forms

Examples

Basic form

Form controls have some default styling without using any special classes.

However labels and controls can be wrapped in `.form-group` tags for optimum spacing.

```
<form>
  <div class="form-group">
    <label for="input-email">Email address</label>
    <input type="email" class="form-control" id="input-email" placeholder="Email">
  </div>
  <div class="form-group">
    <label for="input-password">Password</label>
    <input type="password" class="form-control" id="input-password" placeholder="Password">
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

Read-only and disabled inputs

Add the `readonly` attribute to prevent user input. A readonly field can't be edited

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

Add the `disabled` attribute to disable an input field. A disabled field can't be edited either. The cursor changes to make it more noticeable.

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

Read Forms online: <https://riptutorial.com/twitter-bootstrap/topic/6251/forms>

Chapter 16: Glyphicons

Remarks

This section provides an overview on Bootstrap glyphicons and describes how to use glyphicons.

Examples

How to Use Glyphicons

Twitter Bootstrap supports icons called glyphicons and they can be used with all tags of HTML.

All icons require a base class and individual icon class.

Keep in mind that icon classes cannot be directly combined with other components, so always use inner `<span></span>` tag.

If your HTML code has inner child elements then you are not able to use icon classes for that particular tag.

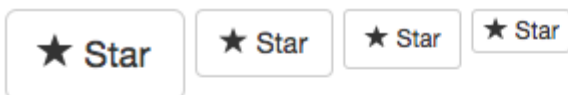
Examples

For example, you are creating a bootstrap button, then the syntax for this button should be like this:

```
<button type="button" class="btn btn-default btn-lg">
  Star
</button>
```

So in above example a simple bootstrap button is created but now you want to add a glyphicon in this button, for this simply add a `<span>` element inside a `<button>` tag. Like this:

```
<button type="button" class="btn btn-default btn-lg">
  <span class="glyphicon glyphicon-star" aria-hidden="true"></span>Star
</button>
```



```
<button type="button" class="btn btn-default" aria-label="Left Align">
  <span class="glyphicon glyphicon-align-left" aria-hidden="true"></span>
</button>

<button type="button" class="btn btn-default btn-lg">
  <span class="glyphicon glyphicon-star" aria-hidden="true"></span> Star
</button>
```

Read Glyphicons online: <https://riptutorial.com/twitter-bootstrap/topic/6098/glyphicons>

Chapter 17: Grid Nesting

Introduction

In Bootstrap it's possible to use grid columns *inside* other columns. This is helpful when creating advanced responsive layouts that utilize [multiple grid tiers](#).

Remarks

We can have as many number of columns as possible in the above mentioned way.

Examples

Nesting columns

```
<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>
```

The example is taken from [<http://getbootstrap.com/css/#grid-nesting>][1]

as the documentation of official website suggests

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

Read Grid Nesting online: <https://riptutorial.com/twitter-bootstrap/topic/9088/grid-nesting>

Chapter 18: Grid system

Introduction

Bootstrap's grid system consists of 12 units known as **Columns** (`.col-*-*` CSS classes) that are used to layout content *left-to-right* across the viewport. Columns are contained within **Rows** (`.row` CSS class) to create horizontal groups of columns. Rows are placed within a fixed or full-width **Container** (`.container` or `.container-fluid`, respectively) for proper alignment. Columns have padding that creates spacing (known as a "gutter") between the content in the columns.

Remarks

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 quickly creating page layouts through a series of rows and columns that house your content.

Examples

Media Queries

Media Queries in Bootstrap allow you to move, show and hide content based on the viewport size. The following media queries are used in LESS files to create the key breakpoints in the Bootstrap grid system:

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

Occasionally these are expanded 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) { ... }
```

Bootstrap Grid Tiers (Breakpoints)

In addition to the concept of [column units](#), Bootstrap has different **breakpoints** or grid sizes known as tiers. The Bootstrap 3 grid has four (4) tiers to accommodate different screen (or viewport) widths. The Bootstrap 3 tiers are `xs`, `sm`, `md`, and `lg`. Bootstrap's grid columns are identified by different `col-{breakpoint}-{units}` CSS classes.

Each grid tier **encompasses a range** that is designed to best-fit typical device screen widths such as that of desktops, laptops, tablets and smartphones.

Bootstrap uses CSS media queries to create responsive breakpoints that establish a boundary for each grid size. These grid sizes enable you to change the layout of columns to best match different screen widths and devices__ the essence of responsive design.

- `col-xs-*` — for the *smallest* screen widths like smartphones < 768 px
- `col-sm-*` — for *small* screen widths like smartphones and tablets >= 768 px
- `col-md-*` — for *medium* screen widths like tablets and laptops >= 992 px
- `col-lg-*` — for *large* screen widths like desktops >= 1200 px

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large de (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
Container width	None (auto)	750px	970px	1170px
Class prefix	<code>.col-xs-</code>	<code>.col-sm-</code>	<code>.col-md-</code>	<code>.col-lg-</code>
# 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			

Reference: [Grid System](#)

Same column width for each device

To create a column that is always 50% of the viewport width (on all devices) you could set `col-*-6` for every tier..

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

However, this is unnecessary extra markup, since `col-xs-6` means 6 units on `xs` and up. The smallest tier you set (xs, sm or md) also defines the size for larger screen widths. For the *same* size column on all tiers, just set the width for the smallest viewport.

Shorter code:

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

Different column width for each device (responsive design)

The `col-*-*` classes can be **combined** to control the column widths on different grid sizes..

For example, create a 50% width column at the `sm` tier, and a 25% width column at the `md` tier...

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

The `sm`, `md` and `lg` grids will all "stack" vertically at viewport widths less than 768 pixels. This is where the `xs` grid fits in. Columns that use the `col-xs-*` classes will not stack vertically and continue to scale down on the smallest screens.

Bootstrap Rows & Columns

Bootstrap's grid system has **12 units** known as **Columns** that can be used to layout content horizontally across the viewport.

The reason for a 12-unit grid (instead of 10, 16, etc..) is that 12 evenly divides into 6 (halves), 4 (quarters) and 3 (thirds). This makes adapting to a variety of layouts much easier. Bootstrap's grid columns are identified by different `col-{breakpoint}-{units}` CSS classes. [Learn more about viewport width and breakpoints \(A.K.A. Tiers\)](#)

So for example, `col-md-3` represents a column that takes up 3 of the 12 units (or 25%) across a medium (`md`) width viewport. To use a column width in your layout, simply use the appropriate `col-{breakpoint}-{units}` class in your HTML markup.

```
<div class="col-{breakpoint}-{units}">
```

Column width is fluid (not fixed width), so the columns consume a *percentage* of their container.

Column units in Bootstrap 3

- `col-*-1`: 1 of 12 (8.33333333% width)
- `col-*-2`: 2 of 12 (16.66666667% width)
- `col-*-3`: 3 of 12 (25% width)
- `col-*-4`: 4 of 12 (33.33333333% width)
- `col-*-5`: 5 of 12 (41.66666667% width)
- `col-*-6`: 6 of 12 (50% width)
- `col-*-7`: 7 of 12 (58.33333333% width)
- `col-*-8`: 8 of 12 (66.66666667% width)
- `col-*-9`: 9 of 12 (75% width)
- `col-*-10`: 10 of 12 (83.33333333% width)
- `col-*-11`: 11 of 12 (91.66666667% width)
- `col-*-12`: 12 of 12 (100% width)

[Demo - Bootstrap's 12 column units](#)

The Bootstrap Row

The Bootstrap `.row` class is used to contain the Columns. Columns should *always* be placed in Rows, and Rows should always be placed inside of a Container (`container` or `container-fluid`). The Row uses negative margins (-15px) to ensure proper spacing between the column's content and the edge of the browser. Rows are used to group columns horizontally.

```
<div class="container">
  <div class="row">
    <!-- one more columns -->
    <div class="col-{breakpoint}-{units}">..</div>
  </div>
</div>
```

Columns will fill the `.row` horizontally left-to-right, and will [wrap](#) to a new line every 12 column units. Therefore, you can use `.rows` to create **horizontal breaks**, or you can add **more than 12** column units in a single `.row` element to have **columns that wrap** (or stack) vertically down the viewport.

When using column wrapping (more than 12 units in a `.row`), you'll need to [use responsive resets \(or clearfixes\)](#) to ensure even wrapping of uneven column content. This is essential when the content of the columns varies in height.

More on Bootstrap Grid Columns & Rows

[Bootstrap 3 fluid grid layout issues?](#)

[Bootstrap 3 - nested row can I have columns add up to more then 12?](#)

[Bootstrap row and col explanation](#)

[How the Bootstrap Grid Works \(Medium\)](#)

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.

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

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

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

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

Note: Containers are not nestable (you cannot put a container inside another container), due to `padding` and more.

Offsetting columns

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

```
<div class="row">
  <div class="col-lg-4"></div>
  <div class="col-lg-4 col-lg-offset-4"></div>
</div>
<div class="row">
  <div class="col-lg-5 col-lg-offset-1"></div>
  <div class="col-lg-5 col-lg-offset-1"></div>
</div>
```

Column order manipulation using push and pull

```
<div class="container content">
  <div class="row">
    <!--Main Content-->
    <div class="col-lg-9 col-lg-push-3">
      Main Content
    </div>

    <!--Sidebar-->
    <div class="col-lg-3 col-lg-pull-9">
      Sidebar
    </div>
  </div>
</div>
```

This change the order of the built-in grid columns.

Syntax: `.col-md-push-*` and `.col-md-pull-*`.

More:

[Column order manipulation using col-lg-push and col-lg-pull in Twitter Bootstrap 3](#)

[Bootstrap 3: Push/pull columns only on smaller screen sizes](#)

[Column Ordering & Stacking in Bootstrap 3](#)

Read Grid system online: <https://riptutorial.com/twitter-bootstrap/topic/3330/grid-system>

Chapter 19: Jumbotron

Introduction

Jumbotron is a standard component of Bootstrap to display some important contents on your website. It is usually used right under the navbar, before the content.

Remarks

Inside the jumbotron, all grid system, container class and row class also works.

Examples

Basic jumbotron with two lines of text and a button

This is a jumbotron with a title, a content and a button.

Code

```
<div class="jumbotron">
  <h1>Title text</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec tortor ipsum, convallis sit.</p>
  <p><a class="btn btn-default" href="#" role="button">A button</a></p>
</div>
```

Result



Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Donec tortor ipsum, convallis sit.

A button

Read Jumbotron online: <https://riptutorial.com/twitter-bootstrap/topic/9188/jumbotron>

Chapter 20: List group

Remarks

You should know how to use bootstrap [Buttons](#) and little information about [Contextual](#) classes.

Examples

Basic example

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

Badges

```
<ul class="list-group">
  <li class="list-group-item">
    <span class="badge">14</span>
    Cras justo odio
  </li>
</ul>
```

Linked Items

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

Button items

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

Disabled Items

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

Contextual classes

```
<ul class="list-group">
  <li class="list-group-item list-group-item-success">Dapibus ac facilisis in</li>
  <li class="list-group-item list-group-item-info">Cras sit amet nibh libero</li>
  <li class="list-group-item list-group-item-warning">Porta ac consectetur ac</li>
  <li class="list-group-item list-group-item-danger">Vestibulum at eros</li>
</ul>
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-success">Dapibus ac facilisis in</a>
  <a href="#" class="list-group-item list-group-item-info">Cras sit amet nibh libero</a>
  <a href="#" class="list-group-item list-group-item-warning">Porta ac consectetur ac</a>
  <a href="#" class="list-group-item list-group-item-danger">Vestibulum at eros</a>
</div>
```

Custom content

```
<div class="list-group">
  <a href="#" class="list-group-item active">
    <h4 class="list-group-item-heading">List group item heading</h4>
    <p class="list-group-item-text">...</p>
  </a>
</div>
```

Read List group online: <https://riptutorial.com/twitter-bootstrap/topic/6347/list-group>

Chapter 21: Migrating to Bootstrap 4

Introduction

Bootstrap 4 is a [major rewrite](#) and there are many changes to be aware of when upgrading from Bootstrap 3. Here are the class name changes, tips and examples of migrating your Bootstrap 3.x code to Bootstrap 4.x.

