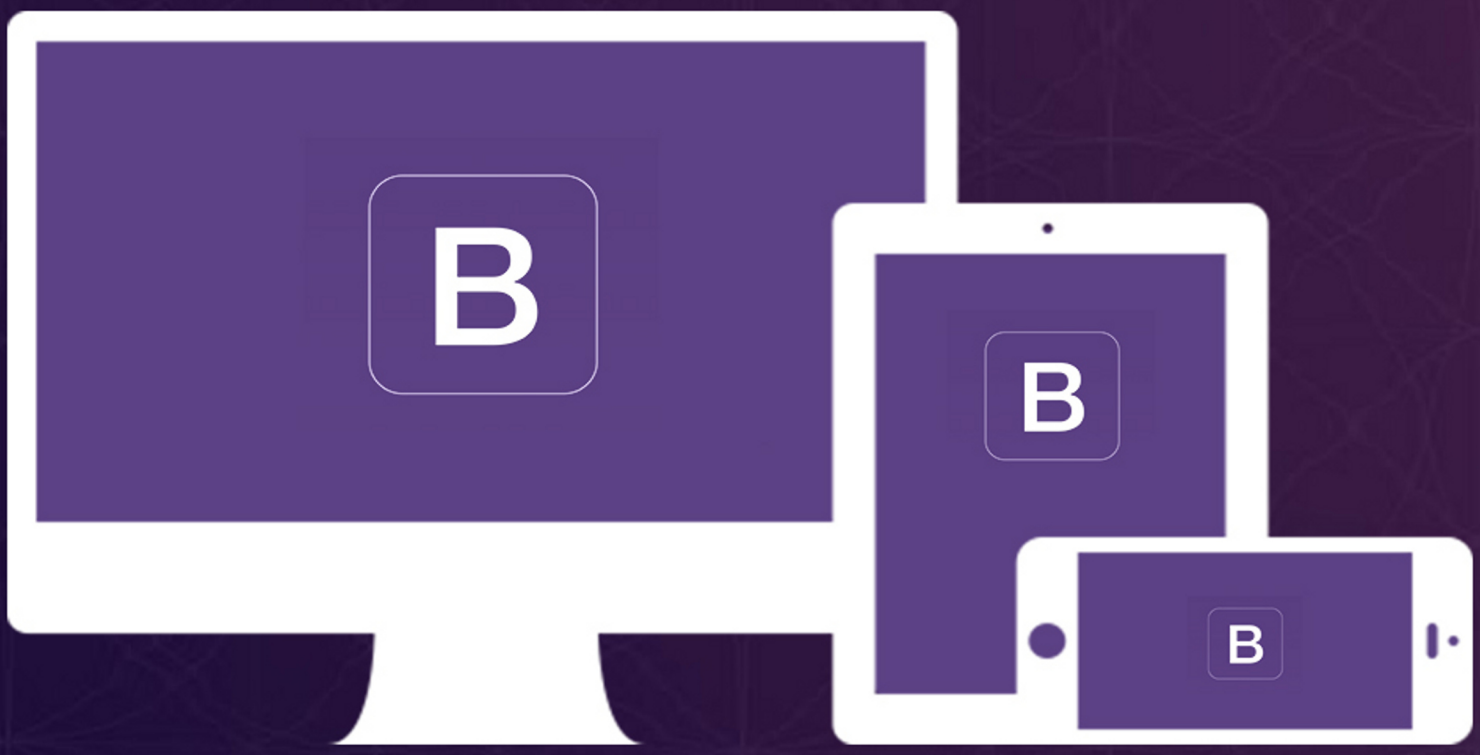
Bootstrap PROGRAMMING COOKBOOK

Hot Recipes for Bootstrap Development



FABIO CIMO

WEB CODE GEEKS
WEB DEVELOPERS RESOURCE CENTER

Bootstrap Programming Cookbook

Contents

Preface

Bootstrap is a free and open-source collection of tools for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. It aims to ease the development of dynamic websites and web applications.

Bootstrap is a front end framework, that is, an interface for the user, unlike the server-side code which resides on the "back end" or server. Bootstrap is the most-starred project on GitHub, with over 88K stars and more than 37K forks. (Source: https://en.wikipedia.org/wiki/Bootstrap_%28front-end_framework%29)

In this ebook, we provide a compilation of Bootstrap based examples that will help you kick-start your own web projects. We cover a wide range of topics, from grids and navigation bar creation, to layouts and forms design. With our straightforward tutorials, you will be able to get your own projects up and running in minimum time.

About the Author

Fabio is a passionate student in web tehnologies including front-end (HTML/CSS) and web design. He likes exploring as much as possible about the world wide web and how it can be more productive for us all. Currently he studies Computer Engineering, at the same time he works as a freelancer on both web programming and graphic design.

Chapter 1

Grid Example

In this tutorial we're considering a very fundamental concept of front-end frameworks, the grid system and we're doing so using Bootstrap. Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

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

Below, we'll have a look at how the grid works and the necessary syntax for several layout cases.

1.1 Introduction and Basic Setup

1.1.1 Introduction

- For proper alignment and padding, rows must be wrapped within a `.container` or `.container-fluid` respectively for fixed-width and full-width.
- Use rows to create horizontal groups of columns.
- Only columns may be immediate children of rows, and content should be placed within columns.
- Bootstrap offers a lot of classes to set up layouts fast, classes like `.row` to create a new row or `.col-xs-4` to create a small sized column equal to 1/3 of the container.
- Gutters between columns are created via `padding`.
- The negative margin is why the examples below are outdented. It's so that content within grid columns is lined up with non-grid content.
- Grid columns are created by specifying the number of twelve available columns you wish to span. It could wither be 12 columns by using for example `.col-sm-1` or three `.col-sm=4` or six `.col-sm-2` etc.
- When trying to add more than 12 columns within a row, the extra columns will automatically wrap onto a new line.

1.1.2 Document Setup

Before we start writing code, create a new HTML document and add the following Bootstrap's basic syntax inside:

```
<!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">
    <title>Bootstrap Basic Template</title>

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

    <!-- 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>
```

If you haven't downloaded Bootstrap, click here and extract the bootstrap folder of the archive and then add the `index.html` inside like so:

Figure 1.1: bg-1

1.2 Grid Options and Examples

1.2.1 The Grid Across Multiple Devices

The following table lets you understand the aspects of how Bootstrap creates the key breakpoints in the grid system:

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

Figure 1.2: bg-2

1.2.2 Example 1: Stacked to Horizontal

In this example we're going to use the class `.col-md-*` where * represents a number and show different layouts. First, let's show how to create a 12 column row. To begin, create a new row by adding a new `div` element with a class of `row`. Inside the row, add 12 div elements with the class `col-md-1`. This will create 12 columns.

```
<div class="container">
<div class="row">
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
  <div class="col-md-1">.col-md-1< /div >
< /div >
< /div >
```

In order to have some visual results, add the following CSS to emphasize the grid:

```
<style type="text/css">
    .col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-6, .col-md-8 {
      background-color: #34495e;
      color: white;
      line-height: 5em;
      border: 1px solid white;
      height: 5em;
    }
    </style>
```

Figure 1.3: bg-3

In the same way, we can define 2, 3, 4 ,6 or combinations like 8 and 4 columns like below:

```
<div class="container">
<div class="row">
  <div class="col-md-4">.col-md-4< /div >
  <div class="col-md-4">.col-md-4< /div >
  <div class="col-md-4">.col-md-4< /div >
< /div >

<div class="row">
  <div class="col-md-3">.col-md-3< /div >
  <div class="col-md-3">.col-md-3< /div >
  <div class="col-md-3">.col-md-3< /div >
  <div class="col-md-3">.col-md-3< /div >
< /div >

<div class="row">
  <div class="col-md-6">.col-md-6< /div >
  <div class="col-md-6">.col-md-6< /div >
< /div >

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

Notice how we separate each set of columns to define new rows. The view of this code would be:

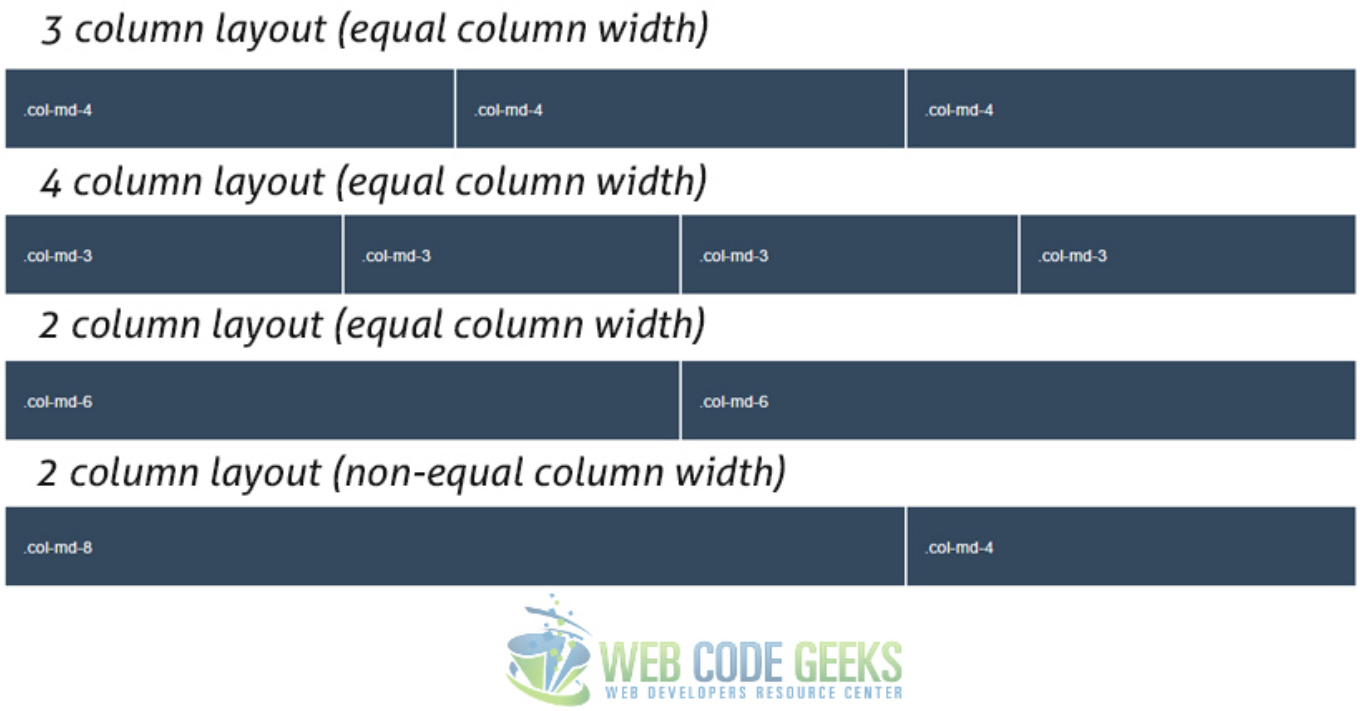


Figure 1.4: bg-4

1.2.3 Example 2 - Multi Device Support

You can define multiple different behaviours based on the device you want to have the grid for. Look at the following example:

```
<div class="row">
  <div class="col-sm-12 col-md-8">.col-sm-12 .col-md-8< /div >
  <div class="col-sm-6 col-md-4">.col-sm-6 .col-md-4< /div >
< /div >

