Developing Responsive Websites with Bootstrap and jQuery

Developing Responsive Websites with Bootstrap and jQuery

© 2017 Aptech Limited
All rights reserved.

No part of this book may be reproduced or copied in any form or by any means – graphic, electronic or mechanical, including photocopying, recording, taping, or storing in information retrieval system or sent or transferred without the prior written permission of copyright owner Aptech Limited.

All trademarks acknowledged.
APTECH LIMITED
Contact E-mail: ov-support@onlinevarsity.com

First Edition - 2017

BE AHEAD OF EVERYONE ELSE

READ ARTICLES

Onlinevarsity

Preface

Developing Responsive Websites with Bootstrap and jQuery course covers in detail, the features of Bootstrap and jQuery supported by the HTML and CSS. The course also explains the responsive Websites and template features of Bootstrap. Example of various Webpages which have successfully and effectively used Boostrap and jQuery are also included. The course explains how to design and develop responsive Websites with Bootstrap and jQuery of various categories.

This book explores Developing Responsive Websites with Bootstrap and jQuery and emphasizes the basic framework of Bootstrap and jQuery library. The course covers basics of Bootstrap and its usage and functions in jQuery. It also briefly explains templates in Bootstrap and various modules in jQuery such as filters and selectors. It further goes on to elaborate on building responsive Websites with the help of Bootstrap and jQuery. By the end of this course, students will be able to create Webpages using bootstrap and responsive content using jQuery.

This book is the result of a concentrated effort of the Design Team, which is continuously striving to bring you the best and the latest in Information Technology. The process of design has been a part of the ISO 9001 Certification for Aptech-IT Division, Education Support Services. As part of Aptech's quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends.