Remarks

This just a small example more detailed examples to be followed.

Examples

Column layout changes of grid system in Bootstrap 4

The first code block is written in Bootstrap 3. In Bootstrap 3 there are 4 types of column specifications, namely `col-md-*` `col-lg-*` `col-sm-*` `col-xs-*`. A fully responsive layout will look like this in Bootstrap 3:

```
<div class="row">
  <div class="col-lg-4 col-md-8 col-sm-8 col-xs-8">
    contents
  </div>
  <div class="col-lg-4 col-md-4 col-sm-4 col-xs-4">
    contents
  </div>
</div>
```

In Bootstrap 4, they have added a new sm grid tier below 768px for more granular control. So Bootstrap 4 has `col-*` (xs), `col-sm-*`, `col-md-*`, `col-lg-*`, and `col-xl-*`. So what used to be `.col-md-6` in v3 is now `.col-lg-6` in v4. Notice that the `-xs` infix has been removed so `.col-6` represent 6 column units at the extra small (default) breakpoint.

So, if we now want to write the same above example in Bootstrap 4, it would look like this:

```
<div class="row">
  <div class="col-xl-8 col-lg-8 col-md-8 col-sm-8 col-8">
    contents
  </div>
  <div class="col-xl-8 col-lg-8 col-md-4 col-sm-4 col-4">
    contents
  </div>
</div>
```

Grid Layout Bootstrap 4

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints			
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	<code>.col-</code>	<code>.col-sm-</code>	<code>.col-md-</code>	<code>.col-lg-</code>	<code>.col-xl-</code>
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Offsets	Yes				
Column ordering	Yes				

Browser support changes

In twitter-bootstrap 4 the support for `IE8`, `IE9`, and `iOS 6` has been dropped. v4 is now only `IE10+` and `iOS 7+`. For sites needing either of those, use v3.

In twitter-bootstrap 4 the official support for `Android v5.0 Lollipop's Browser` and `WebView` has been Added. Earlier versions of the `Android Browser` and `WebView` remain only `unofficially supported`.

Affix class removal

Affix is removed from Bootstrap 4.

It is recommended to use a `position: sticky` polyfill instead.

If you were using Affix to apply additional, non-position styles, the polyfills might not support your use case. One option for such uses is the third-party `ScrollPos-Styler` library.

According to Bootstrap Documentation

Dropped the Affix jQuery plugin. We recommend using a `position: sticky` polyfill instead. See the `HTML5 Please` entry for details and specific polyfill recommendations.

If you were using Affix to apply additional, non-position styles, the polyfills might not support your use case. One option for such uses is the third-party `ScrollPos-Styler` library.

If somebody is migrating from Bootstrap v3 to Bootstrap v4 the fallback approach is given below--

HTML

```
<header>

</header>
<nav class="navbar navbar-light bg-faded" data-toggle="affix">
  <button class="navbar-toggler hidden-sm-up pull-xs-right" type="button" data-
toggle="collapse" data-target="#collapsingNavbar">
    ≡
  </button>
  <a class="navbar-brand" href="#">Brand</a>
  <div class="collapse navbar-toggleable-xs" id="collapsingNavbar">

    <ul class="nav navbar-nav pull-xs-right">
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#"
role="button" aria-haspopup="true" aria-expanded="false">
          Menu
        </a>
        <div class="dropdown-menu" aria-labelledby="Preview">
          <a class="dropdown-item" href="#">Logout</a>

        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item ">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item ">
        <a class="nav-link" href="#">Link</a>
      </li>
    </ul>
  </div>
</nav>

<div class="container" id="main">
  <h2>Hello Bootstrap 4.</h2>
  <div class="row">
    <div class="col-xs-12 col-sm-6 col-md-9">
      <p>3 wolf moon retro jean shorts chambray sustainable roof party. Shoreditch vegan
artisan Helvetica. Tattooed Codeply Echo Park Godard kogi, next level irony ennui twee squid
fap selvage. Meggings flannel Brooklyn literally small batch, mumblecore
PBR try-hard kale chips. Brooklyn vinyl lumbersexual bicycle rights, viral fap
cronut leggings squid chillwave pickled gentrify mustache. 3 wolf moon hashtag church-key Odd
Future. Austin messenger bag normcore, Helvetica Williamsburg
sartorial tote bag distillery Portland before they sold out gastropub
taxidermy Vice.</p>
    </div>
    <div class="col-xs-6 col-md-3">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis pharetra codeply
varius quam sit amet vulputate. Quisque mauris augue, molestie tincidunt codeply condimentum
vitae, gravida a libero. Aenean sit amet felis dolor, in sagittis nisi.
Sed ac orci quis tortor imperdiet venenatis. Duis elementum auctor accumsan.
Aliquam in felis sit amet augue.</p>
    </div>
  </div>
</div>
```



```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis pharetra codeply  
varius quam sit amet vulputate. Quisque mauris augue, molestie tincidunt codeply condimentum  
vitae, gravida a libero. Aenean sit amet felis dolor, in sagittis nisi.
```

```
    Sed ac orci quis tortor imperdiet venenatis. Duis elementum auctor accumsan.  
    Aliquam in felis sit amet augue.
```

```
    </p>  
  </div>  
</div>  
<div class="row">  
  <div class="col-xs-6 col-sm-4">  
    <div class="card card-outline-primary">  
      <div class="card-block">  
        <h3 class="card-title">Card</h3>  
        <p class="card-text">With supporting text below as a natural lead-in to  
additional content.</p>  
        <a href="#" class="btn btn-outline-secondary">Outline</a>  
      </div>  
    </div>  
  </div>  
  <div class="col-xs-6 col-sm-4">  
    <div class="card card-outline-primary">  
      <div class="card-block">  
        <h3 class="card-title">Card</h3>  
        <p class="card-text">With supporting text below as a natural lead-in to  
additional content.</p>  
        <a href="#" class="btn btn-outline-secondary">Outline</a>  
      </div>  
    </div>  
  </div>  
  <div class="col-xs-6 col-sm-4">  
    <div class="card card-outline-primary">  
      <div class="card-block">  
        <h3 class="card-title">Card</h3>  
        <p class="card-text">With supporting text below as a natural lead-in to  
additional content.</p>  
        <a href="#" class="btn btn-outline-secondary">Outline</a>  
      </div>  
    </div>  
  </div>  
</div>  
</div>
```

CSS

```
header {  
  height: 220px;  
  background: #ccc;  
}
```

JAVASCRIPT

```
$(document).ready(function() {  
  
  var toggleAffix = function(affixElement, scrollElement, wrapper) {  
  
    var height = affixElement.outerHeight(),  
        top = wrapper.offset().top;  
  
    if (scrollElement.scrollTop() >= top){
```

```

        wrapper.height(height);
        affixElement.addClass("affix");
    }
    else {
        affixElement.removeClass("affix");
        wrapper.height('auto');
    }
};

$('[data-toggle="affix"]').each(function() {
    var ele = $(this),
        wrapper = $('<div></div>');

    ele.before(wrapper);
    $(window).on('scroll resize', function() {
        toggleAffix(ele, $(this), wrapper);
    });

    // init
    toggleAffix(ele, $(window), wrapper);
});
});

```

Bootstrap 4 Navbar

The new Bootstrap 4 Navbar Component is improved over it's Bootstrap 3.x predecessor. In Bootstrap 4, the Navbar is **responsive by default** and utilizes **flexbox** to make alignment of Navbar content much easier. It's also a simple matter of using the new `navbar-toggleable-*` classes to change the Navbar breakpoint. Now the Navbar has 6 breakpoint sizes or “states” so that you can easily have one of the following Navbar options.

- The Navbar *never* collapses into the vertical mobile view, and is always horizontal.
- The Navbar is *always* collapsed into the vertical view, and toggled via the hamburger.
- The Navbar collapses into vertical view at *one of the 4 responsive breakpoints*.

Basic Bootstrap 4 Navbar

```

<nav class="navbar navbar-toggleable-md">
  <button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse"
data-target="#navbar1">
    <span class="navbar-toggler-icon"></span>
  </button>
  <a class="navbar-brand" href="#">Navbar</a>
  <div class="collapse navbar-collapse" id="navbar1">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
    </ul>
  </div>

```

```
</nav>
```

As you can see from the code above the `navbar-header` class has been removed from Bootstrap 4, and the `container-fluid` is no longer required for a full width Navbar.

Changing the Navbar Breakpoint

The `navbar-toggleable-md` class makes the above Navbar collapse vertically (and show the toggler icon) at the medium (md) breakpoint of 992px. To change this to a different breakpoint, we'd just need to swap out `navbar-toggleable-md` with one of these..

- `navbar-toggleable` = collapse on `xs` widths <576px
- `navbar-toggleable-sm` = collapse on `sm` widths <768px
- `navbar-toggleable-lg` = collapse on `lg` widths <1200px

Bootstrap 4 Breakpoint Navbar Demo

Changing the Navbar Alignment

Flexbox enables us to easily change the alignment of the Navbar and its content (brand, links, forms or text). The default Navbar content is left aligned. Of course there are many other alignment scenarios...

- Brand left (default), links center & right
- Brand center, links left & right
- Brand left and links right
- Brand, links and fill width form input
- No brand, links center & right
- Brand left, links right inside container
- Justified links (fill width) centered

Bootstrap 4 Navbar with Centered Brand, and Left/Right Links

```
<nav class="navbar navbar-toggleable-sm navbar-inverse bg-primary">
  <button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse"
data-target=".dual-collapse">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="navbar-collapse collapse dual-collapse">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#features">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
    </ul>
  </div>
</nav>
```

```

        <li class="nav-item">
            <a class="nav-link" href="#">Link</a>
        </li>
    </ul>
</div>
<a class="navbar-brand d-flex mx-auto" href="#">Navbar 2</a>
<div class="navbar-collapse collapse dual-collapse">
    <ul class="navbar-nav ml-auto">
        <li class="nav-item">
            <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="#">Link</a>
        </li>
    </ul>
</div>
</nav>

```

Bootstrap 4 Navbar with Brand Left, Links Center and Right

```

<nav class="navbar navbar-light navbar-togglerable-sm bg-faded justify-content-center">
    <button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse"
data-target="#collapsingNavbar3">
        <span class="navbar-toggler-icon"></span>
    </button>
    <a href="/" class="navbar-brand d-flex w-50 mr-auto">Brand</a>
    <div class="navbar-collapse collapse" id="collapsingNavbar3">
        <ul class="navbar-nav mx-auto w-100 justify-content-center">
            <li class="nav-item active">
                <a class="nav-link" href="#">Link</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Link</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Link</a>
            </li>
        </ul>
        <ul class="nav navbar-nav ml-auto w-100 justify-content-end">
            <li class="nav-item">
                <a class="nav-link" href="#">Right</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Right</a>
            </li>
        </ul>
    </div>
</nav>

```

Navbar Alignment Demos: <http://www.codeply.com/go/qhaBrcWp3v>

More on the Bootstrap 4 Navbar

[Customizing Color, Alignment or Height](#)

Bootstrap 3 to Bootstrap 4 CSS Changes

Since Bootstrap 4 is a **major** rewrite, many of the Bootstrap 3.x class names have changed or been removed. The restructuring of components such as the Navbar, and the introduction of new CSS classes and Flexbox support means that upgrading to 4.x is *not* a simple conversion process from 3.x.

However, there are some Bootstrap 3.x CSS classes that have a specific Bootstrap 4 replacement.

CSS class name/selector changes from Bootstrap 3.3.7 to 4 (alpha 6)

{t} - represents a **tier** or breakpoint (ie: sm,md,lg,etc..).

xs tier is the default, and doesn't need to be specified: col-3,col-6,etc..