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

<div class="row">
  <div class="col-xs-12 col-sm-6 col-md-8">.col-xs-12 .col-sm-6 .col-md-8< /div >
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4< /div >
< /div >
```

In the first row, we have defined a 2 column layout made of 8 columns (`.col-md-8`) and 4 columns (`.col-md-4`) to divide width for the medium sized devices. Then we've also added another class defining the lyout of each column on small sized devices, in this case, the first column will fill up the whole row (`.col-sm-12`) and consequently the send columns will end up on a new line with a half width of a row (`.col-sm-6`). It goes the same way for the two other cases, ecen when using three classes, to define three different layouts based on the grid breakpoints. If a column or some of them fill up the whole row with columns (12), the remaining ones (if any) will normally enter a new line.

Figure 1.5: bg-5

1.3 Offsetting and Nesting Columns

1.3.1 Example 3 - Offsetting Columns

What if you didn't want a clear layout that fills up all the row? Well, you can set up columns just the way you want by using column offsets. For example, if you wanted to create a `.col-md-3` and move it to the right leaving left just the same space that equals the width of the column, you can just write: `.col-md-3 .col-md-offset-3`. Look at the examples below for a better understanding:

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

In the browser you'd now see columns positioned in various places using the offsets.

Figure 1.6: bg-6

1.3.2 Example 4 - Nesting Columns

You can easily nest columns withing one another by just adding a row and then the nested columns. The example below gives you a clearer understanding:

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

As you can expect, now the .col-md-8 is considered as the row of the new columns nested:

Figure 1.7: bg-7

1.4 Conclusion

To conclude, Bootstrap's grid is a powerful system, fully responsive designed for a wide range of uses and compatible with many devices. Setting up your own layout is easy with the basic knowledge. If you want to create you layout full-width, you'd have to use the `container-fluid` class instead of `container` which will keep the content inside a fixed width. For more information on Bootstrap grid, please refer to the official documentation here.

1.5 Download

Download You can download the full source code of this example here: **Bootstrap Grid Tutorial**

Chapter 2

Navbar Example

The aim of this example is to show how to create a navbar using Bootstrap. Bootstrap is the world's most famous front-end framework. Navbars are responsive meta components that serve as navigation headers for your application or site. They begin collapsed (and are toggleable) in mobile views and become horizontal as the available viewport width increases.

Navbars are important part of websites because they provide a very functional and easy way for users to navigate the several pages of your websites. A navbar typically contains a logo, menu items (with some dropdowns) and optionally a search box.

2.1 Project Setup

In order to begin, first create a new project folder and add all necessary bootstrap components. Make sure you have the following folder structure in your project folder:

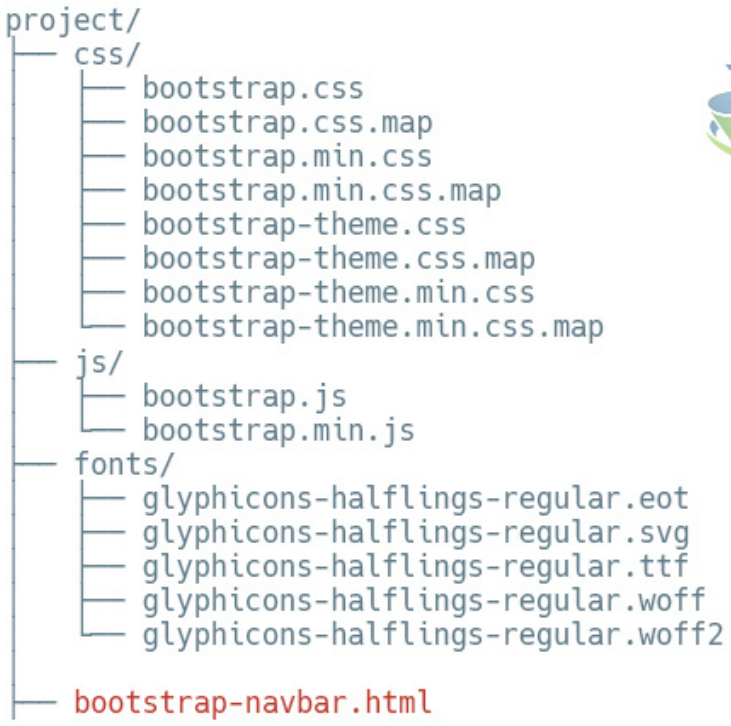


Figure 2.1: navbar-1

Next, in the `bootstrap-navbar.html` file add the basic syntax:

```
<!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">
    <title>Bootstrap Navbar Example</title>

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

    <!-- 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>
```

Note that everything else except jQuery library is linked locally.

2.2 Default Navbar

The default navbar represent the most complete navbar Bootstrap can offer as of version 3. Let's first see it and then explain every bit of code. The following code needs to be set inside the body tag of your HTML.

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data- ↩
          target="#bs-example-navbar-collapse-1" aria-expanded="false">
        <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="#">Brand</a>
    < /div >

    <!-- Collect the nav links, forms, and other content for toggling -->
    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a></li>
        <li><a href="#">Link</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><a href="#">Separated link</a></li>
            <li role="separator" class="divider"></li>
            <li><a href="#">One more separated link</a></li>
          </ul>
        </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" 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><a href="#">Separated link</a></li>
          </ul>
        </li>
      </ul>
    < /div ><!-- /.navbar-collapse -->
  < /div ><!-- /.container-fluid -->
</nav>
```

In the browser, the navbar would look like this:

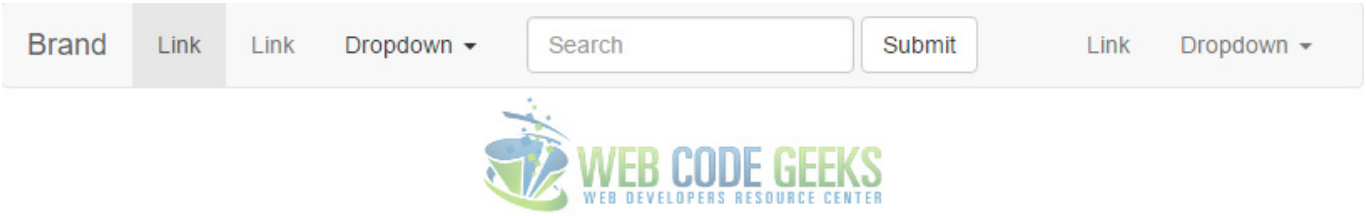


Figure 2.2: navbar-2

If you shrink the browser to a smaller size the navbar would transform, so in small devices like mobile phones, you'd have a dropdown button that would react like so when you tap:

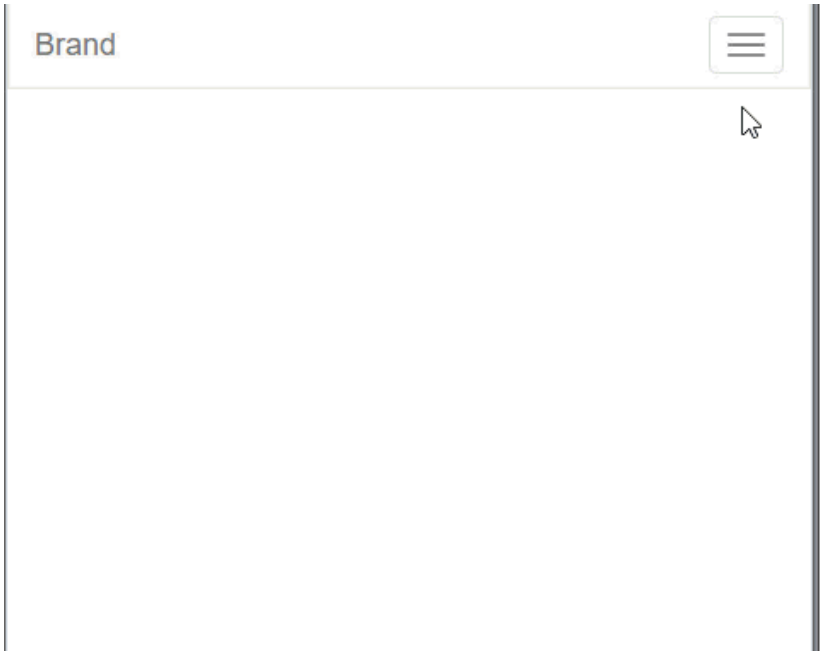


Figure 2.3: navbar-5

2.3 Navbar Components

In this section, we'll be explaining the parts of navbar used to make it the way it is. As a starting point, let's notice that the navbar is wrapped inside the common HTML5 tag `nav`. There are two classes given to this tag. `navbar` defines a navbar compontent of bootstrap styling and the other, `navbar-default` expresses a variation used among the available navbars. Next, inside a `div` which shall lie in 100% browser width (made possible by the class `container-fluid`, everything is put.

2.3.1 Navbar Header

The navbar header (not to be confused, as the navbar is already a header as a whole) is just a grouping of the brand element (dedicated space for your logo image or name) and the necessary dropdown button (which only gets shown on small devices).

Figure 2.4: navbar-6

As you might wonder what that `span` with a class of `sr-only` is doing there, just know that uou should always consider screen readers for accessibility purposes. That is why you don't see any difference when you remove that line. The attributes of the `button` (not shown all in the image) just extend functionality and takes advantage of Bootstrap's predefined classes and attributes functionality.

2.3.2 Navbar Links and Dropdowns

This sections opens with a `div` given a class of `collapse navbar-collapse` which enables this whole section to be collapsed and an id `bs-example-navbar-collapse-1` which corresponds to the id given to the button element in the header section (so the button would trigger this part of the code).

Figure 2.5: navbar-7

2.3.3 Forms

Part of a navbar may be a form element usually made of a search box input and a button which will trigger the search functionality. Therefore, a `form` element is created witht he appropriate classes `navbar-form` (defines form styling) and `navbar-left` (defines form positioning) and then every kind of input is grouped under a div with a class of `form-group` (in this case, only one input). Then, optionally, a button is added to complete the form.

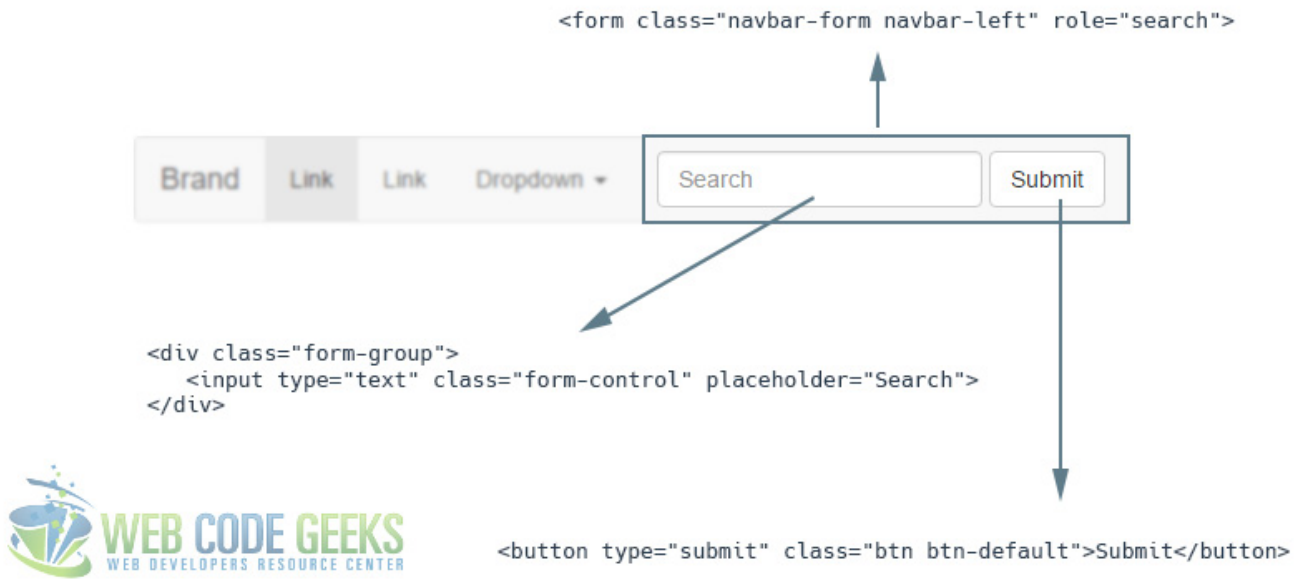


Figure 2.6: navbar-8

Note that the default navbar we declared in the beginning also has two other menu items, a link and a dropdown, but because we already saw how they can be created, no further explanation is nexessary.

2.4 Static and Fixed Navbar

There exist three main ways of navbar positioning in your webpage. They are Fixed to Top, Fixed to Bottom, and Static navbars.

2.4.1 Fixed to Top

Add `.navbar-fixed-top` and include a `.container` or `.container-fluid` to center and pad navbar content. This will keep the navbar to the top of the page regardless of the user scroll through the page. The markup would look like this:

```
<!-- NAVBAR FIXED TOP -->

<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    ...
  < /div >
</nav>
```

The fixed navbar will overlay your other content, unless you add padding to the top of the `body`. By default, the navbar is 50px high.

2.4.2 Fixed to Bottom

Add `.navbar-fixed-bottom` and include a `.container` or `.container-fluid` to center and pad navbar content. This will keep the navbar to the bottom of the page regardless of the user scroll through the page. The markup would look like this:

```
<!-- NAVBAR FIXED BOTTOM -->

<nav class="navbar navbar-default navbar-fixed-bottom">
  <div class="container">
    ...
  < /div >
</nav>
```

The same padding is required for this case, like `padding-bottom:70px;`

2.4.3 Static

Create a full-width navbar that scrolls away with the page by adding `.navbar-static-top` and include a `.container` or `.container-fluid` to center and pad navbar content. Unlike the `.navbar-fixed-*` classes, you do not need to change any padding on the body. The markup would look like this:

```
<!-- NAVBAR STATIC TOP -->

<nav class="navbar navbar-default navbar-static-top">
  <div class="container">
    ...
  < /div >
</nav>
```

2.5 Conclusion

To conclude, just like any other framework, Bootstrap's mission is to make coding as easy as possible, at the same time, providing well-organized classes and structure of the hierarchy as a whole. Navbars are probably one of the elements found on every single website. They vary from design and number of elements, but the core remains the same when using the framework. The default navbar can be customized in a number of ways that you can try at your own pace.