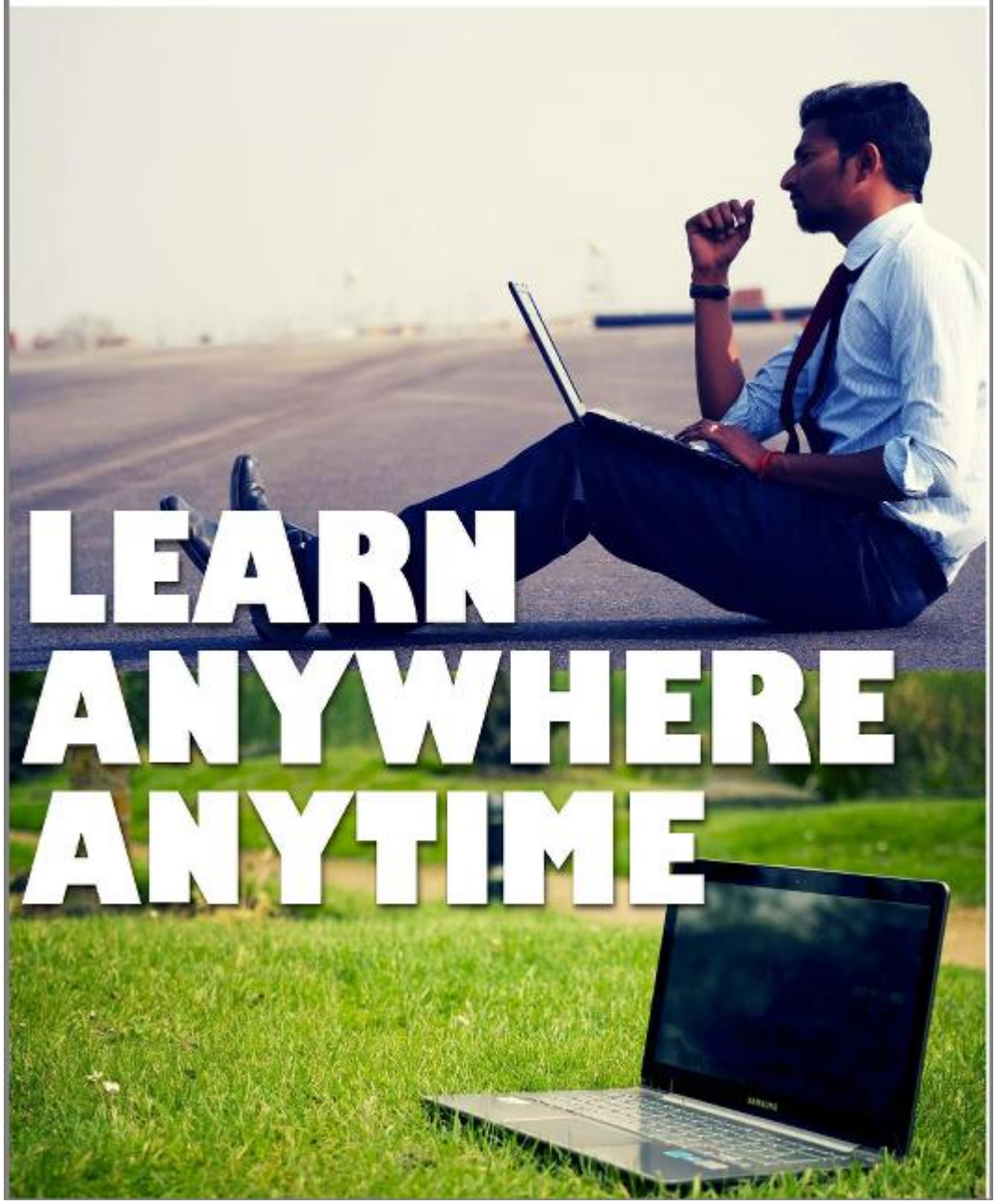
Design Team

Table of Content

Session Name

1. Basics of Bootstrap
2. Understanding Programming in Bootstrap
3. Basics of jQuery
4. Functions, Widgets, Plugins in jQuery
5. Creating Bootstrap Responsive Layout
6. Creating Responsive Webpages Using Bootstrap and jQuery

Onlinevarsity



SESSION 1- Basics of Bootstrap

Session Objectives:

This session introduces the Bootstrap framework along with its basic concepts. It also describes the process of setting up the framework, the file structure of its precompiled version, and the default source code template. The session also briefly explains the use of different Bootstrap templates that are modifiable by changing their Cascading Style Sheet (CSS) and JavaScript (JS) code.

At the end of this session, you will be able to:

- *Describe Bootstrap*
- *Explain the basic concepts of Bootstrap*
- *Describe the process of setting up Bootstrap*
- *Outline the file structure and the default source code in Bootstrap*
- *Use Bootstrap with CSS and JS*
- *Describe the significance of Bootstrap templates*

1.1 Introduction to Bootstrap

Bootstrap is an intuitive and open-source framework that makes front-end development of Web applications simpler as well as quicker. So, it is also known as the front-end framework. Primarily, Bootstrap aims to make responsive Web applications as well as Websites. In fact, it is presently the most popular CSS framework for building responsive Websites.

The framework encompasses JavaScript, CSS, and HTML for developing custom components while designing Webpages. Developed by the Twitter team, Bootstrap also supports HTML5 and CSS3. It includes a free set of tools for creating Web applications and Websites.

Bootstrap consists of CSS and HTML based templates for designing a variety of components such as forms, layout, navigation, and even JavaScript extensions. It also includes a set of defined CSS classes that Web developers can use directly. The framework also uses JavaScript/jQuery framework for creating effects, styles, and actions for its components. In short, bootstrap is a CSS framework, a templating system, and a JavaScript toolkit.

Web developers use CSS for applying styles to the elements of a Webpage by creating as well as assigning classes to these elements. However, Bootstrap automates this styling task by facilitating a developer to include the Bootstrap files and state the predefined class names for the elements. This also eliminates the need of writing custom CSS classes for styling the elements. Thus, Bootstrap saves a lot of time while designing Webpages.

1.2 Benefits of Bootstrap

There are many reasons why Web designers and developers prefer Bootstrap. Following are these reasons:

- **Easy and Flexible Enough to Get Started**: Using Bootstrap is extremely easy, adaptable, and quick. Web developers can use it along with CSS, dynamic stylesheet language (Less), or Syntactically Awesome Style Sheets (Sass). Both Less and Sass are style sheet languages. While Less is compiled into CSS, Sass is interpreted into CSS. Further, there are different ways of adding Bootstrap to a Web project, which shows its flexibility.

- **100% Responsive Designs**: A Website should be responsive to each Internet-enabled device. For example, it should adapt to the screen of a smartphone or tablet without compromising its smooth appearance and pleasant navigation. This is exactly what Bootstrap ensures through its fluid grid layout that readjusts itself dynamically to the screen resolution. The ready-made Bootstrap classes enable identifying the number of spots in the grid that each column should use. This helps in identifying points at which the columns should load horizontally rather than vertically for displaying the requested page accurately on the targeted smartphone. Bootstrap 3 comes with the mobile-first approach by offering mobile-first styles.
- **Great Grid System**: Bootstrap implements a 12-column, responsive grid that supports offset and nested elements.
- **Better Development Speed**: Rather than coding from scratch, Bootstrap allows using ready-made blocks of code for designing a Webpage. Merging these blocks with the code for cross-browser compatibility and CSS-Less functionality can save several hours of coding. Even for backend developers who need to do some user interface modifications, Bootstrap is efficient. This is because there is no need for them to know HTML and CSS well for using Bootstrap.
- **Browser Compatibility**: Bootstrap resolves several cross-browser compatibility issues. This means a Web application made using Bootstrap runs smoothly in Chrome, Safari, Internet Explorer, and Firefox.
- **Customization**: It is possible to use Bootstrap as per the distinct design requirements. Web developers can select the required aspects by ticking off those that are not required. These aspects are available on the Bootstrap customize page, which includes common CSS, components, JavaScript components, and utilities. This is the custom version of Bootstrap, which is ready for download.
- **Consistency**: Initially, a few Twitter team members expanded the framework for improving the consistency across the internal tools. However, later, Mark Otto who is the Co-founder released the first open-source Bootstrap version after comprehending the framework's potential. He also depicted how the framework expanded through the single core concept of combining designers and developers.
- **Big Support**: There is a big supportive community for Bootstrap due to which Web developers can obtain any kind of help at any time. Moreover, the makers frequently update the framework. Currently, it is hosted and updated on GitHub with over 500 contributors. The open-source framework is available at no cost.
- **Bundled JavaScript Components**: Bootstrap comes with a set of JavaScript components for using different functionalities for user interface components. A few examples of these functionalities include tooltips, alerts, and modal windows. This means that there is no need to write scripts for these functionalities.
- **Simple Integration**: Web developers can integrate Bootstrap with other frameworks and platforms, even for the existing Websites. Some Bootstrap elements are usable with an existing CSS.
- **Extensive Pre-styled Components**: Bootstrap comes with a variety of pre-styled components for drop-downs, alerts, and navbars.