{u} - represents a **col unit** size (ie: 1-12)

Bootstrap 3.x	Bootstrap 4
.col-{t}-{u}	.col-{t}-{u} (leave {t} blank for xs)
.col-{t}-offset-{u}	.offset-{t}-{u} (leave {t} blank for xs)
.col-{t}-push-{u}	.push-{t}-{u} (leave {t} blank for xs)
.col-{t}-pull-{u}	.pull-{t}-{u} (leave {t} blank for xs)
.panel	.card
.panel-heading	.card-header
.panel-title	.card-title
.panel-body	.card-block
.panel-footer	.card-footer
.panel-primary	.card-primary.card-inverse
.panel-success	.card-success.card-inverse
.panel-info	.card-info.card-inverse
.panel-warning	.card-warning.card-inverse
.panel-danger	.card-danger.card-inverse
.well	.card.card-block
.thumbnail	.card.card-block
.list-inline > li	.list-inline-item

Bootstrap 3.x	Bootstrap 4
.dropdown-menu > li	.dropdown-item
.nav navbar > li	.nav-item
.nav navbar > li > a	.nav-link
.navbar-right	.ml-auto
.navbar-btn	.nav-item
.navbar-fixed-top	.fixed-top
.nav-stacked	.flex-column
.btn-default	.btn-secondary
.img-responsive	.img-fluid
.img-circle	.rounded-circle
.img-rounded	.rounded
.form-horizontal	(removed)
.radio	.form-check
.checkbox	.form-check
.input-lg	.form-control-lg
.input-sm	.form-control-sm
.control-label	.form-control-label
.table-condensed	.table-sm
.pagination > li	.page-item
.pagination > li > a	.page-link
.item	.carousel-item
.text-help	.form-control-feedback
.pull-right	.float-right
.pull-left	.float-left
.center-block	.mx-auto

Bootstrap 3.x	Bootstrap 4
.collapse.in	.collapse.show
.hidden-sm	.hidden-md-down
.hidden-md	.hidden-lg-down
.hidden-xs	.hidden-xs-down
.visible-xs	.hidden-sm-up
.visible-sm	.hidden-xs-down.hidden-md-up
.visible-md	.hidden-sm-down.hidden-lg-up
.visible-lg	.hidden-md-down.hidden-xl-up
.label	.badge
.badge	.badge.badge-pill

Also see:

[Bootstrap 3.x to 4 Migration Tool](#)

[What's New in Bootstrap 4](#)

Bootstrap 4 Vertical Align

Getting elements to **center** or **bottom** align vertically has always been a challenge with CSS and Bootstrap. The desired vertical alignment may be within a parent container, or relative to adjacent elements.

Now that Bootstrap 4 is **flexbox by default** there are many different approaches to vertical alignment using: [Auto-margins](#), [Flexbox Utilities](#), or the [Display Utilities](#) along with [Vertical Align Utilities](#).

At first, the [Vertical Alignment Utilities](#) would seem an obvious choice, but these *only* work with inline and table display elements. Here are some Bootstrap 4 vertical alignment options and scenarios...

1 - Vertical Center Using Auto Margins:

One way to vertically center is to use `my-auto`. This will center the element within it's container. For example, `h-100` makes the row full height, and `my-auto` will vertically center the `col-sm-12` column.

```
<div class="row h-100">
  <div class="col-sm-12 my-auto">
    <div class="card card-block w-25">Card</div>
  </div>
```

```
</div>
```

Vertical Center Using Auto Margins Demo

`my-auto` represents margins on the vertical y-axis and is equivalent to:

```
margin-top: auto;  
margin-bottom: auto;
```

2 - Vertical Center with Flexbox:

Center

Taller

With supporting text

Outline

Since Bootstrap 4 `.row` is now `display: flex` you can simply use `align-self-center` on any column to vertically center it...

```
<div class="row">  
  <div class="col-6 align-self-center">  
    <div class="card card-block">  
      Center  
    </div>  
  </div>  
  <div class="col-6">  
    <div class="card card-inverse card-danger">  
      Taller  
    </div>  
  </div>  
</div>
```

or, use `align-items-center` on the entire `.row` to vertically center align all `col-*` in the row...

```
<div class="row align-items-center">  
  <div class="col-6">  
    <div class="card card-block">  
      Center  
    </div>  
  </div>  
  <div class="col-6">  
    <div class="card card-inverse card-danger">  
      Taller  
    </div>  
  </div>  
</div>
```


3 - Vertical Center Using Display Utils:

Bootstrap 4 has [display utils](#) that can be used for `display:table`, `display:table-cell`, `display:inline`, etc.. These can be used with the [vertical alignment utils](#) to align inline, inline-block or table cell elements.

```
<div class="row h-50">
  <div class="col-sm-12 h-100 d-table">
    <div class="card card-block d-table-cell align-middle">
      I am centered vertically
    </div>
  </div>
</div>
```

Vertical Center Using Display Utils Demo

Bootstrap 4 Centering

How to center an element, column, or content inside a column works differently in Bootstrap 4.

Horizontal Center

- `text-center` is still used for `display:inline` elements
- `mx-auto` replaces `center-block` to center `display:block` elements
- `offset-*` *or* `mx-auto` can be used to center grid columns

`mx-auto` (auto x-axis margins) will center `display:block` or `display:flex` elements that have a *defined width*, (`%`, `vw`, `px`, etc..). **Flexbox is used by default** on grid columns, so there are also various flexbox centering methods.

Center text or inline elements: `text-center`

```
<div class="container">
  <h1 class="text-center">i'm centered</h1>
  <div class="row">
    <div class="col text-center">i'm centered!</div>
  </div>
</div>
```

Center `display:block` OR `display:flex`: `mx-auto`

```
<div class="row">
  <div class="col-12">
    
  </div>
</div>
```

Center columns using offsets: `offset-*`

```
<div class="row">
  <div class="col-4 offset-4">
    <h6>I'm .col-4 centered (offset 4)
  </div>
</div>
```

Columns can *also* be centered with: `mx-auto`

```
<div class="row">
  <div class="col-4 mx-auto">
    <h6>I'm .col-4 centered</h6>
  </div>
</div>
```

Demo Bootstrap 4 Horizontal Centering

Vertical Center

For vertical centering in Bootstrap 4 (y-axis), see the docs on: [Bootstrap 4 Vertical Align](#)

Bootstrap 4 Column Order

Changing the order (or position) was possible in Bootstrap 3 using the push pull classes. In Bootstrap 4, the **push pull** classes still work, and additionally **flexbox order** can be used.

In Bootstrap 4, the **push pull** classes are now `push-{viewport}-{units}` and `pull-{viewport}-{units}` and the `xs-` infix has been removed. Consider this example that changes the column order to 1-3-2 layout on `xs` and `sm`:

```
<div class="row">
  <div class="col-3 col-md-6">
    1
  </div>
  <div class="col-3 col-md-6 push-6 push-md-0">
    2
  </div>
  <div class="col-6 col-md-12 pull-3 pull-md-0">
    3
  </div>
</div>
```

Bootstrap 4 Push Pull Demo

Since the new version 4 is flexbox, **another option** is to use the **flexbox utility** classes to change the order of columns. Now full width, 12 unit `col-*-12` columns can be reversed using **flexbox** ordering.

```
<div class="row">
  <div class="col-md-12">
    Col 1
  </div>
  <div class="col-md-12 flex-first flex-md-unordered">
    Col 2
  </div>
</div>
```

Flexbox Ordering Demo

Read Migrating to Bootstrap 4 online: <https://riptutorial.com/twitter-bootstrap/topic/9090/migrating-to-bootstrap-4>

Chapter 22: Modal Dialogs

Remarks

For more information, visit the official documentation at <http://getbootstrap.com/javascript/#modals>, where the 'Basic HTML Usage' example was derived from.

Examples

Basic HTML usage

A Bootstrap modal dialog is a Bootstrap component which creates a modal dialog window which floats over page-level content.

Here is an example of the basic usage of a Bootstrap modal dialog in HTML:

```
<div class="modal fade" tabindex="-1" role="dialog">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
        <h4 class="modal-title">Modal title</h4>
      </div>
      <div class="modal-body">
        <p>One fine body&hellip;</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div><!-- /.modal-content -->
  </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
```

Basic Javascript usage and initialization

Modal dialog components can be instantiated via jQuery with the function

`$('#myModal').modal(options)`, where `$('#myModal')` is a top-level reference to the specific modal dialog and `options` is a Javascript object specifying the modal dialog's default attributes.

The `options` object allows for multiple properties to be defined which will affect how the modal dialog behaves. These properties are defined as such:

- The `backdrop` property allows a user to define whether or not they want a grey background overlay to appear behind the modal dialog. Both boolean values and the string "static" are recognized. If "static" is specified, the modal dialog will not be closed when a user clicks on the background overlay.
- The `keyboard` property allows a user to define whether or not they want the modal dialog to

be closed when the escape key is pressed on the keyboard.

- The `show` property allows a user to define whether or not they want the modal dialog to appear when the modal is initialized.

Here is an example of the basic Javascript usage:

```
$('#carModal').modal({ backdrop: false, keyboard: true, show: false });
```

As with other Bootstrap components, the modal's options can also be specified in HTML via data attributes.

Read Modal Dialogs online: <https://riptutorial.com/twitter-bootstrap/topic/5927/modal-dialogs>

Chapter 23: Modals

Remarks

Modals require bootstrap.min.js to function properly.

More details can be found here: <http://getbootstrap.com/javascript/#modals>

Examples

Basic HTML Modal

A modal is a dialog window which can be displayed over the current page.

```
<!-- Clicking the button will open the modal window -->
<button type="button" class="btn btn-success btn-lg" data-toggle="modal" data-
target="#theModal">Open The Modal</button>

<!-- The Modal -->
<div id="theModal" class="modal fade" role="dialog">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal">&times;</button>
        <h4 class="modal-title">Text For The Modal Header</h4>
      </div>
      <div class="modal-body">
        <p>Text for The Modal Body.</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
      </div>
    </div>
  </div>
</div>
</div>
```

Read Modals online: <https://riptutorial.com/twitter-bootstrap/topic/6320/modals>

Chapter 24: Navbar

Examples

Basic Navbar (fixed at the top of page)

```
<div class="navbar navbar-inverse navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <!-- vvv Hamburger icon that gets shown when window reaches a certain scale vvv -->
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <!-- ^^^ Hamburger icon that gets shown when window reaches a certain scale ^^^ -->
      <a class="navbar-brand" href="#">WebSite Title</a>
    </div>
    <div class="navbar-collapse collapse">
      <ul class="nav navbar-nav">
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a asp-controller="Home" asp-action="Contact">Contact</a></li>
      </ul>
    </div>
  </div>
</div>
```

Submenu in navbar

```
<div class="navbar navbar-inverse navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">WebSite Title</a>
    </div>
    <div class="navbar-collapse collapse">
      <ul class="nav navbar-nav">
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a asp-controller="Home" asp-action="Contact">Contact</a></li>
        <!-- vvv Create a submenu in the navbar vvv -->
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown">Testing Stuff
          <b class="caret"></b></a>
```

```

        <ul class="dropdown-menu">
            <li><a href="#">SubItem</a></li>
            <li><a href="#">Something Sub-y</a></li>
        </ul>
    </li>
    <!-- ^^^ Create a submenu in the navbar ^^^ --->
</ul>
</div>
</div>
</div>

```

Navbar divider

```

<div class="navbar navbar-inverse navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">WebSite Title</a>
    </div>
    <div class="navbar-collapse collapse">
      <ul class="nav navbar-nav">
        <li><a href="#">Home</a></li>
        <!-- vvv Create a divider in the nav vvv --->
        <li class="divider"></li>
        <!-- ^^^ Create a divider in the nav ^^^ --->
        <li><a href="#">About</a></li>
        <li><a asp-controller="Home" asp-action="Contact">Contact</a></li>
      </ul>
    </div>
  </div>
</div>

```

Keep current navigation link "active"

```

// Add active class to active navigation link
$(document).ready(function () {
    $('ul.nav.navbar-nav').find('a[href="' + location.pathname + '"]')
        .closest('li').addClass('active');
});