2.6 Download

Download You can download the full source code of this example here: **Bootstrap Navbar**

Chapter 3

Table Example

The aim of this example is to show the various Bootstrap tables that you can use in your web projects. Unless you've been living under a rock, you already know what a table is.

Tables are used to represent a set of data systematically displayed, especially in columns. That makes it easier for us to see and understand each cell's information in a practical manner.

What Bootstrap adds to traditional tables of HMTL, is certainly styling, which is the key to make a difference in layout and design. It uses simple syntax by using default tags and added classes to modify the style.

3.1 Initial Setup

The following initial setup is required in order to continue with tables demonstration.

3.1.1 Bootstrap Setup

To begin, make sure you've set up Bootstrap files and your HTML file in the following structure:

```
bootstrap-tables/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap.min.css.map
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   ├── bootstrap-theme.min.css
│   └── bootstrap-theme.min.css.map
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
│   ├── glyphicons-halflings-regular.eot
│   ├── glyphicons-halflings-regular.svg
│   ├── glyphicons-halflings-regular.ttf
│   ├── glyphicons-halflings-regular.woff
│   └── glyphicons-halflings-regular.woff2
│
├── bootstrap-tables.html
```

Figure 3.1: tables-1

3.1.2 HTML Setup

As you might already know, Bootstrap basic syntax, which links all necessary files and libraries together is:

```
<!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">
    <title>Bootstrap Tables Example</title>

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

    <!-- 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>
```

3.2 Basic Example

For basic styling-light padding and only horizontal dividers we add the base class `.table` to any `table`. It may seem super redundant, but given the widespread use of tables for other plugins like calendars and date pickers, it's opted to isolate Bootstrap's custom table styles. The basic markup would look like this:

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

For this first example, let's fill the table with information to make it feel more practical to you:

```
< table class="table" >
< thead >
< tr >
  < th >#< /th >
  < th >Name< /th >
  < th >Age< /th >
< /tr >
< /thead >
< tbody >
< tr >
  < td >1< /td >
  < td >Fabio< /td >
  < td >22< /td >
< /tr >
  < tr >
  < td >2< /td >
  < td >Jason< /td >
  < td >16< /td >
< /tr >
  < tr >
  < td >3< /td >
  < td >Rahela< /td >
  < td >25< /td >
< /tr >
< /tbody >
< /table >
```

The result in the browser would be:

#	Name	Age
1	Fabio	22
2	Jason	16
3	Rahela	25

Figure 3.2: tables-2

Notice that by default the table will display full page width (or container width), but here I've set a reduced body width for demonstration purposes.

3.3 Table Examples

The following section is a collection of other ways of styling tables rather than the default one.

3.3.1 Striped Rows

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

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

The content represented with "..." is and will be the same as the default example. The view in this case would be:

#	Name	Age
1	Fabio	22
2	Jason	16
3	Rahela	25

Figure 3.3: tables-3

3.3.2 Bordered Table

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

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

Now the table has borders on all sides and even between the cells:

#	Name	Age
1	Fabio	22
2	Jason	16
3	Rahela	25

Figure 3.4: tables-4

3.3.3 Hover Rows

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

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

Now the table has borders on all sides and even between the cells:

#	Name	Age
1	Fabio	22
2	Jason	16
3	Rahela	25

Figure 3.5: tables-5

3.3.4 Condensed table

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

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

The result would be a more compact table with minimal styling:

#	Name	Age
1	Fabio	22
2	Jason	16
3	Rahela	25

Figure 3.6: tables6

3.3.5 Contextual Classes

Use contextual classes to color table rows or individual cells. The following graphic defines each of the classes you can use:

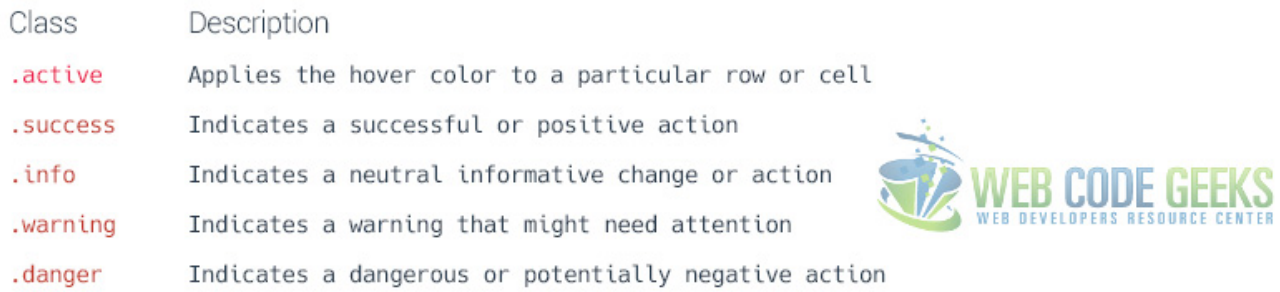


Figure 3.7: tables-7

For this example, let's once again show the table information with these classes added in some cases to table rows and in other cases to specific table data:

```
< table class="table" >
< thead >
< tr >
  < th >#< /th >
  < th >Name< /th >
  < th >Age< /th >
< /tr >
< /thead >
< tbody >
< tr >
  < td class="active" >1< /td >
  < td >Fabio< /td >
  < td >22< /td >
< /tr >
< tr class="success" >
  < td >2< /td >
  < td >Jason< /td >
  < td >16< /td >
< /tr >
< tr >
  < td >3< /td >
  < td >Rahela< /td >
  < td class="danger" >25< /td >
< /tr >
< tr class="warning" >
  < td >4< /td >
  < td >Jean< /td >
  < td >36< /td >
< /tr >
< tr >
  < td >5< /td >
  < td class="info" >Kristos< /td >
  < td >42< /td >
< /tr >
< /tbody >
< /table >
```

The result in the browser would be:

#	Name	Age
1	Fabio	22
2	Jason	16
3	Rahela	25
4	Jean	36
5	Kristos	42

Figure 3.8: tables-8

3.4 Conclusion

Bootstrap has a simple but clean table design concept that will enhance your overall information organization. Just like every other element in Bootstrap, tables are also responsive. You can create responsive tables by wrapping any `.table` inside a `.table-responsive` to make them scroll horizontally on small devices (under 768px). When viewing on anything larger than 768px wide, you will not see any difference in these tables.

3.5 Download

Download You can download the full source code of this example here: **Bootstrap Table**

Chapter 4

Dropdown Menu Example

In this example we're considering Bootstrap dropdown menus. Sometimes referred to as a pull-down menu, drop-down list, or drop-down box, a dropdown menu is a list of items that appears when clicking on a button or text selection.

For example, many programs have a "File" drop down menu at the top left of their screen. Clicking on the "File" text generates a new menu with additional options.

Bootstrap has created it's own dropdown menu, which is normally toggleable, contextual menu for displaying lists of links in your page. Bootstrap 3 is focused in offering the basic elements in a functional manner and providing essential styling that you'll most probably need. However, if that's not enough for you, feel free to browse the web for even fancier designs.

4.1 Project Setup

The following setup is required in order to continue with this tutorial.

4.1.1 Boostrap Folder Structure

First, download bootstrap from here. Next, create a new HTML document and make sure you have the following folder structure in you project folder:

```
bootstrap-dropdown/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap.min.css.map
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   ├── bootstrap-theme.min.css
│   └── bootstrap-theme.min.css.map
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
│   ├── glyphicons-halflings-regular.eot
│   ├── glyphicons-halflings-regular.svg
│   ├── glyphicons-halflings-regular.ttf
│   ├── glyphicons-halflings-regular.woff
│   └── glyphicons-halflings-regular.woff2
│
├── bootstrap-dropdown.html
```

Figure 4.1: dropdown-1

4.1.2 HTML Setup

To start coding, include Bootstrap's basic template code inside your main HTML. This will include basic HTML tags and links necessary to link Bootstrap's libraries.

```
<!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">
    <title>Bootstrap Dropdown Menu Example</title>

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

    <!-- 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>
```

4.2 The Default Example

The most basic dropdown that Bootstrap offers is the one the wraps the dropdown's trigger and the dropdown menu within `.dropdown`, or another element that declares `position:relative;`. Then add the menu's HTML.

```
<div class="dropdown">
  <button class="btn btn-default dropdown-toggle" type="button" id="dropdownMenu1" data- ↩
      toggle="dropdown" aria-haspopup="true" aria-expanded="true">
    Dropdown
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenu1">
    <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>
< /div >
```

And this is how the dropdown looks like:

Figure 4.2: dropdown-2

4.3 Cases and Examples

The following section expands the default basic options of the dropdown by adding new elements to it.

4.3.1 Dropdown Headers

You can optionally add a header to label sections of actions in any dropdown menu.

```
<ul class="dropdown-menu" aria-labelledby="dropdownMenu1">
  ...
  <li class="dropdown-header">Dropdown header</li>
  ...
</ul>
```

The header will look like this:

Figure 4.3: dropdown-3

4.3.2 Dropdown Dividers

To group links, you can add a divider to separate series of links in a dropdown menu.

```
<ul class="dropdown-menu" aria-labelledby="dropdownMenu1">
  ...
  <li role="separator" class="divider"></li>
  ...
</ul>
```

The separator will look like this:

Figure 4.4: dropdown-4

4.3.3 Disabled Menu Items

Add a `.disabled` class to a `<li>` in the dropdown to disable the link.

```
<ul class="dropdown-menu" aria-labelledby="dropdownMenu1">
  ...
  <li class="disabledr">Disabled link</li>
  ...
</ul>
```

The disabled link will look like this:

Figure 4.5: dropdown5

4.4 Did you know?

It might seem surprising, but there is also a **dropup**, apart from the traditional dropdown. The markup is somehow similar:

```
<div class="dropup">
  <button class="btn btn-default dropdown-toggle" type="button" id="dropdownMenu2" data- ↩
      toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Dropup
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenu2">
    <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>
< /div >
```

As expected the dropup would show above the button:

Figure 4.6: dropdown-6

4.5 Conclusion

To conclude, Bootstrap dropdown is an easy way to group links and menu items in a compact but effective way. By default, a dropdown menu is automatically positioned 100% from the top and along the left side of its parent. If you want to override these settings, use the dedicated classes `.dropdown-menu-right` and `.dropdown-menu-left`.

4.6 Download

Download You can download the full source code of this example here: **Boostrap Dropdown Menu**

Chapter 5

Form Example

The aim of this example is to explain and use Bootstrap forms. Just as an introduction, a web form or HTML form on a web page allows a user to enter data that is sent to a server for processing.

Forms can resemble paper or database forms because web users fill out the forms using checkboxes, radio buttons, or text fields.

Bootstrap, just like other front-end frameworks, gives forms extra styling to enhance design and user experience while using them. Here, we'll have a look at different variations of forms (meaning several input types from which a form is made up) and the necessary syntax to achieve this in Bootstrap.

5.1 Project Folder Setup

The following requirements are necessary to be present in order to continue.