1.3 Setting Up Bootstrap

It is easy to download, install, and set up Bootstrap for getting started. There are two ways to start using the framework on a Website, which are:

- Downloading from getbootstrap.com
- Including it from a Content Delivery Network (CDN)

1.3.1 Downloading from GetBootstrap.com

The framework's latest version, 3.3.7, is available at getbootstrap.com. Figure 1.1 shows its home page.

Figure 1.1: Home Page of Getbootstrap.com

Clicking the **Download Bootstrap** button redirects to the **Download** page, as shown in figure 1.2.

Figure 1.2: Download Page

The **Download** page displays the following options, which are the different ways of downloading the framework:

- **Download Bootstrap**: Enables downloading the precompiled, mini versions of Bootstrap CSS, fonts, and JavaScript. It does not provide original files of source code or documentation. The precompiled one facilitates drag-n-drop usage in any project. It includes the compiled CSS and JS files, minified compiled versions of each, optional Bootstrap theme, and Glyphicon fonts.
- **Download Source**: Enables obtaining the latest documentation and JavaScript source code from GitHub. This is the source code version that encompasses the precompiled CSS and JS files, font files, and Bootstrap Less assets. It is a more comprehensive version with a higher learning core than the precompiled version.
- **Download Sass**: Enables obtaining the latest Bootstrap version imported to Sass from LESS for Sass-only projects.

1.3.2 Including from a CDN

Alternatively, Web developers can include Bootstrap from a CDN. This is the option for those developers who do not intend to host the framework by themselves.

A CDN, as a system of distributed servers, delivers Webpages to a user as per the geographic location of the server and the user. The closer the server, the faster is the content sent to the user. CDN aims at accelerating the delivery of Web content for sites with global reach and high traffic. So, using Bootstrap CDN provides the benefit of faster loading time.

1.4 File Structure and Source Code in Bootstrap

After downloading the precompiled Bootstrap version, a ZIP file contains all the directories and files. This arrangement logically clusters the common resources and offers the minified as well as the compiled variations. After extracting the ZIP file, its contents are visible in the form of structured css, js, and fonts directories.

Figure 1.3 shows the file structure of precompiled Bootstrap version.

```
bootstrap/
+-- 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
```

Figure 1.3: Precompiled Bootstrap File Structure

Figure 1.3 shows the most basic form of the framework, which has compiled files.

These files enable a Web developer to use the desired components and plugins quickly by dragging and dropping them onto the required positions on a Webpage of any Web project. It reveals compiled CSS and JS files (**bootstrap.***) along with minified and compiled CSS and JS (**bootstrap.min.***). CSS source maps in the form **bootstrap.*.map** are available for a few browser developer tools.

The structure also includes fonts from Glyphicons and the optional Bootstrap theme. It is worth knowing that JavaScript plugins require the inclusion of jQuery.

1.4.1 Inside the Bootstrap File Structure

In Bootstrap, HTML, CSS, JS, and other components are grouped into categories, which are noticeable atop its documentation. Following are the categories:

- **Scaffolding**: Offers a basic grid structure, simple layouts, background, and link styles.
- **CSS**: Offers global CSS settings, advanced grid system, and styles for basic HTML elements.
- **Components**: Offer styles for a variety of reusable components such as alerts, tabs, page headers, and alerts.
- **JavaScript Plugins**: Offers custom and interactive jQuery plugins for components such as modals, tooltips, and popovers.

The categories of Components and JavaScript plugins together offer the following user interface elements:

- Button groups and button drop-downs
- Page headers and hero unit
- Navbar
- Navigational tabs, pills, and lists
- Labels
- Thumbnails
- Badges
- Drop-downs
- Tooltips
- Alerts
- Modals
- Progress bars

1.4.2 Default Bootstrap Template

Figure 1.4 shows the basic HTML template for Bootstrap, which includes everything that the file structure specifies.