```

Change Navbar breakpoint (mobile vs normal)

max-width is the breakpoint

```

@media (max-width: 1200px) {
    .navbar-header {
        float: none;
    }
    .navbar-left,.navbar-right {

```



```

        float: none !important;
    }
    .navbar-toggle {
        display: block;
    }
    .navbar-collapse {
        border-top: 1px solid transparent;
        box-shadow: inset 0 1px 0 rgba(255,255,255,0.1);
    }
    .navbar-fixed-top {
        top: 0;
        border-width: 0 0 1px;
    }
    .navbar-collapse.collapse {
        display: none !important;
    }
    .navbar-nav {
        float: none !important;
        margin-top: 7.5px;
    }
    .navbar-nav>li {
        float: none;
    }
    .navbar-nav>li>a {
        padding-top: 10px;
        padding-bottom: 10px;
    }
    .collapse.in{
        display: block !important;
    }
    .navbar-nav .open .dropdown-menu {
        position: static;
        float: none;
        width: auto;
        margin-top: 0;
        background-color: transparent;
        border: 0;
        -webkit-box-shadow: none;
        box-shadow: none;
    }
}

```

Close collapsed navbar when clicking outside of the navbar

```

jQuery('body').bind('click', function(e) {
    if(jQuery(e.target).closest('#navbar').length == 0) {
        // click happened outside of .navbar, so hide
        var opened = jQuery('.navbar-collapse').hasClass('collapse in');
        if ( opened === true ) {
            jQuery('#navbar2 .navbar-collapse').collapse('hide');
        }
    }
});

```

Read Navbar online: <https://riptutorial.com/twitter-bootstrap/topic/2267/navbar>

Chapter 25: Navigation Menus

Examples

Horizontal Pill Menu

```
<ul class="nav nav-pills">
  <li role="presentation" class="active"><a href="#">Home</a></li>
  <li role="presentation"><a href="#">Profile</a></li>
  <li role="presentation"><a href="#">Messages</a></li>
</ul>
```

Vertical Pill Menu

```
<ul class="nav nav-pills nav-stacked">
  <li role="presentation" class="active"><a href="#">Home</a></li>
  <li role="presentation"><a href="#">Profile</a></li>
  <li role="presentation"><a href="#">Messages</a></li>
</ul>
```

Full Width Responsive Horizontal Pill

```
<ul class="nav nav-tabs nav-justified">
  <li role="presentation" class="active"><a href="#">Home</a></li>
  <li role="presentation"><a href="#">Profile</a></li>
  <li role="presentation"><a href="#">Messages</a></li>
</ul>
```

Read Navigation Menus online: <https://riptutorial.com/twitter-bootstrap/topic/6404/navigation-menus>

Chapter 26: Navs

Examples

Bootstrap Navs

Navs available in Bootstrap have shared markup, starting with the base `.nav` class, as well as shared states. Swap modifier classes to switch between each style.

Tabs

```
<ul class="nav nav-tabs">
  <li role="presentation" class="active"><a href="#">Home</a></li>
  <li role="presentation"><a href="#">Profile</a></li>
  <li role="presentation"><a href="#">Messages</a></li>
</ul>
```

Pills

```
<ul class="nav nav-pills">
  <li role="presentation" class="active"><a href="#">Home</a></li>
  <li role="presentation"><a href="#">Profile</a></li>
  <li role="presentation"><a href="#">Messages</a></li>
</ul>
```

Justified

```
<ul class="nav nav-tabs nav-justified">
  ...
</ul>
<ul class="nav nav-pills nav-justified">
  ...
</ul>
```

With Dropdowns

```
<ul class="nav nav-tabs">
  <li role="presentation" class="dropdown">
    <a class="dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-
haspopup="true" aria-expanded="false">
      Dropdown <span class="caret"></span>
    </a>
    <ul class="dropdown-menu">
      ...
    </ul>
  </li>
</ul>
```

Read Navs online: <https://riptutorial.com/twitter-bootstrap/topic/6505/navs>

Chapter 27: Pagination

Introduction

Pagination links indicate a series of related content exists across multiple pages. Typically these are used where a multi-page approach to long lists of content improves general performance, such as in search results or inboxes.

Examples

A simple Pagination example

```
<nav aria-label="Page navigation example">
  <ul class="pagination">
    <li class="page-item"><a class="page-link" href="#">Previous</a></li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item"><a class="page-link" href="#">Next</a></li>
  </ul>
</nav>
```

which gives you:



Read Pagination online: <https://riptutorial.com/twitter-bootstrap/topic/10605/pagination>

Chapter 28: Panels

Remarks

The panel component in bootstrap is a (bordered) box with some padding around its content, and optionally heading and footer containers.

Examples

Basic example

By default, all the `.panel` does is apply some basic border and padding to contain some content.

```
<div class="panel panel-default">
  <div class="panel-body">
    Basic panel example
  </div>
</div>
```

Panel with heading

Easily add a heading container to your panel with `.panel-heading`. You may also include any `<h1>`–`<h6>` with a `.panel-title` class to add a pre-styled heading. However, the font sizes of `<h1>`–`<h6>` are overridden by `.panel-heading`.

For proper link coloring, be sure to place links in headings within `.panel-title`.

```
<div class="panel panel-default">
  <div class="panel-heading">Panel heading without title</div>
  <div class="panel-body">
    Panel content
  </div>
</div>

<div class="panel panel-default">
  <div class="panel-heading">
    <h3 class="panel-title">Panel title</h3>
  </div>
  <div class="panel-body">
    Panel content
  </div>
</div>
```

Panel with footer

Wrap buttons or secondary text in `.panel-footer`. Note that panel footers **do not** inherit colors and borders when using contextual variations as they are not meant to be in the foreground.

```
<div class="panel panel-default">
  <div class="panel-body">
    Panel content
  </div>
  <div class="panel-footer">Panel footer</div>
</div>
```

Read Panels online: <https://riptutorial.com/twitter-bootstrap/topic/2848/panels>

Chapter 29: Printing in Bootstrap.

Examples

Basic HTML usage

The print elements of Bootstrap allow you to designate which items should be visible when printed and which should be hidden.

To make something visible use either of the following depending on the element and how it should appear when printed:

```
.visible-print-block  
.visible-print-inline  
.visible-print-inline-block
```

To hide something from being printed, use the following:

```
.hidden-print
```

Read **Printing in Bootstrap**. online: <https://riptutorial.com/twitter-bootstrap/topic/6707/printing-in-bootstrap>

Chapter 30: Tables

Examples

Simple Table

While styling effects can vary depending on the theme, the `.table` class is used to create a uniform and consistent appearance for tables across an application:

```
<table class="table">
  <tr>
    <th>Season</th>
    <th>Doctor</th>
    <th>Companion</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Christopher Eccleston</td>
    <td>Rose Tyler</td>
  </tr>
</table>
```

Read Tables online: <https://riptutorial.com/twitter-bootstrap/topic/6299/tables>

Chapter 31: Tables

Remarks

Content order and complex tables Beware that the `table-reflow` style changes the visual order of content. Make sure that you only apply this style to **well-formed** and simple data tables (and in particular, don't use this for layout tables) with appropriate table header cells for each row and column.

In addition, this class will not work correctly for tables with cells that span multiple rows or columns (using `rowspan` or `colspan` attributes).

Examples

Basic table

Bootstrap defines a custom styling for table using the `.table` class. Just add the `.table` class to any `<table>` to see horizontal dividers and padding:

```
<table class="table">
  <thead><tr><th>First Name</th><th>Last name</th></tr></thead>
  <tbody>
    <tr><td>John</td><td>Doe</td></tr>
    <tr><td>Fred</td><td>Bloggs</td></tr>
  </tbody>
</table>
```

Table with advanced styling

Bootstrap provides a couple of classes for advanced table styling.

Striped rows

You will have a table with striped rows, if you add `.table-striped` class:

```
<table class="table table-striped">
  <thead><tr><th>First Name</th><th>Last name</th></tr></thead>
  <tbody>
    <tr><td>John</td><td>Doe</td></tr>
    <tr><td>Fred</td><td>Bloggs</td></tr>
  </tbody>
</table>
```

Note that:

Striped tables are styled via the `:nth-child` CSS selector, which is not available in

Bordered table

You will have a table with borders on all sides of the table and cells, if you add `.table-bordered` class:

```
<table class="table table-bordered">
  <thead><tr><th>First Name</th><th>Last name</th></tr></thead>
  <tbody>
    <tr><td>John</td><td>Doe</td></tr>
    <tr><td>Fred</td><td>Bloggs</td></tr>
  </tbody>
</table>
```

Hover on rows

If you add `.table-hover` class, you will have a table with highlighted rows when the user hovers over a row:

```
<table class="table table-hover">
  <thead><tr><th>First Name</th><th>Last name</th></tr></thead>
  <tbody>
    <tr><td>John</td><td>Doe</td></tr>
    <tr><td>Fred</td><td>Bloggs</td></tr>
  </tbody>
</table>
```

Condensed table

If you add `.table-condensed` class, the default cell padding will be cut in half, so you will have a more compact table:

```
<table class="table table-condensed">
  <thead><tr><th>First Name</th><th>Last name</th></tr></thead>
  <tbody>
    <tr><td>John</td><td>Doe</td></tr>
    <tr><td>Fred</td><td>Bloggs</td></tr>
  </tbody>
</table>
```

Contextual classes

Bootstrap tables support contextual colors. To change background color of a table row or cell you just have to add one of the following contextual classes: `.active`, `.success`, `.info`, `.warning`, `.danger`

```
<table class="table">
  <thead><tr><th>First Name</th><th>Last name</th></tr></thead>
  <tbody>
    <tr class="success"><td>John</td><td>Doe</td></tr>
    <tr><td>Fred</td><td class="info">Bloggs</td></tr>
  </tbody>
</table>
```

Responsive tables

You have to wrap any `.table` in html container with `.table-responsive` class to create responsive tables:

```
<div class="table-responsive">
  <table class="table">
    <thead><tr><th>First Name</th><th>Last name</th></tr></thead>
    <tbody>
      <tr><td>John</td><td>Doe</td></tr>
      <tr><td>Fred</td><td>Bloggs</td></tr>
    </tbody>
  </table>
</div>
```

Responsive tables will scroll horizontally on small devices (<768px). There will be no differences for screens larger than 768px wide.

Table Reflow - Vertical headers

Getting a table with vertical headers.

Twitter bootstrap now support vertical header on a well formatted normal table. To achieve this just use `.table-reflow` class

Use twitter bootstrap `.table-reflow` class on a well formed table to achieve a table with vertical headers. Additionally you can combine with using `.table-striped` and `.table-hover` for hovering on columns this time.

```
<table class="table table-striped table-hover table-reflow">
  <thead>
    <tr>
      <th><strong> First Name: </strong></th>
      <th><strong> Last Name: </strong></th>
      <th><strong> Email: </strong></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td> John </td>
      <td> Doe </td>
      <td> john.doe@email.com </td>
    </tr>
    <tr>
      <td> Joane </td>
      <td> Donald </td>
```

```
        <td> jane@email.com </td>
    </tr>
</tbody>
</table>
```

You should check the v4 alpha docs here: [twitter-bootstrap .table-reflow](#)

Read Tables online: <https://riptutorial.com/twitter-bootstrap/topic/6360/tables>

Chapter 32: Tabs

Examples

Basic HTML

```
<ul class="nav nav-tabs" role="tablist">
  <li role="presentation">
    <a href="#id-of-content-1" role="tab" data-toggle="tab">Tab 1</a>
  </li>
  <li role="presentation">
    <a href="#id-of-content-2" role="tab" data-toggle="tab">Tab 2</a>
  </li>
  <li role="presentation">
    <a href="#id-of-content-3" role="tab" data-toggle="tab">Tab 3</a>
  </li>
</ul>

<div class="tab-content">
  <div role="tabpanel" id="id-of-content-1" class="tab-pane">Tab content 1</div>
  <div role="tabpanel" id="id-of-content-2" class="tab-pane">Tab content 2</div>
  <div role="tabpanel" id="id-of-content-3" class="tab-pane">Tab content 3</div>
</div>
```

This will create a tab set with 3 tabs and 3 associated content divs.