5.1.1 Folder Structure

After downloading Bootstrap and creating a new empty HTML document, make sure you have the folder structure like this:

```
bootstrap-forms/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap.min.css.map
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   ├── bootstrap-theme.min.css
│   └── bootstrap-theme.min.css.map
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
│   ├── glyphicons-halflings-regular.eot
│   ├── glyphicons-halflings-regular.svg
│   ├── glyphicons-halflings-regular.ttf
│   ├── glyphicons-halflings-regular.woff
│   └── glyphicons-halflings-regular.woff2
│
├── bootstrap-forms.html
```

Figure 5.1: form-1

5.1.2 HTML Document

Your HTML file should have all Bootstrap links and libraries needed, and the syntax shall basically look like this:

```
<!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">
    <title>Bootstrap Form Example</title>

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

    <!-- 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>
```

5.2 Basic Example

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

```
  <form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" placeholder="Email">
  < /div >
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1" placeholder=" ↩
        Password">
  < /div >
  <div class="form-group">
    <label for="exampleInputFile">File input</label>
    <input type="file" id="exampleInputFile">
    Example block-level help text here.
  < /div >
  <div class="checkbox">
    <label>
      <input type="checkbox"> Check me out
    </label>
  < /div >
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

What you'd see in the browser is:

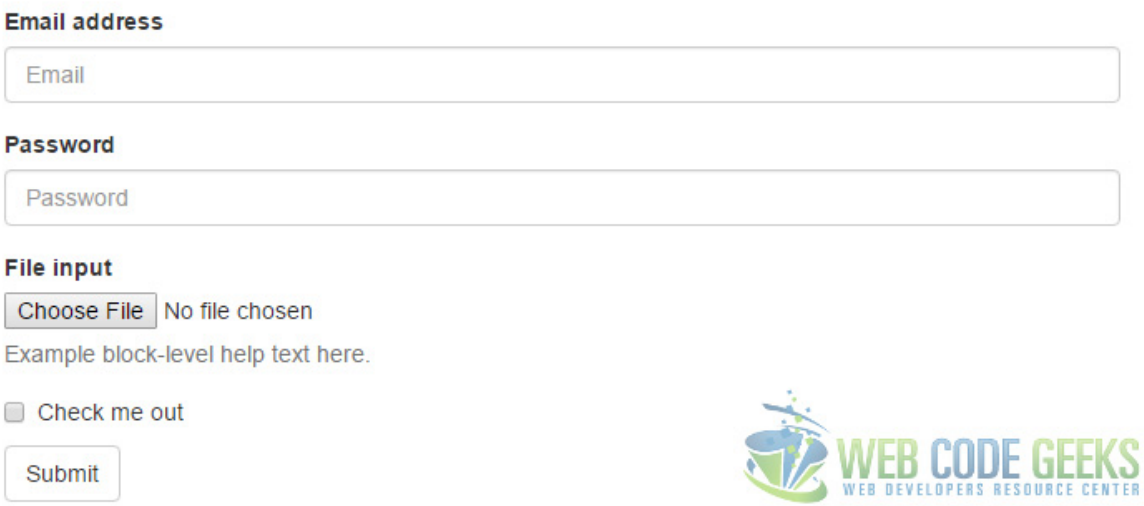


Figure 5.2: form-2

5.3 Cases and Examples

The following section gives examples of Bootstrap's varius form elements.

5.3.1 Inline Form

Add `.form-inline` to your form (which doesn't have to be a <form>) for left-aligned and inline-block controls.

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

This would produce the following result in the browser window:

Figure 5.3: form-3

5.3.2 Horizontal Form

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

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

```
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <button type="submit" class="btn btn-default">Sign in</button>
    < /div >
  < /div >
</form>
```

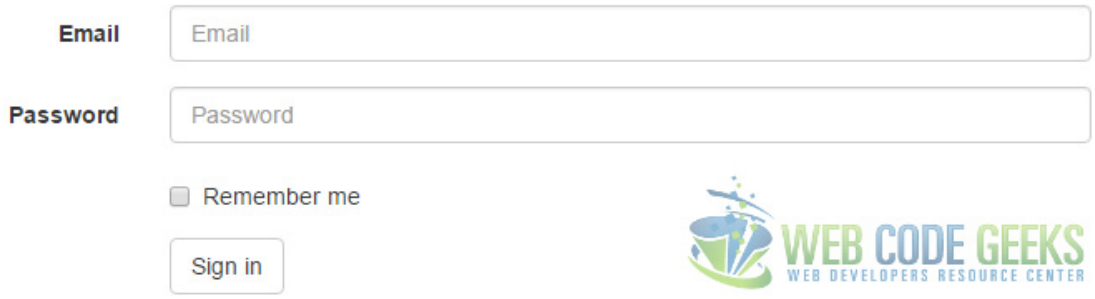


Figure 5.4: form-4

5.3.3 Supported Controls

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

5.3.3.1 Inputs

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

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

5.3.3.2 Textarea

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

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

These two elements look like this:

Figure 5.5: form-5

5.3.3.3 Checkboxes and Radios

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

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

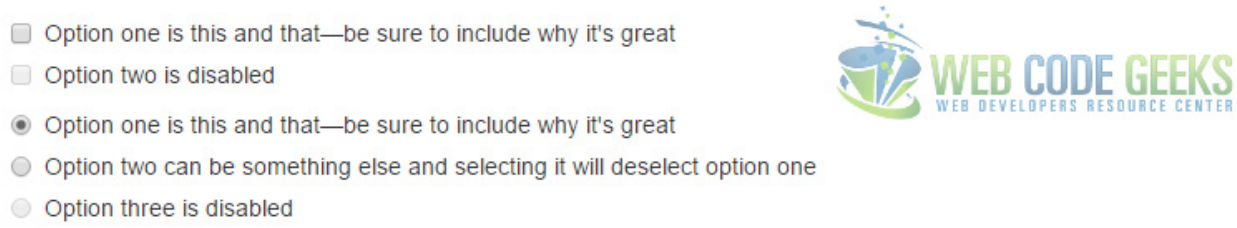


Figure 5.6: form-6

5.3.3.4 Inline checkboxes and radios

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

```
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox1" value="option1"> 1
</label>
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox2" value="option2"> 2
</label>
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox3" value="option3"> 3
</label>
<br/><br/>
<label class="radio-inline">
  <input type="radio" name="inlineRadioOptions" id="inlineRadio1" value="option1"> 1
</label>
<label class="radio-inline">
  <input type="radio" name="inlineRadioOptions" id="inlineRadio2" value="option2"> 2
</label>
<label class="radio-inline">
  <input type="radio" name="inlineRadioOptions" id="inlineRadio3" value="option3"> 3
</label>
```

Figure 5.7: form-7

5.3.3.5 Selects

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

```
<select class="form-control">
  <option>1</option>
```

```
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
</select>

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

Figure 5.8: form-8

5.3.4 Static Control

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

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

Figure 5.9: form-9

5.3.5 Readonly State

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

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

Figure 5.10: form-10

5.3.6 Validation States

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

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

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

Figure 5.11: form-11

5.4 Conclusion

To conclude, forms are important HTML elements (but not only) which are given a lot of styles by Bootstrap. Forms and their various elements have a default size (i.e. width and height). If you want a smaller or bigger size, include respective classes `.input-sm` and `.input-lg` to change the size. Forms can also be customized in a few other ways not mentioned here, but you can find these (and a lot more) in the official Bootstrap site.

5.5 Download

Download You can download the full source code of this example here: **Boostrap Form Example**

Chapter 6

Layout Example

The aim of this example is to create and use a layout with Bootstrap. Layouts are the very base model of your upcoming website. A layout usually contains the skeleton of the page, meaning how it is divided into rows and columns, the height of each section, how content is organized and overall appearance.

As you might already know, Bootstrap makes it easy to create custom layouts using pre-defined styling on so many elements. And then, what's left for you, is really few aspects like colors, fonts and images that are unique to your site. A good layout enhances user experience and makes it more convenient for users to browse the page you're building, so with a little effort in the beginning you can have a great and long-term layout.

6.1 Project Setup

The following requirements need to be met in order to continue creating a new page layout with Bootstrap.

6.1.1 Project Folder Setup

Create a new HTML file, which will be your main one, and make sure you have the following folder structure after downloading Boostrap.

```
bootstrap-layouts/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap.min.css.map
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   ├── bootstrap-theme.min.css
│   └── bootstrap-theme.min.css.map
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
├── fonts/
│   ├── glyphicons-halflings-regular.eot
│   ├── glyphicons-halflings-regular.svg
│   ├── glyphicons-halflings-regular.ttf
│   ├── glyphicons-halflings-regular.woff
│   └── glyphicons-halflings-regular.woff2
│
├── bootstrap-layouts.html
```

Figure 6.1: layout-1

6.1.2 Main HTML Setup

Bootstrap already provides a base HTML, which contains links and references to all its' libraries, including a CDN version of jQuery. Your main HTML file should have the following basic syntax:

```
<!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">
    <title>Bootstrap Layout Example</title>

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

    <!-- 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>
```

6.2 What is to be done?

Below is a general view of the layout we are going to create and populate with Bootstrap.

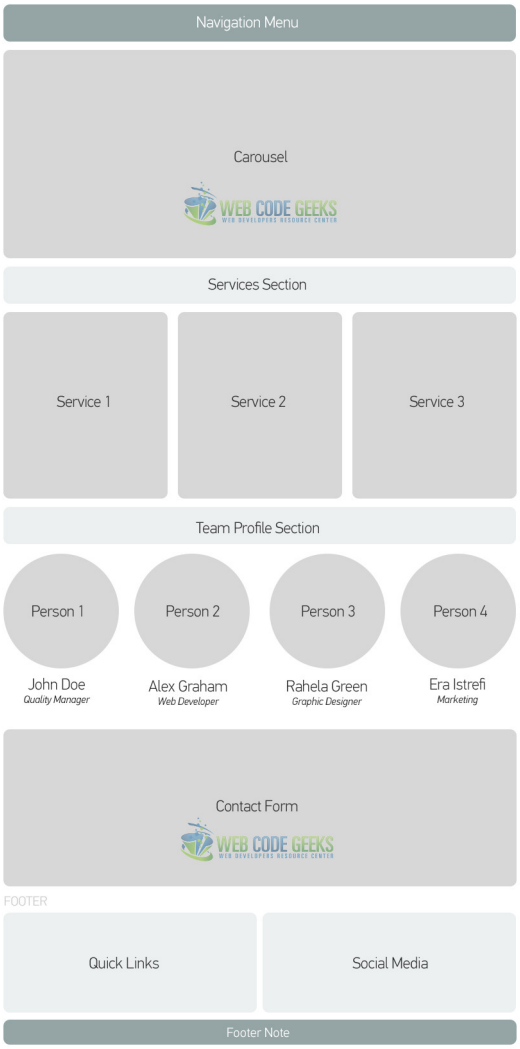


Figure 6.2: layout-21

6.3 Coding the Layout with Bootstrap

The following sections will contain the real code to achieve the layout we want.

6.3.1 Coding the Navigation Menu

To create your basic navbar, follow the default Bootstrap syntax and you'll get:

```
<div class="container">
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data- ↩
          target="#bs-example-navbar-collapse-1" aria-expanded="false">
        <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="#">Brand</a>
    < /div >

    <!-- Collect the nav links, forms, and other content for toggling -->
    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a></li>
        <li><a href="#">Link</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><a href="#">Separated link</a></li>
            <li role="separator" class="divider"></li>
            <li><a href="#">One more separated link</a></li>
          </ul>
        </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>
    < /div ><!-- /.navbar-collapse -->
  < /div ><!-- /.container-fluid -->
</nav>
< /div >
```

We just created the navbar. It looks like this in the browser:

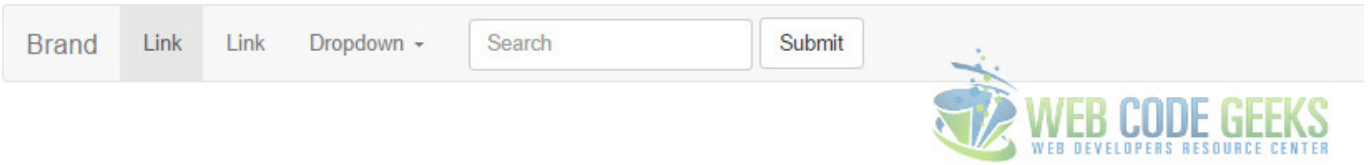


Figure 6.3: layout-3

6.3.2 Coding the Carousel

The carousel is another exciting feature of Bootstrap, and you can set it up in minutes:

```
<div class="container" style="margin-top:-1.5em">
  <div id="myCarousel" class="carousel slide" data-ride="carousel">
    <!-- Indicators -->
    <ol class="carousel-indicators">
      <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
      <li data-target="#myCarousel" data-slide-to="1"></li>
      <li data-target="#myCarousel" data-slide-to="2"></li>
    </ol>

    <!-- Wrapper for slides -->
```

```
    <div class="carousel-inner" role="listbox">
      <div class="item active">
        <img src="http://i.imgur.com/ELwJbNq.jpg?1" alt="Image1" width="1170" height="658">
      < /div >

      <div class="item">
        <img src="http://i.imgur.com/tI844yf.jpg?1" alt="Image2" width="1170" height="658">
      < /div >

      <div class="item">
        <img src="http://i.imgur.com/rMqPvig.jpg?1" alt="Image3" width="1170" height="658">
      < /div >
    < /div >

    <!-- Left and right controls -->
    <a class="left carousel-control" href="#myCarousel" 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="#myCarousel" role="button" data-slide="next">
      <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
      <span class="sr-only">Next</span>
    </a>
  < /div >
< /div >
```

The carousel will look like this:

Figure 6.4: layout-41

6.3.3 Coding the Services Section

This section will contain 3 columns in a row which will then hold content for services. Here, we're also adding some styling and new elements that are not necessarily part of Bootstrap.

```
  <style type="text/css">
  .section-title {
    text-align:center;
    margin-top: 1em;
    background-color: #ecf0f1;
    padding: .5em 0;
    border-radius: .3em;
  }

  .service {
    border-radius: .3em;
    background-color: #34495e;
    text-align: center;
    padding: 3em 2em 1em 2em;
    color: white;
  }

  .service .glyphicon {
    font-size: 5em;
  }
  </style>
```

```
<div class="container">
<h2 class="section-title"><strong>Our Services</strong></h2>
  <div class="row">
    <div class="col-md-4">
      <div class="service">
      <span class="glyphicon glyphicon-cloud" aria-hidden="true"></span>
      <h3><strong>Cloud Storage</strong></h3>
      Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor ↩
          incididunt ut labore et dolore magna aliqua
      < /div >
    < /div >
    <div class="col-md-4">
      <div class="service">
      <span class="glyphicon glyphicon-user" aria-hidden="true"></span>
      <h3><strong>User Dashboard</strong></h3>
      Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor ↩
          incididunt ut labore et dolore magna aliqua
      < /div >
    < /div >
    <div class="col-md-4">
      <div class="service">
      <span class="glyphicon glyphicon-globe" aria-hidden="true"></span>
      <h3><strong>Web Services</strong></h3>
      Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor ↩
          incididunt ut labore et dolore magna aliqua
      < /div >
    < /div >
< /div >
```

The services section would look like this:

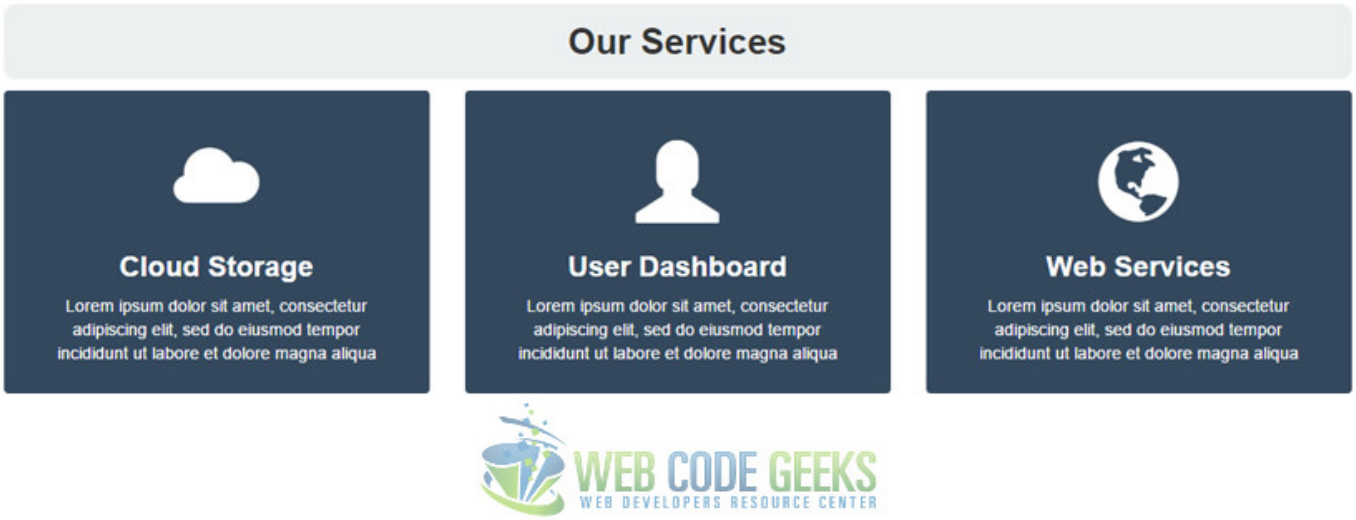


Figure 6.5: layout-51

6.3.4 Coding the Team Section

The team section utilizes bootstrap `.img-circle` class to make images round. The row is divided in 4 columns.

```
<div class="container">
<h2 class="section-title"><strong>Our Team</strong></h2>
  <div class="row">
    <div class="col-md-3" style="text-align:center;">
      <img src="http://goo.gl/1nWwbc" alt="Team" class="img-circle">
      <h3><strong>John Doe</strong></h3>
      <em>Quality Manager</em>
    < /div >
    <div class="col-md-3" style="text-align:center;">
      <img src="http://goo.gl/Q2qUtg" alt="Team" class="img-circle">
      <h3><strong>Alex Graham</strong></h3>
      <em>Web Developer</em>
    < /div >
    <div class="col-md-3" style="text-align:center;">
          <img src="https://goo.gl/xbDwGV" alt="Team" class="img-circle">
          <h3><strong>Rahela Green</strong></h3>
          <em>Graphic Designer</em>
        < /div >
    <div class="col-md-3" style="text-align:center;">
          <img src="https://goo.gl/ivMjuD" alt="Team" class="img-circle">
          <h3><strong>Era Istrefi</strong></h3>
          <em>Marketing</em>
        < /div >
< /div >
```

The team section will look like this in the browser:

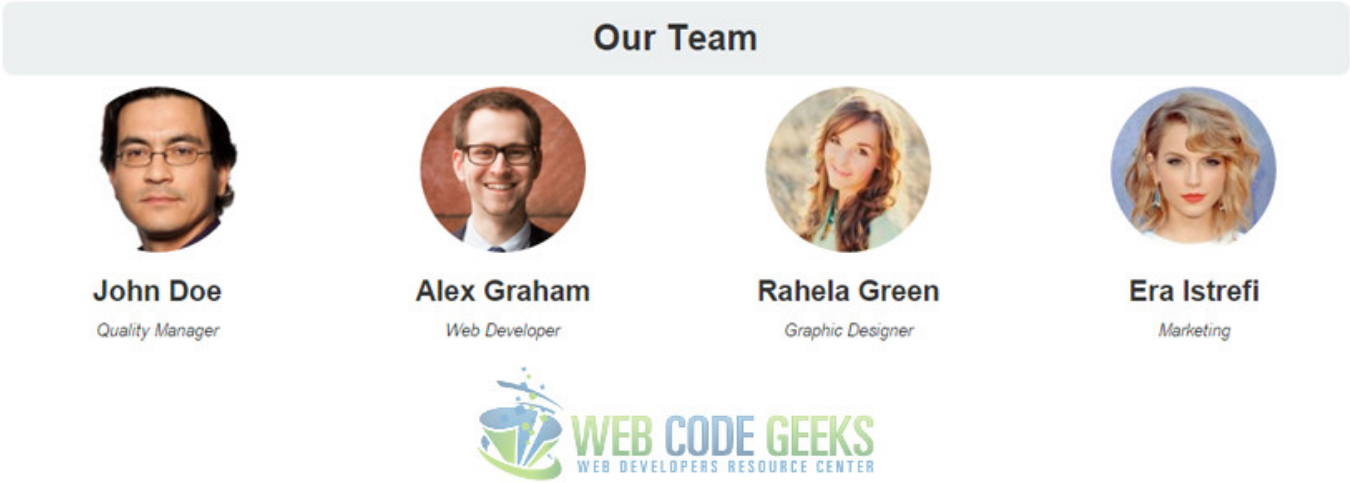


Figure 6.6: layout-6

6.3.5 Coding the Contact Form

This section consists of a contact form:

```
<h2><strong>Contact Us</strong></h2>
<br/>
  <form class="contact-form">
     <div class="container">
       <div class="row">
         <div class="col-sm-4" style="padding-left: 0;">
           <input type="name" class="form-control" placeholder="Name (required)">
         < /div >
         <div class="col-sm-4">
           <input type="email" class="form-control" placeholder="E-Mail (required)">
         < /div >
         <div class="col-sm-4" style="padding-right: 2em;">
           <input type="subject" class="form-control" placeholder="Subject (optional)">
         < /div >
       < /div >
       <br/>
       <div class="row">
         <div class="message" style="padding-right: 2em;"><textarea rows="3" style="max- ↩
             height:8em;"class="form-control" id="message" placeholder="Message (required)"> ↩
             </textarea>< /div >
       < /div >
     < /div >
     <br/>
      <div class="btn btn-default pull-right">Send Message< /div >
     </form>
```

This section would have this view in the browser:

Figure 6.7: layout-7

6.3.6 Coding the Footer Section

Our footer has two columns which will hold quick links and social media respectively:

```
<footer class="footer" style="margin: 5em 0em 1em 0em;">
<div class="container" style="padding-left: 0">
  <div class="row">
    <div class="col-md-6">
    <div class="wrapper" style="background-color:#ecf0f1; padding: 1em 2em;">
      <h3><strong>Quick Links</strong></h3>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Portfolio</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Partners</a></li>
      </ul>
    < /div >
    < /div >
    <div class="col-md-6">
    <div class="wrapper" style="background-color:#ecf0f1; padding: 1em 2em;">
      <h3><strong>Follow Us:</strong></h3>
      <ul>
        <li><a href="#">Facebook</a></li>
        <li><a href="#">Twitter</a></li>
        <li><a href="#">Instagram</a></li>
        <li><a href="#">Youtube</a></li>
      </ul>
    < /div >
    < /div >
  < /div >
  <div class="row" style="text-align: center; padding: 1em; margin: 1em 0; background-color ↩
     :#ecf0f1;">
    All Rights Reserved Â©. Web Code Geeks 2015.
  < /div >
< /div >
</footer>
```

And the last view for this layout would look like this:

Figure 6.8: layout-8

6.4 Conclusion

To conclude, it is important to mention that you have such an infinite number of choices when it comes to designing layouts for your web pages, and layouts vary from page to page depending on the type of content that is going to be put there. Bootstrap helps you code fast by providing a very comprehensive help with the grid system and wide range of styled elements for you to populate your layout. You can see the whole page that we built in the download files.

6.5 Download

Download You can download the full source code of this example here: **Boostrap Layout**

Chapter 7

Tooltip Example

In this example we're going through Bootstrap **tooltips**. The tooltip or infotip or a hint is a common graphical user interface element. It is used in conjunction with a cursor, usually a pointer. The user hovers the pointer over an item, without clicking it, and a tooltip may appear-a small "hover box" with information about the item being hovered over.

In other words, a tooltip is a message which appears when a cursor is positioned over an icon, image, hyperlink, or other element in a graphical user interface. Tooltips do not usually appear on mobile operating systems, because there is no cursor. Bootstrap has quite good support of tooltips and makes them available to everyone to use such styled and animated elements.

7.1 Project Setup

The following requirements need to be met in order to continue adding tooltips with Bootstrap.

7.1.1 Project Folder Setup

Create a new HTML file, which will be your main one, and make sure you have the following folder structure after downloading Boostrap.

Figure 7.1: tooltip-1

7.1.2 Main HTML Setup

Bootstrap already provides a base HTML, which contains links and references to all its' libraries, including a CDN version of jQuery. Your main HTML file should have the following basic syntax inside:

```
<!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">
    <title>Bootstrap Tooltip Example</title>

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

    <!-- 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>
```

7.2 Usage & Examples

In order to use a tooltip in HTML, it must be initialized in Javascript. For performance reasons, the Tooltip and Popover data-apis are opt-in, meaning you must initialize them yourself. One way to initialize all tooltips on a page would be to select them by

their `data-toggle` attribute:

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

In general, the tooltip plugin generates content and markup on demand, and by default places tooltips after their trigger element. A more general way to initialize tooltips is the following:

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

7.2.1 Tooltips on Links

Wouldn't it be great to have some extra information for bits of text (words) or links? This is how you do it in a link:

```
<a href="#" data-toggle="tooltip" title="" data-original-title="Link Tooltip">Link</a>
```

In a more real-world scenario, where a paragraph could have tooltips, we'd have for example:

```
<div class="myText">
    Oat cake jelly beans cake ice cream biscuit.
    <a href="#" data-toggle="tooltip" title="" data-original-title="Here's a Candy Tooltip" ↩
        >Candy</a>
    sesame snaps caramels pudding macaroon marshmallow apple pie cake topping. Halvah
    <a href="#" data-toggle="tooltip" title="" data-original-title="Whoa! A Croissant  ↩
        Tooltip">croissant</a>
    bonbon jelly. Bonbon pastry jelly. Fruitcake lemon drops gummi bears cookie jelly  ↩
        lollipop apple pie chupa chups pastry. Caramels dessert muffin. Chupa chups
    <a href="#" data-toggle="tooltip" title="" data-original-title="Another Sweet Tooltip"> ↩
        sweet</a>
    cotton candy sweet roll candy icing liquorice pudding. Gummies candy canes oat cake  ↩
        pastry.
    < /div >
```

Don't forget to initialize the tooltip like we did in the section above, otherwise no tooltip will appear.