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <!-- The above 3 meta tags *must* come first in the head;-->
8      <title>Bootstrap Basic Template</title>
9
10     <!-- Bootstrap -->
11     <link href="css/bootstrap.min.css" rel="stylesheet">
12
13     <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
14     <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
15     <!--[if lt IE 9]>
16       <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
17       <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
18     <![endif]-->
19   </head>
20   <body>
21     <h1>Hello, world!</h1>
22
23     <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
24     <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
25     <!-- Include all compiled plugins (below), or include individual files as needed -->
26     <script src="js/bootstrap.min.js"></script>
27   </body>
28 </html>
```

Figure 1.4: Basic Bootstrap Template

In figure 1.4, the template includes the **bootstrap.min.css, jquery.min.js**, and **bootstrap.min.js** files. This converts the normal HTML file into a Bootstrapped one. It is essential to include the jQuery library prior to the Bootstrap library. The template also includes **html5shiv.min.js** and **respond.min.js** files for displaying HTML5 elements in Internet Explorer (IE) 8.

This HTML page is responsive, as it has the `viewport` meta tag in the `<head>` element. This tag takes care of rendering and touch zooming on the Internet-enabled mobile gadgets. The `width` property ensures that the page fits into the screen of the target device. By setting its value as `device-width`, the page is rendered properly across various devices such as tablets, desktops, and smartphones. The `initial-scale = 1.0` property renders the page at a 1:1 scale at the time of loading.

Figure 1.5 shows the output of the default Bootstrap template.

Figure 1.5: Basic Bootstrap Template Output

MaxCDN also offers the desired JS and CSS files for working with Bootstrap. Code Snippet 1 shows the different links to be added under the `head` tag for using Bootstrap through MaxCDN.

Code Snippet 1:

```
<!-- Compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.
min.css">

<!-- Optional theme -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-
theme.min.css">

<!-- Compiled and minified jQuery -->
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.j
s"></script>

<!-- Compiled and minified JavaScript -->
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.mi
n.js"></script>
```

In the snippet, the links are referring to the MaxCDN, a public distributed network hosting the desired CSS and JS files. The first link is of the compiled and minified version of CSS. The second one is optional and links to a Bootstrap theme. The third link is of the compiled and minified JS.

Code Snippet 2 shows how to add the required links under the `head` tag.

Code Snippet 2:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap through CDN</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1">

   <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.
min.css">

  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.j
s"></script>

  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.mi
n.js"></script>

</head>
```

In the snippet, the order of linking is important. The reference to minified version of jQuery (**jquery.min.js**) will always come before the minimized version of JS (**bootstrap.min.js**).

1.5 Bootstrap with CSS and JavaScript

Bootstrap is a robust front-end framework to quickly develop a functional Website. To give a unique and interactive look and feel to the site, it is essential to write JavaScript and CSS.

1.5.1 Using CSS

Bootstrap uses a few HTML elements and CSS properties that need the HTML5 doctype. It is essential to include it at the beginning of any Web project. Code Snippet 3 shows how to include the doctype.

Code Snippet 3:

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

While Bootstrap 2 came with the discretionary mobile-friendly styles, Bootstrap 3 is mobile-friendly right from scratch. Rather than including the mobile styles, the version 3 framework have them right at its core. Undeniably, Bootstrap is mobile first, which means the mobile-friendly styles are present throughout the library rather than in isolated files.

For ensuring perfect rendering and zooming on touch, a Web developer should include the `viewport` meta tag in `<head>`, as shown in figure 1.4. It is possible to disable zooming on any mobile device by including `user-scalable=no` to this tag. This allows users to scroll and disables zooming, which gives a feel of a native application. However, experts do not recommend disabling zooming for each site.

Bootstrap allows specifying the standard global display, link styles, and typography. These styles are present in **scaffolding.less**.

In Bootstrap, all site contents and a grid system exist within a container, which is a containing element. There are two containers namely, fixed and fluid. They are non-nestable, which means it is not possible to include a container within another container.

Fixed Width Responsive Container

The fixed container has a preset width in pixels with respect to the whole page's layout (viewport). It remains the same across different screens and browsers. This means that the container's width does not change with a change in the resolution.

For example, in a viewport with a width of 960 pixels, each column has a preset width that does not change. Now, if the user shrinks the browser to 959 pixels, the container snaps to a new layout as per a media query having a maximum width of 768 px. The columns remain unchanged when the width of the browser is between 768 and 960 pixels.

Such a layout is easier to use and personalize in terms of design. It also eliminates the need to specify the minimum and maximum widths, which some browsers do not support.

To create a fixed container, a Web developer needs to use the `.container` class in `<div>`.