Animated Tabs

To make tabs fade in, add `.fade` to each `.tab-pane`. The active tab pane must also have `.in` class to make the initial content visible.

```
<ul class="nav nav-tabs" role="tablist">
  <li role="presentation">
    <a href="#id-of-content-1" role="tab" data-toggle="tab">
      Tab 1
    </a>
  </li>
  <li role="presentation" class="active">
    <a href="#id-of-content-2" role="tab" data-toggle="tab">
      Tab 2
    </a>
  </li>
  <li role="presentation">
    <a href="#id-of-content-3" role="tab" data-toggle="tab">
      Tab 3
    </a>
  </li>
</ul>

<div class="tab-content">
  <div role="tabpanel" id="id-of-content-1" class="tab-pane fade">
    Tab content 1
  </div>
  <div role="tabpanel" id="id-of-content-2" class="tab-pane fade active in">
```

```
        Tab content 2
    </div>
    <div role="tabpanel" id="id-of-content-3" class="tab-pane fade">
        Tab content 3
    </div>
</div>
```

Read Tabs online: <https://riptutorial.com/twitter-bootstrap/topic/5980/tabs>

Chapter 33: Tooltip

Remarks

The tooltip is a user interface element that looks like a small pop-up box. It is usually triggered when a user hovers their pointer over an other element, without clicking it.

For performance reasons, tooltips must be initialized with jQuery. The following code will enable all tooltips in the DOM:

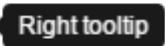
```
<script>
  $(document).ready(function() {
    $('[data-toggle="tooltip"]').tooltip();
  });
</script>
```

Examples

Positioning Tooltips

By default, the tooltip will appear on top of the element. We can use `data-placement` attribute to set the position of the tooltip on top, bottom, left or the right side of the element.

```
<a href="#" data-toggle="tooltip" data-placement="top" title="Top tooltip">Hover</a>
<a href="#" data-toggle="tooltip" data-placement="bottom" title="Bottom tooltip">Hover</a>
<a href="#" data-toggle="tooltip" data-placement="left" title="Left tooltip">Hover</a>
<a href="#" data-toggle="tooltip" data-placement="right" title="Right tooltip">Hover</a>
```

Hover Hover Hover Hover 

We can also use `data-placement="auto"`, to dynamically reorient the tooltip. The tooltip in the next example the tooltip will display to the left when possible, otherwise it will display right.

```
<a href="#" data-toggle="tooltip" data-placement="auto left" title="To the left?">Hover</a>
```

Basic Example

To create a tooltip, we only need to add `data-toggle="tooltip"` attribute and a `title` to the HTML element that will have the tooltip. Title attribute is used to specify the text that is displayed inside the tooltip.

```
<span data-toggle="tooltip" title="Hello world!">Hover over me</span>
```

Hello world!

Hover over me

Read Tooltip online: <https://riptutorial.com/twitter-bootstrap/topic/3731/tooltip>

Chapter 34: Twitter Bootstrap Style Customization

Remarks

One thing to note is that one has to mention `custom.css` name after the main `bootstrap.css` , otherwise the values of `custom.css` won't get actually implemented.

Examples

Overriding Default CSS

Everybody loves [twitter bootstrap](#), but some of us don't like it's default design. So here's a simple guide on how to start customizing bootstrap's design. Twitter bootstrap when cloned provides a set of default css files which we can override.

The main css file that we need to override is the `bootstrap.min.css` under the `bootstrap/dist/css` directory.

To override bootstrap's default design follow this 2 easy steps.

1. Create a `custom.css` (or you can name it whatever you want) and link it to your `index.html`

```
<html>
<head>
  <title>Customize Bootstrap</title>

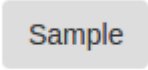
  <link rel="stylesheet" type="text/css" href="path/to/bootstrap.min.css">
  <!-- This must be declared after the bootstrap.min.css -->
  <link rel="stylesheet" type="text/css" href="path/to/your/custom.css">
</head>
<body>
  <!-- Do something -->
</body>
</html>
```

2. Start customizing. For example we want to change the color of the default button. If you want to use bootstrap's default button style you need to add the `btn` class on your `<button class="btn">Sample</button>` tag. Just write the following code on your `custom.css`.

```
.btn{
  background-color:red;
}
```

The code above will produce something like this.

Default :

A rectangular button with a light gray background and rounded corners. The word "Sample" is centered in a dark gray, sans-serif font.

Custom :

A rectangular button with a solid red background and rounded corners. The word "Sample" is centered in a white, sans-serif font.

This technique will save us from rewriting the whole button styles that were already written by bootstrap contributors. This also saved us from writing our own css class which for me is less tedious.

Read Twitter Bootstrap Style Customization online: <https://riptutorial.com/twitter-bootstrap/topic/6030/twitter-bootstrap-style-customization>

Chapter 35: Using Clearfix in Rows and Cols

Introduction

When creating advanced layouts, there may be scenarios when you'll need to use **more than 12 column units** in a single `.row` element. The concept of **column wrapping** and **responsive resets** (A.K.A. **clearfixes**) are *essential* to understanding responsive design with Bootstrap.

Basics of the Bootstrap Grid

Remarks

Bootstrap's grids are remarkably powerful and elegant. However, you must remember that the name of the framework is "Bootstrap", not "WeDidItForYou". Bootstrap **enables** responsive design, it does not **guarantee** it.

It is still up to you to make your design truly responsive, and give your users the best possible end-user experience.

Examples

The Naive First Attempt

Before we begin, let's define some CSS for the examples. This is the `head` section of our sample. I always use `border-radius` and `background-color` when I'm testing, because it makes seeing cell divisions simple without adding any border size which could affect the size of the cells.

```
<head>
  <title></title>
  <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <style>
    /* colorize all col- */
    [class^="col-"] {
      min-height: 30px;
      border-radius: 10px;
      background-color: lightblue;
    }
    /* a tall cell */
    .cell-tall {
      height: 100px;
      background-color: orange;
    }
    /* a medium-height cell */
    .cell-med {
      height: 50px;
      background-color: lightgreen;
    }
    /* padding top-bottom for some row examples */
    .row.padded {
```

```
padding: 1rem 0 1rem 0;
}
</style>
</head>
```

With that out of the way, let's define a grid and look at the perfect results at all viewport sizes!

Using `col-xs-6 col-md-3`

```
<div class="container-fluid">
  <div class="row">
    <div class="col-xs-6 col-md-3">1</div>
    <div class="col-xs-6 col-md-3">2</div>
    <div class="col-xs-6 col-md-3">3</div>
    <div class="col-xs-6 col-md-3">4</div>
    <div class="col-xs-6 col-md-3">5</div>
    <div class="col-xs-6 col-md-3">6</div>
    <div class="col-xs-6 col-md-3">7</div>
    <div class="col-xs-6 col-md-3">8</div>
    <div class="col-xs-6 col-md-3">9</div>
    <div class="col-xs-6 col-md-3">10</div>
    <div class="col-xs-6 col-md-3">11</div>
  </div>
</div>
```

`col-xs-6 col-md-3`



`col-xs-6 col-md-3`



The previous two images show the rendering at medium and small screen sizes. Remember, we'll get FOUR columns on medium+ because of `col-md-3`, and TWO cells at small- because of `col-xs-6`.

Looks pretty good, right? I think we're done here! Said a LOT of naive Bootstrap sites out there just waiting to break...

The Height Problem

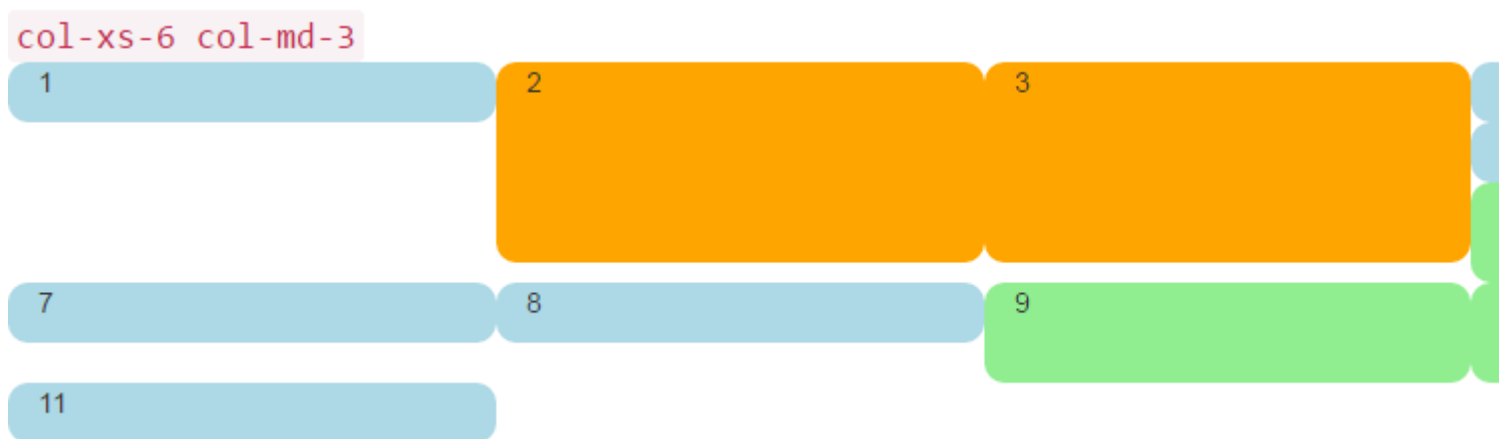
In our "naive example", all of our cells were the same height. The browser willingly broke the lines exactly where we wanted, and all seemed right with the world. Until height comes into the picture.

Let's take the previous example and give some height to some of the cells, maybe like you would see on a dashboard-type page.

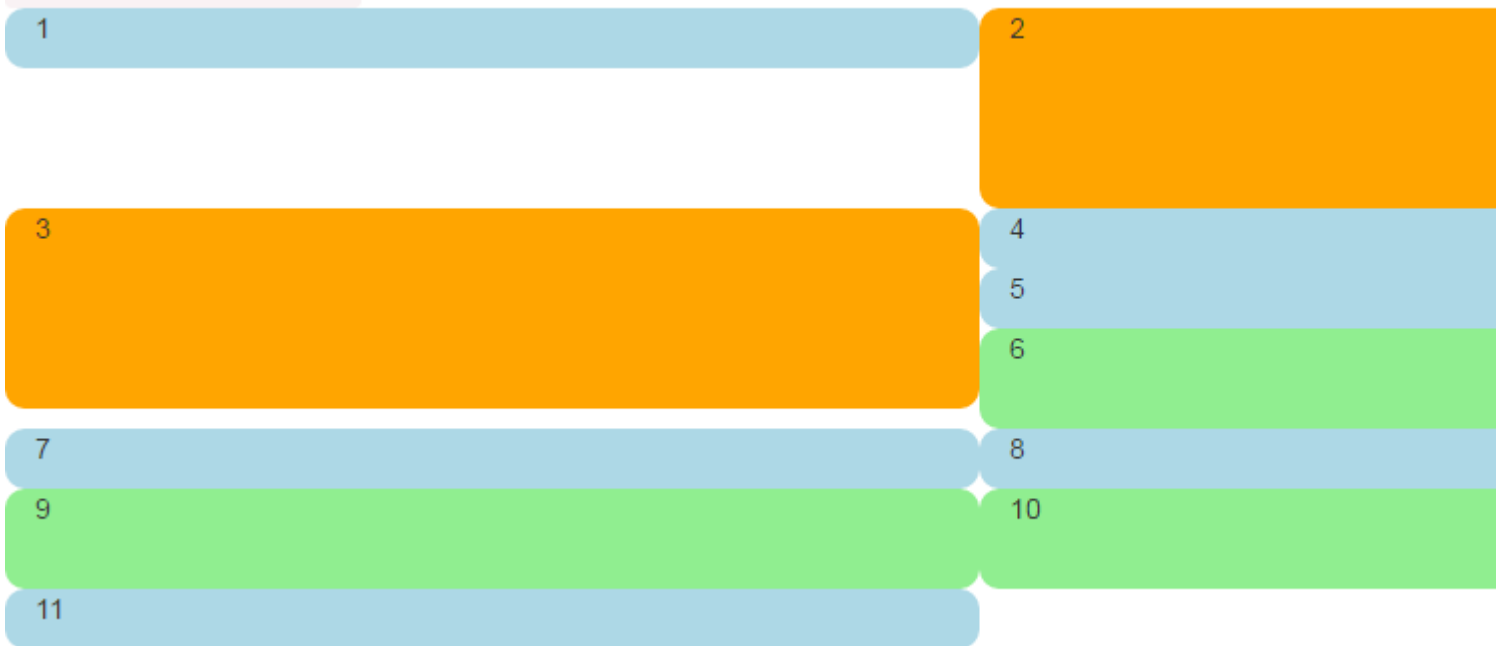
```
<div class="container-fluid">
  <div class="row">
    <div class="col-xs-6 col-md-3">1</div>
    <div class="col-xs-6 col-md-3 cell-tall">2</div>
    <div class="col-xs-6 col-md-3 cell-tall">3</div>
    <div class="col-xs-6 col-md-3">4</div>
    <div class="col-xs-6 col-md-3">5</div>
    <div class="col-xs-6 col-md-3 cell-med">6</div>
    <div class="col-xs-6 col-md-3">7</div>
    <div class="col-xs-6 col-md-3">8</div>
    <div class="col-xs-6 col-md-3 cell-med">9</div>
    <div class="col-xs-6 col-md-3 cell-med">10</div>
    <div class="col-xs-6 col-md-3">11</div>
  </div>
</div>
```