Figure 7.2: tooltip-2

7.2.2 Tooltips on Buttons

Not only can we also set tooltips on buttons, but also place them in a different direction, rather than the random top position. We can do so using the `data-placement` attribute inline along with the other tooltip attributes, like so:

```
<button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="left"  ↩
    title="Tooltip on left">Tooltip on left</button>
<button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="top"  ↩
    title="Tooltip on top">Tooltip on top</button>
<button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="bottom" ↩
     title="Tooltip on bottom">Tooltip on bottom</button>
<button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="right"  ↩
    title="Tooltip on right">Tooltip on right</button>
```

Now you'd see tooltips on all sides of a button:

Figure 7.3: tooltip-3

7.3 Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-*`.

7.3.1 animation

Type: boolean **Default**: true **Description**: Apply a CSS fade transition to the tooltip

7.3.2 container

Type: string **Default**: false **Description**: Appends the tooltip to a specific element. Example: `container:'body'`. This option is particularly useful in that it allows you to position the tooltip in the flow of the document near the triggering element - which will prevent the tooltip from floating away from the triggering element during a window resize.

7.3.3 delay

Type: number/object **Default**: 0 **Description**: Delay showing and hiding the tooltip (ms) - does not apply to manual trigger type. If a number is supplied, delay is applied to both hide/show. Object structure is: delay: `{ "show":500, "hide":100 }`

7.3.4 html

Type: boolean **Default**: false **Description**: Insert HTML into the tooltip. If false, jQuery's text method will be used to insert content into the DOM. Use text if you're worried about XSS attacks.

7.3.5 placement

Type: string/function **Default**: "top" **Description**: How to position the tooltip - top | bottom | left | right | auto. When "auto" is specified, it will dynamically reorient the tooltip. For example, if placement is "auto left", the tooltip will display to the left when possible, otherwise it will display right.

7.3.6 selector

Type: string **Default**: false **Description**: If a selector is provided, tooltip objects will be delegated to the specified targets. In practice, this is used to enable dynamic HTML content to have tooltips added. See this and an informative example.

7.3.7 title

Type: string/function **Default**: **Description**: Default title value if `title` attribute isn't present. If a function is given, it will be called with its this reference set to the element that the tooltip is attached to.

7.3.8 trigger

Type: string **Default**: *hover focus* **Description**: How tooltip is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space. `manual` cannot be combined with any other trigger.

7.4 Methods

Additionally, you can add methods to the tooltip in javascript.

7.4.1 .tooltip(*show*)

Reveals an element's tooltip. **Returns to the caller before the tooltip has actually been shown** (i.e. before the `shown.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip. Tooltips with zero-length titles are never displayed.

```
$('#element').tooltip('show')
```

7.4.2 .tooltip(*hide*)

Hides an element's tooltip. **Returns to the caller before the tooltip has actually been hidden** (i.e. before the `hidden.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip.

```
$('#element').tooltip('hide')
```

7.4.3 .tooltip(*toggle*)

Toggles an element's tooltip. `Returns to the caller before the tooltip has actually been shown or hidden` (i.e. before the `shown.bs.tooltip` or `hidden.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip.

```
$('#element').tooltip('toggle')
```

7.4.4 .tooltip(*destroy*)

Hides and destroys an element's tooltip. Tooltips that use delegation (which are created using the `selector` option) cannot be individually destroyed on descendant trigger elements.

```
$('#element').tooltip('destroy')
```

7.5 Events

The following events can be considered using with the tooltip:

1. `show.bs.tooltip` - This event fires immediately when the `show` instance method is called.
2. `shown.bs.tooltip` - This event is fired when the tooltip has been made visible to the user.
3. `hide.bs.tooltip` - This event is fired immediately when the `hide` instance method has been called.
4. `hidden.bs.tooltip` - This event is fired when the tooltip has finished being hidden from the user (will wait for CSS transitions to complete).
5. `inserted.bs.tooltip` - This event is fired after the `show.bs.tooltip` event when the tooltip template has been added to the DOM.

For example:

```
$('#myOwnTooltip').on('inserted.bs.tooltip', function () {
  // do something...
})
```

7.6 Conclusion

To conclude, tooltips are a simple and nice way to add extra information to a link, button etc. Inspired by the excellent jQuery.tipsy plugin, tooltips are an updated version, which don't rely on images, use CSS3 for animations, and data-attributes for local title storage. Tooltips with zero-length titles are never displayed.

7.7 Download

Download You can download the full source code of this example here: **Boostrap Tooltip**

Chapter 8

Panel Example

The aim of this example is to create and use Bootstrap panels. While not always necessary, sometimes you need to put your DOM in a box. A panel in bootstrap is a bordered box with some padding around its content.

So, why do you need it? It's always necessary to find ways to organize information inside a webpage. It helps maintain user interaction and enhance experience, as well as gives your page a systematic look.

Panels are another simple and clean way of showing content. Panels may have headers and footers, and also include tables or list groups. For more creativity, contextual panels can be used.

8.1 Project Setup

The following requirements need to be met in order to continue adding panels with Bootstrap.

8.1.1 Project Folder Setup

Create a new HTML file, which will be your main one, and make sure you have the following folder structure after downloading Boostrap.

Figure 8.1: panel-1

8.1.2 Main HTML Setup

Bootstrap already provides a base HTML, which contains links and references to all its' libraries, including a CDN version of jQuery. Your main HTML file should have the following basic syntax inside:

```
<!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">
    <title>Bootstrap Panel Example</title>

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

    <!-- 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>
```

8.2 Default Example

Panels are created with the `.panel` class, and content inside the panel has a `.panel-body` class. 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 >
```

The result in the browser would be:

Figure 8.2: panel-2

8.3 Cases and Examples

The following sections introduces a series of examples of how you can customize and use the panel in Bootstrap.

8.3.1 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 Goes Here
  < /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 Goes Here
  < /div >
< /div >
```

The result would be:

Figure 8.3: panel-3

8.3.2 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 >
```

The result would be:

Figure 8.4: panel-4

8.3.3 Contextual Alternatives

Like other components, easily make a panel more meaningful to a particular context by adding any of the contextual state classes.

```
<div class="panel panel-primary">...< /div >
```

```
<div class="panel panel-success">...< /div >
<div class="panel panel-info">...< /div >
<div class="panel panel-warning">...< /div >
<div class="panel panel-danger">...< /div >
```

The contextual classes would result in the following:

Figure 8.5: panel-5

8.3.4 Panels with Tables

Add any non-bordered `.table` within a panel for a seamless design. If there is a `.panel-body`, we add an extra border to the top of the table for separation.

```
<div class="panel panel-default">
  <!-- Default panel contents -->
  <div class="panel-heading">Panel heading< /div >
  <div class="panel-body">
    ...
  < /div >
```

```
  <!-- Table -->
  < table class="table" >
    ...
  < /table >
< /div >
```

The panel with a table in it would look like this:

Panel heading

Muffin carrot cake wafer sesame snaps marzipan jelly beans. Candy oat cake marshmallow. Brownie wafer oat cake liquorice macaroon candy. Danish tootsie roll chocolate bar donut tiramisu candy canes soufflé sweet candy canes. Cake sweet jelly beans brownie. Dessert chocolate bar pastry wafer tart caramels dessert.

#	Name	Age
1	Fabio	21
2	Rahela	32

Figure 8.6: panel-6

8.3.5 Panels with List Groups

In addition to tables, Bootstrap also

```
<div class="panel panel-default">
  <!-- Default panel contents -->
  <div class="panel-heading">Panel heading< /div >
  <div class="panel-body">
    ...
  < /div >

  <!-- List group -->
  <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>
< /div >
```

The panel with list group would look like so:

Figure 8.7: panel-7

8.4 Conclusion

To conclude, panels in Bootstrap are just another tool pre-styled and in harmony with other elements of this framework. It helps you show information in a clean and elegant way, so you don't have to worry about organizing content. Contextual classes add extra design to the panels. All modifications are as easy as 1,2,3 meaning that you just have to add classes to `div` elements.

8.5 Download

Download You can download the full source code of this example here: **Boostrap Panel**

Chapter 9

Popover Example

In this example, we'll be creating and using Bootstrap popovers. The Popover plugin is similar to tooltips, it is a pop-up box that appears when the user clicks on an element. The difference is that the popover can contain much more content. Plugin dependency: Popovers require the tooltip plugin (tooltip.js) to be included in your version of Bootstrap.

They enable you to add small overlays of content, like those on the iPad, to any element for housing secondary information. Here, we're having a look at creating popovers and then looking at what methods or options can be added to this plugin.

9.1 Project Setup

The following requirements need to be met in order to continue adding popovers with Bootstrap.

9.1.1 Project Folder Setup

Create a new HTML file, which will be your main one, and make sure you have the following folder structure after downloading Boostrap.

Figure 9.1: Project Folder Setup

9.1.2 Main HTML Setup

Bootstrap already provides a base HTML, which contains links and references to all its' libraries, including a CDN version of jQuery. Your main HTML file should have the following basic syntax inside:

```
<!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">
    <title>Bootstrap Popover Example</title>

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

    <!-- 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>
```

9.2 Cases and Examples

9.2.1 A Default Popover

The default popover will be shown on the right of a button (which is the element we're adding a popover) and will need initialization like in the following code:

```
<button type="button" class="btn btn-primary" data-toggle="popover" title="Popover title"
    data-content="And here's some amazing content. It's very engaging. Right?">Click to
    toggle popover</button>
```

```
<script type="text/javascript">
   $(function () {
     $('[data-toggle="popover"]').popover()
   })
</script>
```

In the browser, the popover would show up this way:

Figure 9.2: Next Click Dismiss Popover Example

9.2.2 Four Directions on Popovers

Just like tooltips, even popovers can be placed on top, bottom and right, left. This is achieved using the `data-placement` attribute in the HTML element.

```
<button type="button" class="btn btn-default" data-container="body" data-toggle="popover"
    data-placement="left" data-content="Vivamus sagittis lacus vel augue laoreet rutrum
    faucibus.">
  Popover on left
</button>

<button type="button" class="btn btn-default" data-container="body" data-toggle="popover"
    data-placement="top" data-content="Vivamus sagittis lacus vel augue laoreet rutrum
    faucibus.">
  Popover on top
</button>

<button type="button" class="btn btn-default" data-container="body" data-toggle="popover"
    data-placement="bottom" data-content="Vivamus sagittis lacus vel augue laoreet rutrum
    faucibus.">
  Popover on bottom
</button>

<button type="button" class="btn btn-default" data-container="body" data-toggle="popover"
    data-placement="right" data-content="Vivamus sagittis lacus vel augue laoreet rutrum
    faucibus.">
  Popover on right
</button>
```

Similarly, the view would be:

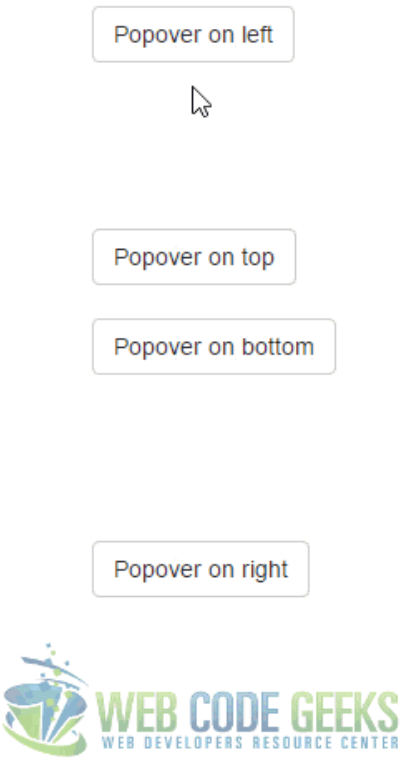


Figure 9.3: Four Directions Popover Example

9.2.3 Dismissable Popover

Use the `focus` trigger to dismiss popovers on the next click that the user makes.