Figure 1.6 shows how to create fixed with responsive container.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Bootstrap Container</title>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
   <h1>Bootstrap Container Page</h1>
   <p>This part is inside a .container class.</p>

</div>
</body>
</html>
```

Figure 1.6: Fixed With Responsive Container Code

In figure 1.6, the `<div>` tag contains the `container` class to indicate that its content will come in a fixed-width container.

Figure 1.7 shows the output.

Figure 1.7: Fixed Responsive Container

In figure 1.7, it is noticeable that the content inside the fixed width container is aligned in the middle.

Fluid Width Responsive Container

This container spans across the width of the whole viewport. In simple words, it stretches to cover the available width for fitting to the window regardless of how wide it gets. The viewport's width changes even if there is a smallest change in the size of the browser or screen. It continuously resizes as the browser's width changes such that there is no additional empty space on the sides, unlike a fixed width container.

The width is usually a percentage of the window instead of fixed number of pixels. For example, with 100% layout's width, each column adjusts its width as per the computed relative size (for instance, 25%) when the browser window resizes. Thus, fluid layouts stretch to contract as per the screen's width.

To create a fluid container, a Web developer should use the `.container-fluid` class in `<div>`.

Figure 1.8 shows how to create fluid width responsive container.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Bootstrap Container</title>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container-fluid">
   <h1>My Bootstrap Page</h1>
   <p>This part is inside a .container-fluid class.</p>

</div>
</body>
</html>
```

Figure 1.8: Fluid Responsive Container Code

In figure 1.8, the `<div>` tag contains the `container-fluid` class to indicate that its content will come in a fluid container.

Figure 1.9 shows the output.

Figure 1.9: Fluid Responsive Container

When the user resizes the browsers window, the content within the container adjusts itself while the width remains stretched across the viewport. A fluid design adjusts to the user's screen settings. It also eliminates the need to include a horizontal scroll bar in case of a smaller resolution of the screen, unlike a fixed width container.

It is unwise to modify the bootstrap.css file, as such a modification would invite complications while upgrading Bootstrap. Thus, the best way to edit bootstrap CSS is to make a custom css file and overwrite it whenever there is a different requirement. For example, if the default top bar is in black color but the requirement is gray color, just create a specific selector for this bar and apply the color rule. For instance, it can be `<table class="zig-zag mycustomclass">`. Declaring the custom css file just after bootstrap.css overwrites that part of code that needs a change.

1.5.2 Using JavaScript

Web developers can include JavaScript plugins in one of the following ways:

- Individually, using the ***.js** files.
- All at once, using the **bootstrap.js** or **bootstrap.min.js** file, as each of them have all plugins.

There are two ways of using JavaScript plugins namely, through data attributes and via JavaScript.

Data Attributes

Web developers can utilize all Bootstrap plugins via the markup Application Programming Interface (API).

This eliminates the need to write JavaScript. This is the first-class API of Bootstrap and is considered initially for using a JavaScript plugin.

In a few scenarios, it might be necessary to turn off this functionality. Code Snippet 4 shows how to disable the data attribute API.

Code Snippet 4:

```
$('body').off('.data-api')
```

Note: Through data attributes, the developer can use only a single plugin per element.

The example unbinds all events on the body associated with the namespace, `data-api`. On the other hand, to turn the functionality off for a specific plugin, a Web developer should include the plugin's name before the `data-api` namespace. Code Snippet 5 shows how to include the alert plugin's name before the namespace.

Code Snippet 5:

```
$('body').off('.alert.data-api')
```

Now, let's see how to use data attributes for displaying a modal. A modal is a restructured yet flexible dialog box with least required functionality and wise defaults. It is a special window that may contain a header, body, tooltip, video, and/or a popover for boosting the user experience.

Figure 1.10 shows the code for displaying a modal in a Webpage by using data attributes.

```
1  <body>
2
3  <div class="container">
4    <h2>Demo Modal Example</h2>
5    <!-- Trigger the modal with a button -->
6    <button type="button" class="btn btn-info btn-lg" data-toggle="modal"
7     data-target="#myModal">Open Modal</button>
8    <!-- Modal -->
9    <div class="modal fade" id="myModal" role="dialog">
10     <div class="modal-dialog">
11
12       <!-- Modal content-->
13       <div class="modal-content">
14         <div class="modal-header">
15           <button type="button" class="close" data-dismiss="modal">&times;</button>
16           <h4 class="modal-title">Modal Header</h4>
17         </div>
18         <div class="modal-body">
19           <p>This is my Demo Modal.</p>
20         </div>
21         <div class="modal-footer">
22           <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
23         </div>
24       </div>
25
26     </div>
27   </div>
28
29 </div>
30
31 </body>
```

Figure 1.10: Code for Displaying Modal Using Data Attributes

In figure 1.10, the following steps take place:

- In step 6, the `Open Modal` button is set on the Webpage. Clicking it displays a modal window. The `btn btn-info btn-lg` CSS class sets the large style for this button.