Here we have added some `cell-tall` and `cell-med` CSS that we defined above. This will have the effect of changing the height of some of the cells. I wonder how it will look...

Here they are again at medium and small screen sizes:



col-xs-6 col-md-3



Oh my, what a mess. I don't think that's what we wanted. At medium-large size, 5 and 6 are way out of place, and somehow 7 ended up starting a new row. At small size we have two cells in the first row, and **four** in the second row, with 4, 5, and 6 all stacked up on the right at both screen sizes!

So, how do we solve this?

Clearfix to the Rescue

One way to help the situation certainly, would be to use more than one `row`:

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

This is usually the first thing that new Bootstrappers try. It seems to make sense: "I want four cells in each row, so I'll just create a new `row` for each 4 `col` divs".

But there is problem with this line of reasoning: The whole point of Bootstrap 3 and the upcoming version 4 is to be **responsive**. By placing "four `col` in a `row`", you are not really "thinking responsively".

A good understanding of the `clearfix` CSS class will help you begin to see that multiple `row` divs have really been clouding your understanding of the way that responsive design was **meant** to work. In short, you simply **cannot** know how many `col` to put in a `row` anyway - the browser hasn't

rendered your work yet!

Remember in First Things First, we said you need to think in "inverse of 12"? Without further ado, let's fix our problem here, using comments right in the code to hopefully clear up any confusion. Yes, it looks like a lot more code, but **most** of the extra is comments.

```
<div class="container-fluid">
  <div class="row">
    <div class="col-xs-6 col-md-3">1</div>
    <div class="col-xs-6 col-md-3 cell-tall">2</div>
    <!--
      We have rendered TWO cells.
      On small and extra small devices, the viewport will render TWO cells
      (12 / 6 = 2), so we need a clearfix every TWO cells. We also need to
      say "don't show this clearfix when the viewport will render FOUR cells",
      which it will do at medium size and up (12 / 3 = 4). We do that by adding
      hidden-md and hidden-lg to our clearfix div, in effect instructing the
      browser to not show it at all on a wider screen.
    -->
    <div class="clearfix hidden-md hidden-lg"></div>
    <!-->
    <div class="col-xs-6 col-md-3 cell-tall">3</div>
    <div class="col-xs-6 col-md-3">4</div>
    <!--
      We have now rendered FOUR cells.
      We are never going to have more than FOUR cells side by side. So every
      FOURTH cell, we place a clearfix that will ALWAYS show. We do this by
      just leaving off any of the hidden-* classes.
    -->
    <div class="clearfix"></div>
    <!-->
    <div class="col-xs-6 col-md-3">5</div>
    <div class="col-xs-6 col-md-3 cell-med">6</div>
    <!--
      We have now rendered SIX cells.
      After the sixth cell, we are at a multiple of TWO, but not FOUR so we
      repeat the clearfix that we used after cell TWO.
    -->
    <div class="clearfix hidden-md hidden-lg"></div>
    <!-->
    <div class="col-xs-6 col-md-3">7</div>
    <div class="col-xs-6 col-md-3">8</div>
    <!--
      Now we have rendered EIGHT cells, which is a multiple of TWO AND FOUR,
      so we put in a clearfix that's always visible.
    -->
    <div class="clearfix"></div>
    <!-->
    <div class="col-xs-6 col-md-3 cell-med">9</div>
    <div class="col-xs-6 col-md-3 cell-med">10</div>
    <!--
      After the 10th cell, once again a multiple of TWO but not FOUR...
    -->
    <div class="clearfix hidden-md hidden-lg"></div>
    <!-->
    <div class="col-xs-6 col-md-3">11</div>
  </div>
</div>
```

The `clearfix` is a CSS class that renders a tiny (virtually invisible) div, and its purpose is to "clear" the `left` floats that have been used by the `col` divs.

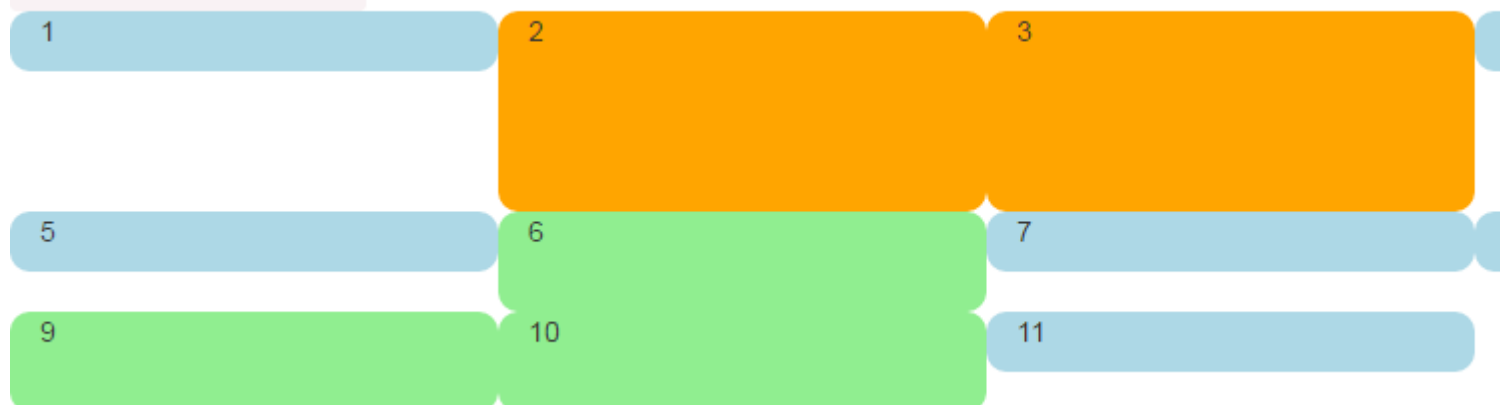
The genius is really in the `hidden-sm`, `hidden-md`, etc classes. These classes are placed **on the `clearfix` div**, NOT on the `col` divs! This causes the `clearfix` div to magically appear or disappear from the rendering stream at certain viewport widths! Genius!

Bootstrap has a baffling array of `hidden-*` and `visible-*` classes in version 3, and unfortunately they are not really the "inverse" of one another. So I find it clearest and safest to just always use the `hidden-*` classes on the clearfixes.

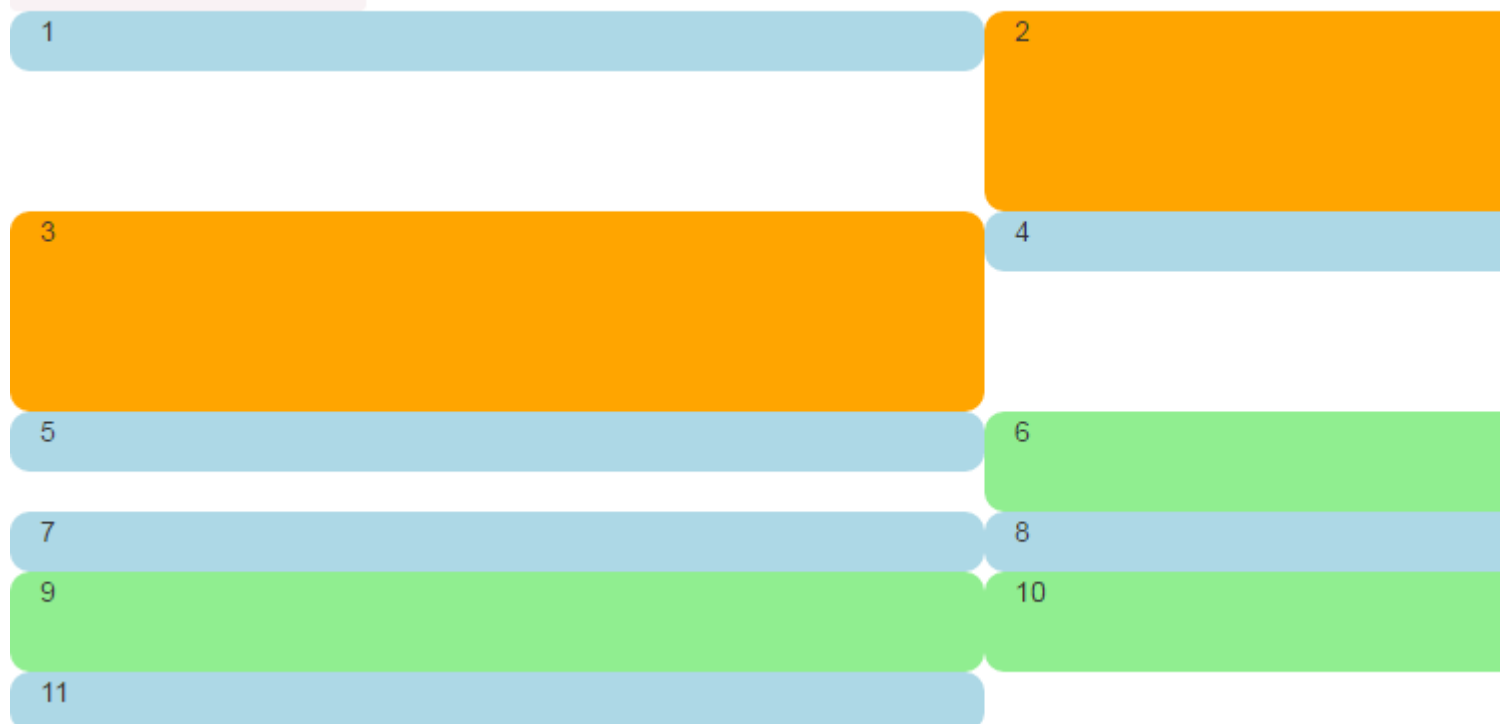
This looks like it may change for the better in Bootstrap 4, with classes like `hidden-*-up` and `hidden-*-down` (they are getting rid of the `visible-*` classes entirely).

Well enough verbiage, what does it look like now?

`col-xs-6 col-md-3`



`col-xs-6 col-md-3`



That's what we want! In the large screen, we always have FOUR across, in the smaller screen, always TWO across. No more stacking in weird places, and gaps are where we would expect them to be.