```
<a tabindex="0" class="btn btn-lg btn-danger" role="button" data-toggle="popover" data- ↩
    trigger="focus" title="Dismissible popover" data-content="And here's some amazing  ↩
    content. It's very engaging. Right?">Dismissible popover</a>
```

On the next click, the popover is going to fade out:

Figure 9.4: Default Popover Example

9.3 Options

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

9.3.1 animation

Type: Boolean **Default**: True **Description**: Apply a CSS fade transition to the popover.

9.3.2 container

Type: String **Default**: False **Description**: Appends the popover to a specific element. Example: `container:'body'`. This option is particularly useful in that it allows you to position the popover in the flow of the document near the triggering element - which will prevent the popover from floating away from the triggering element during a window resize.

9.3.3 content

Type: String | Function **Default**: '' **Description**: Default content value if `data-content` attribute isn't present. If a function is given, it will be called with its this reference set to the element that the popover is attached to.

9.3.4 delay

Type: Number | Object **Default**: 0 **Description**: Delay showing and hiding the popover (ms) - does not apply to manual trigger type. If a number is supplied, delay is applied to both hide/show. Object structure is: `delay:{ "show":500, "hide": 100 }`

9.3.5 html

Type: Boolean **Default**: False **Description**: Insert HTML into the popover. If false, jQuery's text method will be used to insert content into the DOM. Use text if you're worried about XSS attacks.

9.3.6 placement

Type: Boolean **Default**: Right **Description**: How to position the popover - top | bottom | left | right | auto. When "auto" is specified, it will dynamically reorient the popover. For example, if placement is "auto left", the popover will display to the left when possible, otherwise it will display right. When a function is used to determine the placement, it is called with the popover DOM node as its first argument and the triggering element DOM node as its second. The `this` context is set to the popover instance.

9.3.7 selector

Type: String **Default**: False **Description**: If a selector is provided, popover objects will be delegated to the specified targets. In practice, this is used to enable dynamic HTML content to have popovers added.

9.3.8 title

Type: String | Function **Default**: '' **Description**: Default title value if `title` attribute isn't present. If a function is given, it will be called with its this reference set to the element that the popover is attached to.

9.3.9 trigger

Type: String **Default**: "click" **Description**: How popover is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space. `manual` cannot be combined with any other trigger.

9.3.10 viewport

Type: String | Object | Function **Default**: { selector: *body*, padding: 0 } **Description**: Keeps the popover within the bounds of this element. Example: `viewport:'#viewport'` or `{ "selector":"#viewport", "padding":0 }` If a function is given, it is called with the triggering element DOM node as its only argument. The this context is set to the popover instance.

9.4 Methods

`$().popover(options)` - Initializes popovers for an element collection.

9.4.1 .popover(*show*)

Reveals an element's popover. **Returns to the caller before the popover has actually been shown** (i.e. before the `shown.bs.popover` event occurs). This is considered a "manual" triggering of the popover. Popovers whose both title and content are zero-length are never displayed.

```
$('#element').popover('show')
```

9.4.2 .popover(*hide*)

Hides an element's popover. **Returns to the caller before the popover has actually been hidden** (i.e. before the `hidden.bs.popover` event occurs). This is considered a "manual" triggering of the popover.

```
$('#element').popover('hide')
```

9.4.3 .popover(*toggle*)

Toggles an element's popover. **Returns to the caller before the popover has actually been shown or hidden** (i.e. before the `shown.bs.popover` or `hidden.bs.popover` event occurs). This is considered a "manual" triggering of the popover.

```
$('#element').popover('toggle')
```

9.4.4 .popover(*destroy*)

Hides and destroys an element's popover. Popovers that use delegation (which are created using the `selector` option) cannot be individually destroyed on descendant trigger elements.

```
$('#element').popover('destroy')
```

9.5 Events

show.bs.popover - This event fires immediately when the `show` instance method is called.

shown.bs.popover - This event is fired when the popover has been made visible to the user (will wait for CSS transitions to complete).

hide.bs.popover - This event is fired immediately when the `hide` instance method has been called.

hidden.bs.popover - This event is fired when the popover has finished being hidden from the user (will wait for CSS transitions to complete).

inserted.bs.popover - This event is fired after the `show.bs.popover` event when the popover template has been added to the DOM.

Example:

```
$('#myPopover').on('hidden.bs.popover', function () {
  // do something...
})
```

9.6 Conclusion

To conclude, I would just like to add that this is yet another elegant way to place content in your web pages, allowing clean interface and quite good design that you would otherwise need to start from scratch.

9.7 Download

Download You can download the full source code of this example here: **Boostrap Popover**

Chapter 10

Tabs Example

The aim of this example is to explain and use Bootstrap Tabs. Tabs are used to separate content into different panes where each pane is viewable one at a time. They enable you to add quick, dynamic tab functionality to transition through panes of local content, even via dropdown menus.

Bootstrap uses CSS to style tabs and content inside them and Javascript to switch to the desired tab whenever you click over it. Tabs are a great way to easily navigate to distinct content without having to scroll or switch web pages.

Many websites nowadays use tabs to organize information better and add animations to tabs to have a better user interaction.

10.1 Project Setup

The following requirements need to be met in order to continue creating tabs with Bootstrap.

10.1.1 Project Folder Setup

Create a new HTML file, which will be your main one, and make sure you have the following folder structure after downloading Boostrap.

Figure 10.1: tabs-1

10.1.2 Main HTML Setup

Bootstrap already provides a base HTML, which contains links and references to all its' libraries, including a CDN version of jQuery. Your main HTML file should have the following basic syntax inside:

```
<!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">
    <title>Bootstrap Tabs Example</title>

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

    <!-- 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>
```

10.2 The Main Example

The default example is the most important one to consider when using tabs. The markup follows a logic way of organizing code. First the markup for the titles of the tabs are created, together with their respective HTML attributes. Then, tab content is added

and one of them is assigned as active to be the first active tab to be shown to the user.

10.2.1 The Markup

```
<ul id="myTabs" class="nav nav-tabs" role="tablist">
  <li role="presentation" class="active"><a href="#home" id="home-tab" role="tab" data- ↩
      toggle="tab" aria-controls="home" aria-expanded="true">Home</a></li>
  <li role="presentation" class=""><a href="#profile" role="tab" id="profile-tab" data- ↩
      toggle="tab" aria-controls="profile" aria-expanded="false">Profile</a></li>
  <li role="presentation" class="dropdown"> <a href="#" id="myTabDrop1" class="dropdown- ↩
      toggle" data-toggle="dropdown" aria-controls="myTabDrop1-contents" aria-expanded=" ↩
      false">Dropdown <span class="caret"></span></a>
  <ul class="dropdown-menu" aria-labelledby="myTabDrop1" id="myTabDrop1-contents">
    <li class=""><a href="#dropdown1" role="tab" id="dropdown1-tab" data-toggle="tab" aria- ↩
        controls="dropdown1" aria-expanded="false">Dropdown1</a></li>
    <li><a href="#dropdown2" role="tab" id="dropdown2-tab" data-toggle="tab" aria-controls= ↩
        "dropdown2" aria-expanded="false">Dropdown2</a></li>
    </ul>
  </li>
</ul>

<div id="myTabContent" class="tab-content">
  <div role="tabpanel" class="tab-pane fade active in" id="home" aria-labelledby="home-tab" ↩
      >
    Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu ↩
        stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg ↩
        carles vegan helvetica. Reprehenderit butcher retro keffiyeh dreamcatcher synth. ↩
        Cosby sweater eu banh mi, qui irure terry richardson ex squid. Aliquip placeat ↩
        salvia cillum iphone. Seitan aliquip quis cardigan american apparel, butcher ↩
        voluptate nisi qui.
  < /div >
  <div role="tabpanel" class="tab-pane fade" id="profile" aria-labelledby="profile-tab">
    Food truck fixie locavore, accusamus mcsweeney's marfa nulla single-origin coffee squid ↩
        . Exercitation +1 labore velit, blog sartorial PBR leggings next level wes anderson ↩
        artisan four loko farm-to-table craft beer twee. Qui photo booth letterpress, ↩
        commodo enim craft beer mlkshk aliquip jean shorts ullamco ad vinyl cillum PBR. Homo ↩
         nostrud organic, assumenda labore aesthetic magna delectus mollit. Keytar helvetica ↩
         VHS salvia yr, vero magna velit sapiente labore stumptown.
    < /div >
  <div role="tabpanel" class="tab-pane fade" id="dropdown1" aria-labelledby="dropdown1-tab" ↩
      >
    Etsy mixtape wayfarers, ethical wes anderson tofu before they sold out mcsweeney's ↩
        organic lomo retro fanny pack lo-fi farm-to-table readymade. Messenger bag gentrify ↩
        pitchfork tattooed craft beer, iphone skateboard locavore carles etsy salvia banksy ↩
        hoodie helvetica. DIY synth PBR banksy irony. Leggings gentrify squid 8-bit cred ↩
        pitchfork. Williamsburg banh mi whatever gluten-free, carles pitchfork biodiesel ↩
        fixie etsy retro mlkshk vice blog. Scenester cred you probably haven't heard of them ↩
        , vinyl craft beer blog stumptown. Pitchfork sustainable tofu synth chambray yr.
    < /div >
  <div role="tabpanel" class="tab-pane fade" id="dropdown2" aria-labelledby="dropdown2-tab" ↩
      >
    Trust fund seitan letterpress, keytar raw denim keffiyeh etsy art party before they ↩
        sold out master cleanse gluten-free squid scenester freegan cosby sweater. Fanny ↩
        pack portland seitan DIY, art party locavore wolf cliche high life echo park Austin. ↩
         Cred vinyl keffiyeh DIY salvia PBR, banh mi before they sold out farm-to-table VHS ↩
        viral locavore cosby sweater. Lomo wolf viral, mustache readymade thundercats ↩
        keffiyeh craft beer marfa ethical. Wolf salvia freegan, sartorial keffiyeh echo park ↩
         vegan.
    < /div >
< /div >
```

10.2.2 The Result

Now the tabs are created, and they look like so:

Figure 10.2: tabs-2

10.3 Methods & Events

10.3.1 Methods

`$().tab` - Activates a tab element and content container. Tab should have either a `data-target` or an `href` targeting a container node in the DOM. In the above examples, the tabs are the <a> with `data-toggle="tab"` attributes.

`.tab('show')` - Selects the given tab and shows its associated content. Any other tab that was previously selected becomes unselected and its associated content is hidden. **Returns to the caller before the tab pane has actually been shown** (i.e. before the `shown.bs.tab` event occurs).

Example:

```
$('#someTab').tab('show')
```

10.3.2 Events

When showing a new tab, the events fire in the following order:

1. `hide.bs.tab` (on the current active tab)
2. `show.bs.tab` (on the to-be-shown tab)
3. `hidden.bs.tab` (on the previous active tab, the same one as for the hide.bs.tab event)
4. `shown.bs.tab` (on the newly-active just-shown tab, the same one as for the show.bs.tab event)

If no tab was already active, then the `hide.bs.tab` and `hidden.bs.tab` events will not be fired.

show.bs.tab - This event fires on tab show, but before the new tab has been shown. Use `event.target` and `event.relatedTarget` to target the active tab and the previous active tab (if available) respectively.

shown.bs.tab - This event fires on tab show after a tab has been shown. Use `event.target` and `event.relatedTarget` to target the active tab and the previous active tab (if available) respectively.

hide.bs.tab - This event fires when a new tab is to be shown (and thus the previous active tab is to be hidden). Use `event.target` and `event.relatedTarget` to target the current active tab and the new soon-to-be-active tab, respectively.

hide.bs.tab - This event fires after a new tab is shown (and thus the previous active tab is hidden). Use `event.target` and `event.relatedTarget` to target the previous active tab and the new active tab, respectively.

Example:

```
$('a[data-toggle="tab"]').on('shown.bs.tab', function (e) {
  e.target // newly activated tab
  e.relatedTarget // previous active tab
})
```

10.4 Conclusion

To conclude, tab based navigations provides an easy and powerful mechanism to handle huge amount of content within a small area through separating content into different panes where each pane is viewable one at a time. The user can quickly access the content through switching between the panes without leaving the page.

10.5 Download

Download You can download the full source code of this example here: **Boostrap Tabs**

Chapter 11

Modal Example

The Bootstrap Modal is a responsive JavaScript popup used for many purposes, such as log in/signup forms, videos, images and alert dialog boxes. It can even be used for showcasing dialog prompts for terms and conditions, and one of it's best features is being responsive and customizable.

Modals' structure:

Theoretically, modals have a three-part structure, formed by a header, a body and a footer section; though you can't really trigger a modal without a handle, which can be a button or a link.

You might find yourself asking: Can't we go on without the handle? The answer is actually a surprising yes, you can. But you won't be able to trigger the modal, which by default stays hidden until triggered, so it's pointless to write the code for it.

In the header section, usually we only place the title of the modal, and maybe even a close button. In the body section we place whatever we wanted to showcase, be it terms and conditions, log in forms, videos, or even a single question. The footer section is where we usually place the button which take the user to the main page, or save changes or anything we want the user to do next.

We will use:: Twitter's Bootstrap

11.1 HTML markup for modals

Linking and scripting

First of all,after creating an HTML file called `index.html` in our project's directory, we have to give a name to our HTML document and link Bootstrap's CSS file and script Bootstrap's JavaScript file. So we put the name between the `<title> </title>` tags, and after it we place this code:

```
<link rel="stylesheet" href="css/bootstrap.min.css">
<script src="js/bootstrap.min.js"></script>
```

Trigger on the document...

Now we can go on with creating our modal. First we create the trigger button, which can be placed anywhere in the document's body.The code will look like this:

```
<!-- Button trigger modal -->
<button class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">
    Launch modal
</button>
```

You may have guessed by now that the class attribute just makes the trigger button look big and blue. You will see there that the button also has two other data attributes which are `data-toggle` and `data-target`. `data-toggle` is the one to tell the button what to toggle when the button is clicked (in our case it's a modal), while `data-trigger` tells it which one to trigger as there can be multiple modals.

Where is my modal?

Time to tell the browser that I'm building a modal right in the place where I'll place the following code snippet:

```
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby=" ↩
    myModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
        < /div >
    < /div >
< /div >
```

So let's explain what we've written there:

The class `modal fade` makes the modal fade into and out of view. The id which in our case has the attribute `myModal` should be the same as the id we placed in the button. You may even add a description to your modal by adding the `aria-describebdy` attribute. Furthermore, I'd like to address the issue of modals disappearing when the backdrop (back overlay area) is clicked. You can make this default property go away by adding `data-backdrop="static"` as an attribute.

The new guys in town are `aria-labelledby` and `aria-view`. These attributes are used to make the modal more accessible (you can learn more about web accessibility here). The other classes you see are self-explanatory.

Let's structure it!

Right after the tag, we place the whole code for the content, which is divided in three parts (remember the structure of a modal?): header, body and footer.

You will find the code for the header looking like this:

```
<!-- Header-->
<div class="modal-header">
        <button type="button" class="close" data-dismiss="modal">
                <span aria-hidden="true">&times;</span>
                <span class="sr-only">Close</span>
        </button>
        <h4 class="modal-title" id="myModalLabel">Modal title</h4>
< /div >
```

While you already understand that the `<h3>` tag contains the title, you also have to know that the button represents the small typical X which closes the modal when clicked. The span element inside the button contains the icon and the hidden description for it's function, when you hover your cursor over it.

On to creating the body. After writing the code for the header, you write the code for the body, which should look something like this:

```
<!-- Body -->
<div class="modal-body">
  Lorem ipsum ...
< /div >
```

The only thing you need to do is create a element with the class `modal-body` and then do your thing.

Just as simple is the code for the footer, which in itself is just a element with a `modal-footer` class, containing two buttons, doing whatever you want them to do. In the following code snippet, the buttons just save changes or close the modal:

```
<!-- Footer -->
<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 >
```

And that's it. You can now see your modal.

But...

11.2 It doesn't work!

If you wrote the code being really careful, or even copy-pasted it and your modal still won't show, then you're as unlucky as me (and many others). There are some cases that the modal just won't show. This can be for two reasons.

The first is that `boostrap.min.js` doesn't work properly without jQuery, which can be solved by adding this part of code before `bootstrap.min.js`:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.1/jquery.min.js"></script>
<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.10.3/jquery-ui.min.js"></script>
```

If it still doesn't work then the problem is with your version of Bootstrap: some versions' `hide` class has the property `display: none;` which makes your modal disappear. To fix that you can instead use the custom class `hide-block` with this code:

```
.hide-block{
    display: none;
}
```

Now your modal is up and running, and looking like this:

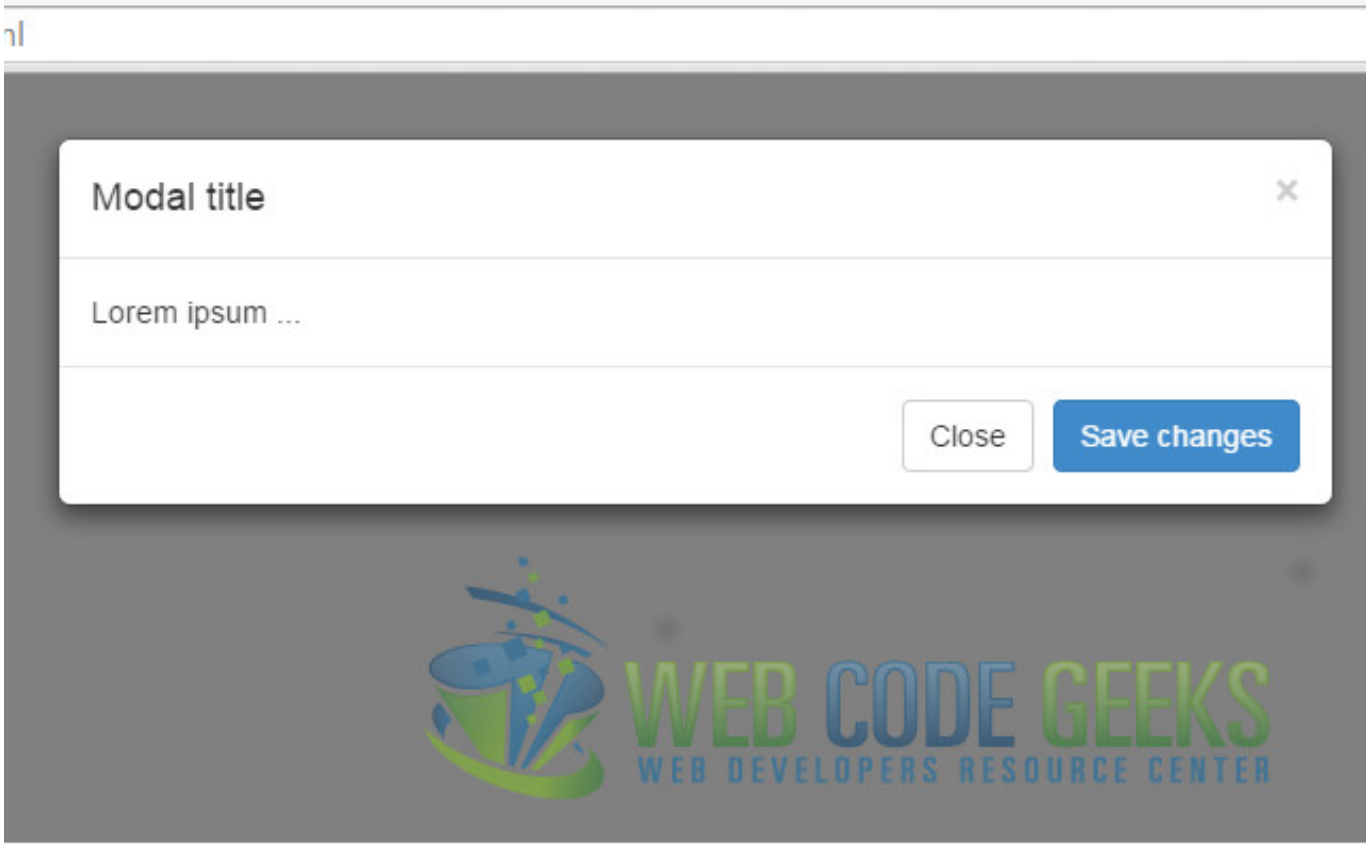


Figure 11.1: Modal created with Twitter's Bootstrap

11.3 Customize your modal

You are now able to create a modal with basic properties, but we'll show you more: CUSTOMIZING. You can make your modal multifunctional and fancy. Here's how:

Changing it's size Modals have two available classes: large and small. You can activate them by placing modifier classes on the `modal-dialog` class. The modifier classes in question would be `modal-lg` and `modal-sm`.

Removing animation By default, modals appear by fading into the view. You can remove this animation by simply removing the `fade` class from your modal markup.

Cool, huh?! Now on to even cooler stuff!

Embedding Youtube videos

We mentioned earlier that you can use modals to showcase videos. If I want to embed a Youtube video, how exactly do I do that? Here's how:

As I'll be building a modal that plays the video as soon as it is triggered, I will be adding this attribute to the launch button:

`data-theVideo="http://www.youtube.com/embed/loFtozxZG0s"`.

Then we place this code in the modal's body:

```
<iframe width="100%" height="350" src=""></iframe>
```

And then we script the function to get and autoplay the video from Datatag, and also the function call like this:

```
<script>
    //function to get and autoplay youtube video from datatag:
    function autoPlayYouTubeModal(){
        var trigger = $("body").find('[data-toggle="modal"]');
        trigger.click(function() {
            var theModal = $(this).data( "target" ),
                    videoSRC = $(this).attr( "data-theVideo" ),
                    videoSRCauto = videoSRC+"?autoplay=1" ;
            $(theModal+' iframe').attr('src', videoSRCauto);
            $(theModal+' button.close').click(function () {
                $(theModal+' iframe').attr('src', videoSRC);
            });
        });
    }
    //the function call:

            $(document).ready(function(){
                autoPlayYouTubeModal();
            });
</script>
```

That is, if you're as lazy as me you can script it like that. If you're not, you can place the code in a different file, and just script the file.This is what you get:

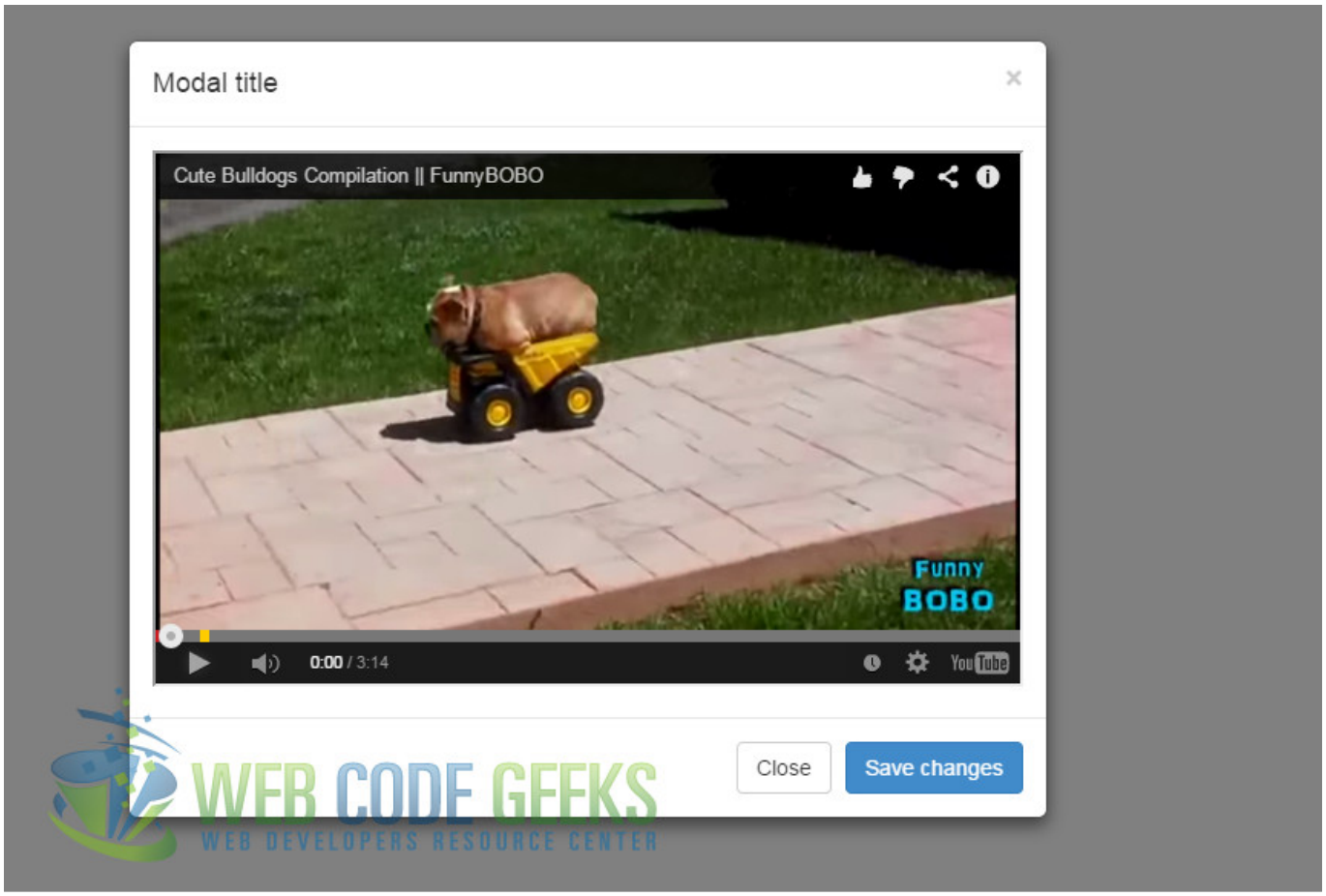


Figure 11.2: YouTube video embedded in modal

Using JavaScript

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

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

Same as with calling modals, you can do a number of actions on modals using JavaScript. Some of the most important are toggle, show and hide.

Now you can go and conquer the Twitter Bootstrap's modal.

11.4 Download

Download You can download the full source code of this example here: **Boostrap Modal**