- The `data-toggle="modal"` data attribute is a custom HTML5 attribute that conveys Bootstrap what to do, which is to display a modal. The `data-target` data attribute conveys Bootstrap which element to open. In this case, it is a modal, as its value points to an element having an `id` attribute with value as `myModal`.
- In step 9, the `modal fade` CSS class sets the modal layout.
- In step 10, the `modal-dialog` class defines the dialog layout of the modal.
- In step 13, the class `modal-content` defines what the modal will contain.
- In step 14, the `modal-header` class defines the style of the modal's header.
- In step 15, the `class="close"` property defines the style of the close button in the modal window. The `data-dismiss` data attribute is a custom HTML5 attribute, which is specified for closing the modal window.
- In step 16, the modal-title class sets the title of the modal.
- In step 18, the `modal-body` Bootstrap CSS class sets the style for the modal's body.
- Similarly in step 21, the `modal-footer` Bootstrap CSS class sets the style for the modal's footer.
- In step 22, the CSS classes namely, `btn` and `btn-default`, specify the style for button in the modal's footer.

Figure 1.11 shows the output.

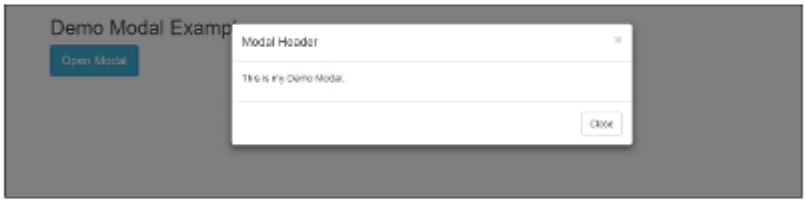


Figure 1.11: Modal with Header, Body, and Footer

Programmatic API

Another way to use Bootstrap plugins is via JavaScript API. Any public API is a single, chainable method that returns a collection. A method should receive an optional object called **options**, which is a string targeting a specific method. It can even receive nothing, which triggers a plugin with a default behavior. Code Snippet 6 illustrates both the statements.

Code Snippet 6:

```
$("#newModal").modal('show') // show is the string targeting the
show method to display the modal dialog immediately.

$("#newModal").modal()  // This statement triggers Default
initialization.
```

The only difference between `modal('show')` and `modal()` is that the former manually opens a modal. It also returns to the invoker prior to raising the relevant event, `shown.bs.modal` for showing the modal.

Figure 1.12 shows how to display a modal by using modal ('show').

```
<!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>Example of Bootstrap 3 Modals</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
 integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmB/rxW/wm3RYgoXbPxhcD8/kNXxZGI2Dtk7" crossorigin="anonymous">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRUmgx6BWfZzTqG+HW7uzfz" crossorigin="anonymous">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity=
"sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUfHRa4Hq+RF2HzOttqBZJGixP/RK4BX+" crossorigin="anonymous"></script>
<script type="text/javascript">
    $(document).ready(function(){
        $("myModal").modal('show');
    });
</script>
</head>
<body>
<div id="myModal" class="modal fade">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
                <h4 class="modal-title">Confirmation</h4>
            </div>
            <div class="modal-body">
                <p>Do you want to save changes you made to document before closing?</p>
                <p class="text-warning"><small>If you don't save, your changes will be lost.</small></p>
            </div>
          <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                <button type="button" class="btn btn-primary">Save changes</button>
            </div>
        </div>
    </div>
</div>
</body>
</html>
```

Figure 1.12: JavaScript Code for Displaying a Modal

In figure 1.12, the MaxCDN links for Bootstrap contain the `integrity` and `crossorigin` attributes. Both ensure that an obtained resource was sent without unfavourable manipulation. The `integrity` attribute enables a browser to check the source and prevent its loading if the source is manipulated. This kind of security is essential, as the CDN is a public network on which any malicious user can manipulate the files on it. The `crossorigin` attribute set to anonymous conveys the browser to execute the specified CDN link without transferring the user credentials. The `anonymous` keyword indicates that there is no need of exchanging the credentials through HTTP authentication, security certificates, or cookies.

In the script tag, `document.ready()` indicates that the HTML page is ready for changes through JavaScript. Through it, JavaScript loads a modal through `modal.show()`, which is also invoked for closing the modal when the user clicks the `Close` button.

Figure 1.13 shows the output.

Figure 1.13: Output of JavaScript Code for Displaying a Modal

Code Snippet 7 shows how to initialize the modal plugin through the default method shown in Code Snippet 6.

Code Snippet 7:

```
<script>
$(document).ready(function(){
    $("#myBtn").click(function(){
        $("#newModal").modal();
    });
});
</script>
```

In this snippet, when a user clicks the button whose id is `#myBtn`, the modal plugin is initialized by invoking its default method.

In short, there are two ways of activating a modal and other JavaScript plugins, which are as follows:

- Through data attributes, as shown in figure 1.10, wherein `data-toggle="modal"` is set on the button controller and `data-target="myModal"` is set to toggle a particular modal
- Through JavaScript by initializing a modal with its id or invoking the `show()` method.

Each Bootstrap plugin also has a raw constructor, which is invoked through its `Constructor` property. Code Snippet 8 shows how to use this property.

Code Snippet 8:

```
$.fn.popover.Constructor
```

For retrieving a specific plugin instance, it is possible to do so directly from an element. Code Snippet 9 shows how to do so.