A Dashboard

Well enough of those colored rounded things, let's put something more interesting than numbers in those divs. Let's take that same set of columns and make a real dashboard. Use the following CSS:

```
<head>
  <title></title>
  <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <style>
    body {
      padding-top: 15px;
    }
    .panel-tall .panel-body {
      height: 175px;
    }
    .panel-med .panel-body {
      height: 100px;
    }
    .panel-short .panel-body {
      height: 70px;
    }
  </style>
</head>
```

And here is the "dashboard" code:

```
<div class="container-fluid">
  <div class="row">
    <div class="col-xs-6 col-md-3">
      <div class="panel panel-default panel-med">
        <div class="panel-heading">
          Heading 1
        </div>
        <div class="panel-body">
          Body 1
        </div>
        <div class="panel-footer">
          Footer 1
        </div>
      </div>
    </div>
    <div class="col-xs-6 col-md-3 cell-tall">
      <div class="panel panel-danger panel-tall">
        <div class="panel-heading">
          Heading 2
        </div>
        <div class="panel-body">
          Body 2. Look out, this needs some attention!
        </div>
      </div>
    </div>
  </div>
</div>
```

```

        <div class="panel-footer">
            Footer 2
        </div>
    </div>
</div>
<!--
    On small and extra small devices, the viewport will render TWO cells
    (12 / 6 = 2), so we need a clearfix every TWO cells. We also need to
    say "don't show this clearfix when the viewport will render FOUR cells",
    which it will do at medium size and up (12 / 3 = 4). We do that by adding
    hidden-md and hidden-lg to our clearfix div, in effect instructing the
    browser to not show it at all on a wider screen.
-->
<div class="clearfix hidden-md hidden-lg"></div>
<!-->
<div class="col-xs-6 col-md-3 cell-tall">
    <div class="panel panel-success panel-short">
        <div class="panel-heading">
            Heading 3
        </div>
        <div class="panel-body">
            Body 3. The file has successfully uploaded.
        </div>
        <div class="panel-footer">
            Footer 3
        </div>
    </div>
</div>
<div class="col-xs-6 col-md-3">
    <div class="panel panel-default panel-tall">
        <div class="panel-heading">
            Heading 4 Chart
        </div>
        <div class="panel-body">
            Body 4. Is this a cool graph or what?
        </div>
        <div class="panel-footer">
            Footer 4
        </div>
    </div>
</div>
</div>
<!--
    We are never going to have more than FOUR cells. So every FOURTH cell,
    we place a clearfix that will ALWAYS show. We do this by just leaving off
    any of the hidden-* classes.
-->
<div class="clearfix"></div>
<!-->
<div class="col-xs-6 col-md-3">
    <div class="panel panel-warning panel-short">
        <div class="panel-heading">
            Heading 5
        </div>
        <div class="panel-body">
            Body 5.
        </div>
        <div class="panel-footer">
            Footer 5
        </div>
    </div>
</div>
</div>

```

```

<div class="col-xs-6 col-md-3 cell-med">
  <div class="panel panel-warning panel-tall">
    <div class="panel-heading">
      Heading 6
    </div>
    <div class="panel-body">
      Body 6.
    </div>
  </div>
</div>
<!--
  After the sixth cell, we are at a multiple of TWO, but not FOUR so we
  repeat the clearfix that we used after cell TWO.
-->
<div class="clearfix hidden-md hidden-lg"></div>
<!------>
<div class="col-xs-6 col-md-3">
  <div class="panel panel-info panel-tall">
    <div class="panel-heading">
      Heading 7
    </div>
    <div class="panel-body">
      Body 7.
    </div>
    <div class="panel-footer">
      Footer 7
    </div>
  </div>
</div>
<div class="col-xs-6 col-md-3">
  <div class="panel panel-info panel-med">
    <div class="panel-heading">
      Heading 8
    </div>
    <div class="panel-body">
      Body 8.
    </div>
    <div class="panel-footer">
      Footer 8
    </div>
  </div>
</div>
<!--
  Now we have rendered EIGHT cells, which is a multiple of TWO AND FOUR,
  so we put in a clearfix that's always visible.
-->
<div class="clearfix"></div>
<!------>
<div class="col-xs-6 col-md-3 cell-med">
  <div class="panel panel-info panel-short">
    <div class="panel-heading">
      Heading 9
    </div>
    <div class="panel-body">
      Body 9.
    </div>
    <div class="panel-footer">
      Footer 9
    </div>
  </div>
</div>

```

```

<div class="col-xs-6 col-md-3 cell-med">
  <div class="panel panel-info panel-tall">
    <div class="panel-heading">
      Heading 10
    </div>
    <div class="panel-body">
      Body 10.
    </div>
    <div class="panel-footer">
      Footer 10
    </div>
  </div>
</div>
<!--
  After the 10th cell, once again a multiple of TWO but not FOUR...
-->
<div class="clearfix hidden-md hidden-lg"></div>
<!------>
<div class="col-xs-6 col-md-3">
  <div class="panel panel-info panel-tall">
    <div class="panel-heading">
      Heading 11
    </div>
    <div class="panel-body">
      Body 11.
    </div>
    <div class="panel-footer">
      Footer 11
    </div>
  </div>
</div>
</div>
</div>

```

That code will look like this:

Heading 1
Body 1
Footer 1

Heading 2
Body 2. Look out, this needs some attention!
Footer 2

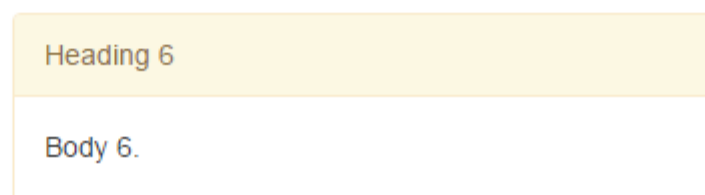
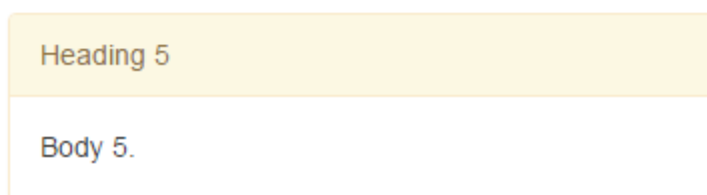
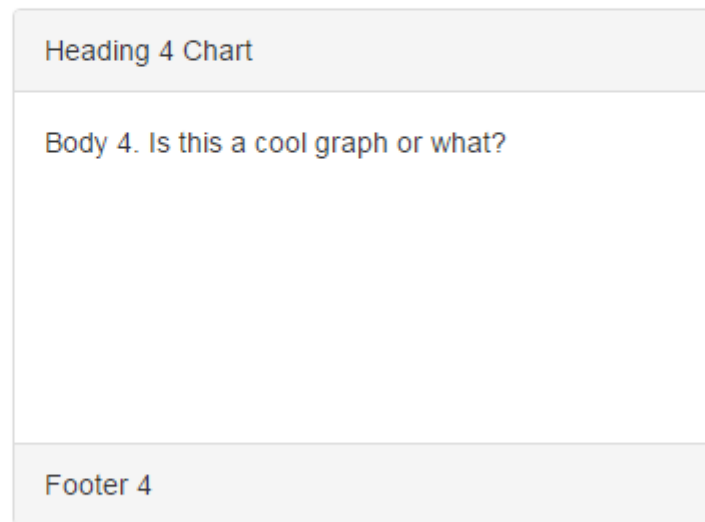
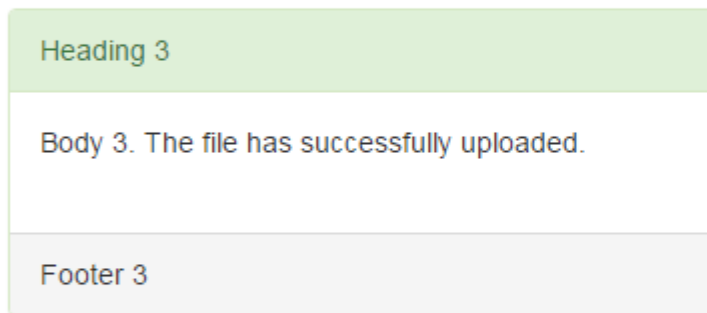
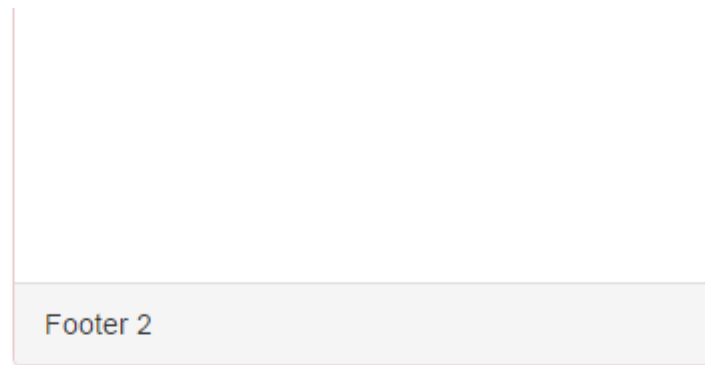
Heading 3
Body 3. The file has successfully uploaded.
Footer 3

Heading 5
Body 5.
Footer 5

Heading 6
Body 6.

Heading 7
Body 7.
Footer 7

And like this in smaller viewports:



By the way I'm using the Bootstrap 3 `panel` class, which will go away in Bootstrap 4 and be replaced by the much more descriptive and specific `card`. Looking at these images, you can see why `card` will be a much better name than the ambiguous `panel`.

2,4,6 Layout with Clearfixes

Here's a layout that renders two, four, or six cells across depending on screen size.

```
<div class="container-fluid">
  <div class="row">
    <div class="col-xs-6 col-md-3 col-lg-2">1</div>
    <div class="col-xs-6 col-md-3 col-lg-2 cell-tall">2</div>
  <!--
    On small and extra small devices, the viewport will render TWO cells
    (12 / 6 = 2), so we need a clearfix every TWO cells. We also need to
    say "don't show this clearfix when the viewport will render FOUR cells",
    which it will do at medium size (12 / 3 = 4). We do that by adding
    hidden-md and hidden-lg to our clearfix div, in effect instructing the
    browser to not show it at all on a wider screen.
  -->
  <div class="clearfix hidden-md hidden-lg"></div>
<!-->
```

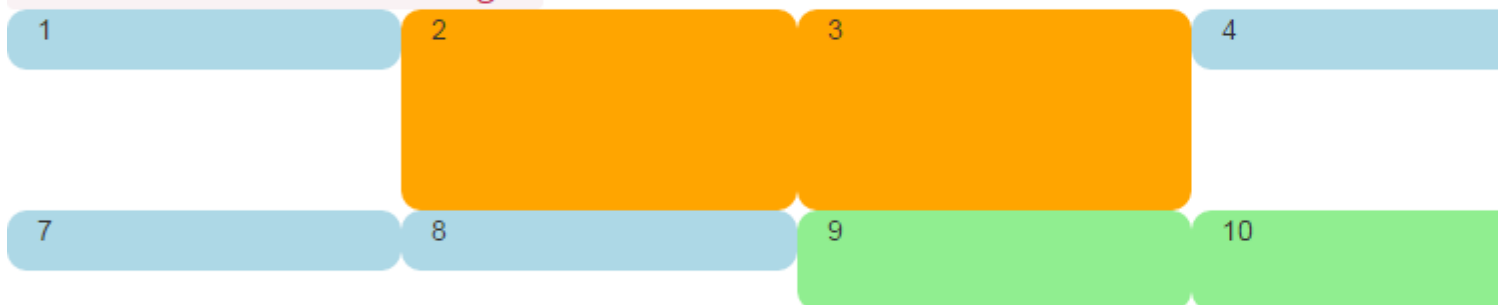
```

<div class="col-xs-6 col-md-3 col-lg-2 cell-tall">3</div>
<div class="col-xs-6 col-md-3 col-lg-2">4</div>
<!--
    After the FOURTH cell, we need a clearfix, but it still needs to be
    hidden on LARGE viewports, because remember we will have a maximum of
    SIX cells now.
-->
<div class="clearfix hidden-lg"></div>
<!-->
<div class="col-xs-6 col-md-3 col-lg-2">5</div>
<div class="col-xs-6 col-md-3 col-lg-2 cell-med">6</div>
<!--
    After the SIXTH cell, we need to show on SMALL and LARGE, but not on
    MEDIUM. Remember, our MEDIUM viewport only wants a clearfix when we
    are at a multiple of FOUR.
-->
<div class="clearfix hidden-md"></div>
<!-->
<div class="col-xs-6 col-md-3 col-lg-2">7</div>
<div class="col-xs-6 col-md-3 col-lg-2">8</div>
<!--
    Now we have rendered EIGHT cells, which is a multiple of TWO AND FOUR,
    so we put in a clearfix that's not visible on LARGE, because we are NOT
    at a multiple of SIX.
-->
<div class="clearfix hidden-lg"></div>
<!-->
<div class="col-xs-6 col-md-3 col-lg-2 cell-med">9</div>
<div class="col-xs-6 col-md-3 col-lg-2 cell-med">10</div>
<!--
    After the 10th cell, small only.
-->
<div class="clearfix hidden-md hidden-lg"></div>
<!-->
<div class="col-xs-6 col-md-3 col-lg-2">11</div>
</div>
</div>

```

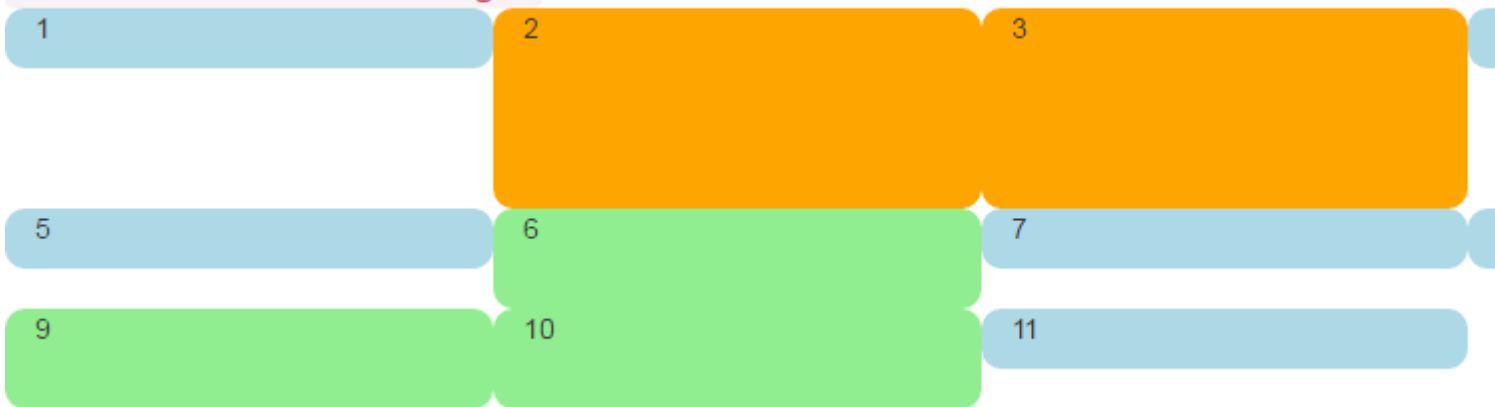
Large Screen:

col-xs-6 col-md-3 col-lg-2



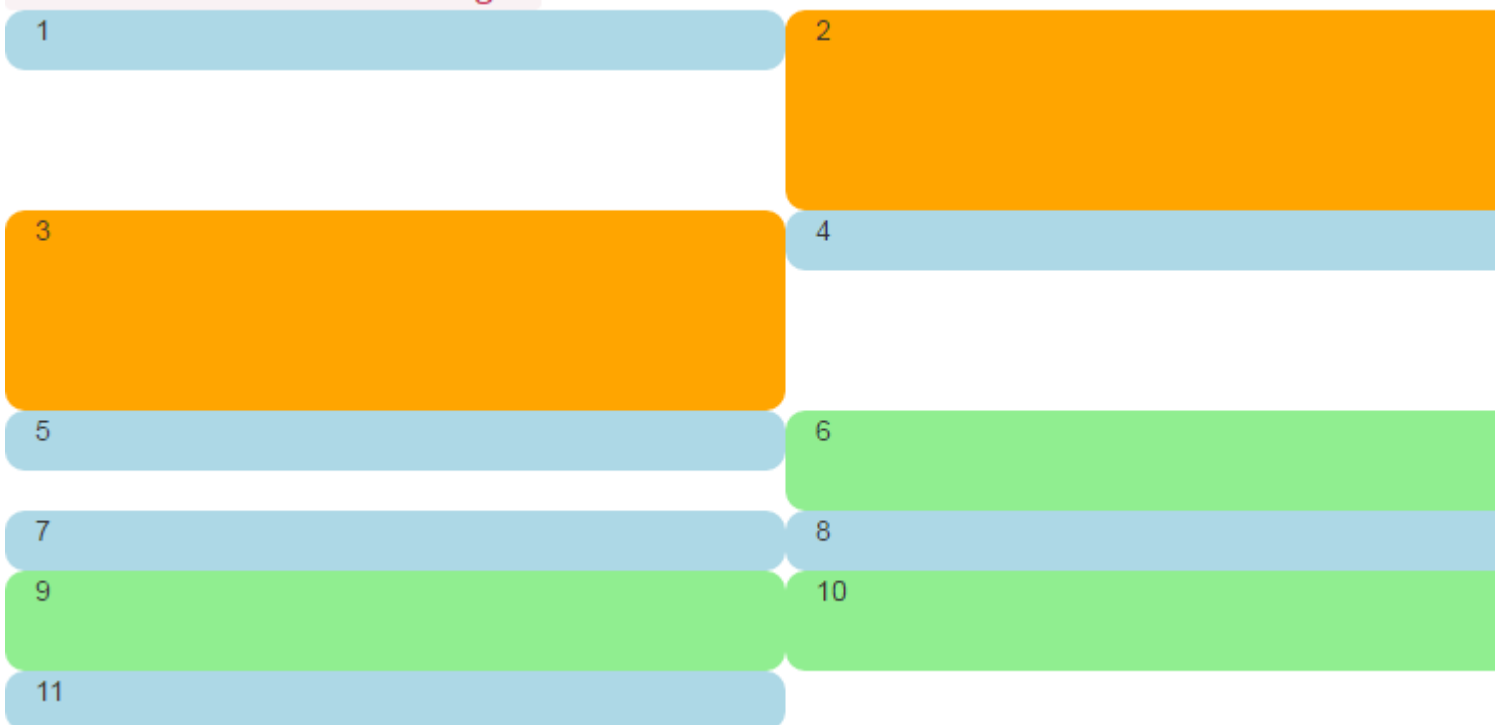
Medium Screen:

`col-xs-6 col-md-3 col-lg-2`



Small Screen:

`col-xs-6 col-md-3 col-lg-2`



Why Would Bootstrap Columns Exceed 12 in a Row?

There are many responsive scenarios where it's *necessary* to have column units exceeding 12 in a single `.row` element. This is known as [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.

For example, consider a layout where we want...

- 3 columns across on large & medium devices, and
- 2 columns across on small & smallest devices

Large



Small



To get this layout in Bootstrap, we'd use (correct)..

```
<div class="row">
  <div class="col-xs-6 col-md-4"> x </div>
  <div class="col-xs-6 col-md-4"> x </div>
  <div class="col-xs-6 col-md-4"> x </div>
  <div class="col-xs-6 col-md-4"> x </div>
  <div class="col-xs-6 col-md-4"> x </div>
  <div class="col-xs-6 col-md-4"> x </div>
</div>
```

Correct method demo

As you see in the example, the *total* of column units in the `.row` element **exceeds 12**. This technique is known as [column wrapping](#) and it's one of Bootstrap's most powerful responsive design features. The desired layout would *not be possible* (other than duplicating markup) if we tried to stick with the **common misconception that column units must add up to 12 in a single row**.

The layout is *not* possible when we don't exceed 12 (wrong)..

```
<div class="row">
  <div class="col-xs-6 col-md-4"> x </div>
  <div class="col-xs-6 col-md-4"> x </div>
</div>
<div class="row">
  <div class="col-xs-6 col-md-4"> x </div>
  <div class="col-xs-6 col-md-4"> x </div>
</div>
<div class="row">
  <div class="col-xs-6 col-md-4"> x </div>
  <div class="col-xs-6 col-md-4"> x </div>
</div>
```

Wrong method demo (fails 3 columns across on large)

Remember, a `.row` is *not* the same as a single line across the viewport. A `.row` is a grouping of columns. The columns exceeding 12 units in a single `.row` element will [wrap to a new line](#) (down the viewport). That's why it is essential to understand that the 12 columns represent horizontal **units** across the viewport.

Additionally, [responsive resets](#) (clearfix) must be used for even wrapping when [columns vary in height](#).

Read [Using Clearfix in Rows and Cols](#) online: <https://riptutorial.com/twitter-bootstrap/topic/6124/using-clearfix-in-rows-and-cols>

Chapter 36: Utility Classes

Examples

Generate `.hidden-*` classes for all breakpoints - SCSS

```
// Mixin to generate hidden classes
@mixin generate-hidden-classes {
  @each $bp in map-keys($grid-breakpoints) {
    .hidden-#{$bp} {
      @include media-breakpoint-only($bp) {
        display: none !important;
      }
    }
  }
}

// Call to the mixin
@include generate-hidden-classes();
```

Read Utility Classes online: <https://riptutorial.com/twitter-bootstrap/topic/6217/utility-classes>

Credits

S. No	Chapters	Contributors
1	Getting started with twitter-bootstrap	andreaem , Atul Mishra , bpoiss , Community , Evan , Gleb Kemarsky , H. Pauwelyn , kernal lora , Kritner , MattD , Mingle Li , Nhan , Prashanth Benny , the12 , tmg , Toby , VincenzoC , Vishnu Y S
2	Alert	JHS , Madalina Taina , tmg
3	Bootstrap Affix	Ilyas karim
4	Bootstrap Badges and Labels	mmativ
5	Bootstrap Components	alex
6	Bootstrap Containers	Neha Chopra
7	Bootstrap Dropdowns	Ismail Farooq , MattD
8	Bootstrap Navbar	Ilyas karim
9	Bootstrap Themes	KAI
10	Bootstrap Validation	Amy Barrett , mnoronha
11	Buttons	Madalina Taina , Muhammad Abdullah , Richard Hamilton , the12
12	Carousels	alex , Boysenb3rry
13	Columns	kybernaut.cz
14	Dropdowns	alex
15	Forms	Community , Jens , Owen Pauling
16	Glyphicons	Madalina Taina , tmg , Umer Farooq
17	Grid Nesting	neophyte , ZimSystem
18	Grid system	Ani Menon , Boysenb3rry , bpoiss , Harshal Patil , leowebguy , Madalina Taina , Mingle Li , mmativ , Stephen Leppik , the12 , tmg , ZimSystem

19	Jumbotron	Gabriel Chi Hong Lee
20	List group	Ilyas karim
21	Migrating to Bootstrap 4	Chris Farmer , neophyte , ZimSystem
22	Modal Dialogs	alex , mnoronha
23	Modals	John Blanchard
24	Navbar	Kritner , Krunal Mevada , kybernaut.cz
25	Navigation Menus	Ignacio Correia
26	Navs	leowebguy
27	Pagination	TheDarkKnight
28	Panels	JackPoint , tmg
29	Printing in Bootstrap.	MattD
30	Tables	atjoedonahue
31	Tabs	DavidG , tmg
32	Tooltip	Madalina Taina , tmg
33	Twitter Bootstrap Style Customization	CENT1PEDE , Vikas Yadav
34	Using Clearfix in Rows and Cols	Bruce Pierson , ZimSystem
35	Utility Classes	ajju