Code Snippet 9:

```
$('[rel=popover]').data('popover')
```

Events

Bootstrap offers custom events to handle most distinct actions of plugins. Usually, they are available in the infinitive and past participle form. The infinitive form is invoked in the beginning of an event, while the completion of the action triggers its past participle form. For example, the `show` event is triggered in the beginning, while the `shown` event is invoked once the action is completed.

An infinitive event comes with the `preventDefault` functionality for stopping the action's execution prior to its beginning. Code Snippet 10 shows how to use this functionality.

Code Snippet 10:

```
$('#newModal').on('show', function (e)
{
   if (!data) return e.preventDefault() //stops modal from display
})
```

Transitions

Web developers should include Transition.js for applying a few basic transition effects. This is not needed if they are using the compiled **bootstrap.js**, as the file is already included. **Transition.js** is a helper file that other plugins use to verify the CSS transition support. Developers can use the plugin for achieving the sliding and fading effects for modals, tabs, and alerts.

Figure 1.14 illustrates how to use the JavaScript API in Bootstrap to inform a user that something went wrong after clicking the `Submit` button on a Webpage.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Bootstrap Example</title>
5    <meta charset="utf-8">
6    <meta name="viewport" content="width=device-width, initial-scale=1">
7    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
8    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
9    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10   <script>
11    $(document).ready(function () {
12        $('#btnSubmit').click(function () {
13            $('#myAlert').show('fade');
14        });
15        $('#linkClose').click(function () {
16            $('#myAlert').hide('fade');
17        });
18    });
19 </script>
20 </head>
21 <body>
22 <div class="container">
23   <button id="btnSubmit" class="btn btn-primary">
24     Submit
25 </button>
26
27  <br /><br />
28 <div id="myAlert" class="alert alert-danger collapse">
29     <a id="linkClose" href="#" class="close">&times;</a>
30     <strong>Error!</strong> There is a problem submitting your form
31 </div>
32 </body>
33 </html>
```

Figure 1.14: JavaScript Code for Displaying an Alert Message

In figure 1.14, the JavaScript code uses the `id` selector, `btnSubmit`, to find the button. As soon as the button is clicked, the corresponding JavaScript function is triggered to display an alert box. This box displays an error message. When a user clicks the `Close` link, the alert box hides itself by fading away.

Figure 1.15 shows the output.

Figure 1.15: Output of JavaScript Code for Displaying an Alert Message

1.6 Templates in Bootstrap

A bootstrap template refers to a mock Website that is designed using the bootstrap framework. Its components are JS code, CSS code, HTML pages, fonts, and different images belonging to its design. The template is usually designed for accomplishing a purpose, which can be to have an e-commerce site, a corporate portal, a blog, or simply a vertical landing page.

Figure 1.16 shows a Bootstrap template for a blog.

Figure 1.16: Blog Bootstrap Template

Figure 1.17 shows a Bootstrap template for a business frontpage.

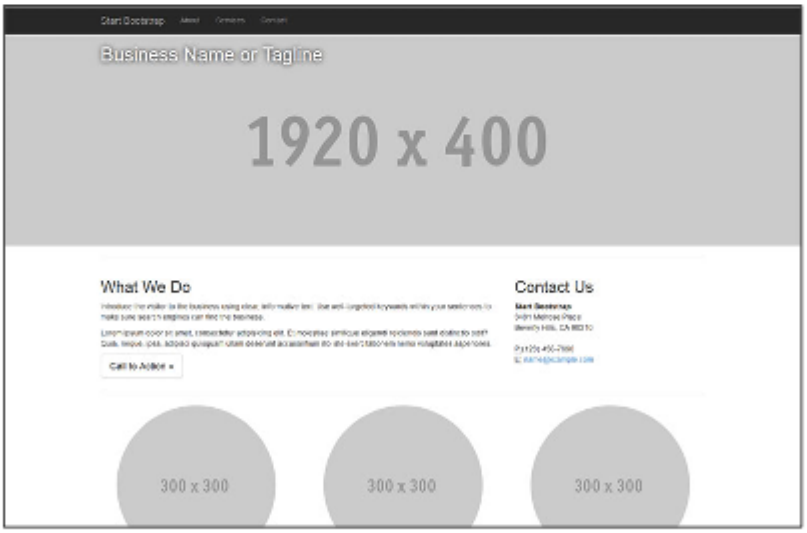


Figure 1.17: Business Frontpage Template

Figure 1.18 shows a Bootstrap template for shopping item.

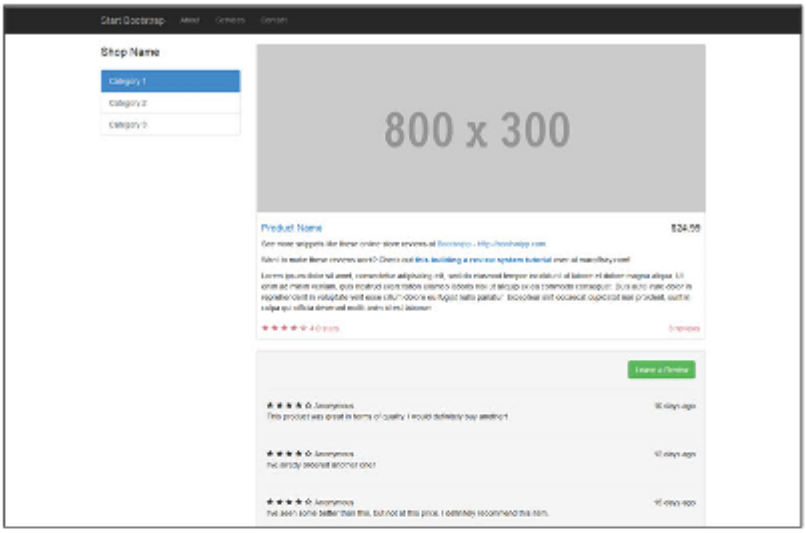


Figure 1.18: Shop Item Template

As per the purpose, Web designers try to find and include all relevant templates. For instance, if templates are required for an e-commerce Website, they may find a product category page, a cart page, a contact us page, and a customer testimonial page. While there are a couple of more templates for such a Website, it is rational to pick only those that are required.

Bootstrap templates are also available in different versions. For adapting the template to the targeted site or purpose by easily changing its content, it is essential to have some basic knowledge about HTML, CSS, and JS. All Bootstrap Web templates are responsive, easy to customize, and ready-to-use. They are reusable due to which Web developers can save a lot of time in designing some of the most common Webpages.

Quick Test 1.1

1. It is possible to disable scrolling on a Bootstrap Webpage.
   a. True
   b. False

2. It is easier to display a modal window without JavaScript.
   a. True
   b. False

Summary

- Bootstrap is an open-source Web framework for quickly developing front-end and responsive applications.
- Bootstrap uses CSS, HTML, JavaScript, and jQuery design interactive Webpages.
- Customization, browser compatibility, fully responsive pages, and an assortment of built-in components are the major benefits of Bootstrap.
- There are two ways to start using Bootstrap namely, downloading its files from getbootstrap.com or including them from a CDN.
- Structure of the precompiled Bootstrap version includes the css, js, and fonts directories offering components and plugins for drag-n-drop usage.
- The basic HTML Bootstrapped template includes **bootstrap.min.css**, **jquery.min.js**, **bootstrap.min.js**, and the jQuery library prior to the Bootstrap library.
- The default Bootstrapped template contains the `viewport` meta tag to make a Webpage responsive.
- Bootstrap 3 is mobile first, which means the mobile-friendly styles are present throughout the library rather than in isolated files.
- Bootstrap comes with two types of containers namely, fixed width and full width containers.
- There are two ways of using JavaScript plugins namely, through data attributes and via JavaScript.
- Bootstrap template are sample Webpages designed for using the bootstrap framework.

Check Your Progress

1. **Which of the following Bootstrap statements is true?**
   a. It is hard to get started.
   b. It is a templating system.
   c. It is a new version of JS.
   d. All of these
2. **Which of the following options allows downloading Bootstrap without the source code or documentation?**
   a. Precompiled
   b. Source
   c. Saas
   d. None of these
3. **Which of the following file naming conventions indicates minified and compiled files in the Bootstrap structure?**
   a. Bootstrap.*.min
   b. Bootstrap.min.*
   c. Bootstrapmin.*
   d. Bootstrap.min*
4. **Which of the following properties ensures that the page shall fit into the screen of the target device?**
   a. Viewport
   b. Initial-scale
   c. Device-width
   d. Width
5. **Which of the following contains two different form of verbs for handling plugin actions?**
   a. Events
   b. Data attributes
   c. Programmatic API
   d. Only HTML components

Answers to Check Your Progress

1. It is a templating system.
2. Precompiled
3. Bootstrap.min.*
4. Width
5. Events

Answers to Quick Test

Quick Test 1.1

1. False
2. True

SESSION 2 – Understanding Programming in Bootstrap

Session Objectives:

This session introduces different types of components available in Bootstrap. It also explains how to use these components and how to apply Cascading Style Sheet (CSS) and JavaScript (JS) to them. The session also briefly explains how to start coding in Bootstrap by modifying its default template.

At the end of this session, you will be able to:

- *Use different Bootstrap components*
- *Write programs in Bootstrap*
- *Apply CSS and JavaScript to Bootstrap components*

2.1 Components in Bootstrap

The modular Bootstrap framework encompasses a collection of Less stylesheets that utilize its different components. Bootstrap compiles them into a bundle for adding to Webpages although individual sections are removable or replaceable. The stylesheets offer standard style definitions for all major HyperText Markup Language (HTML) components. They offer an even and stylish look by enabling formatting of text and forms.

There are a set of configuration variables that regulate the

visible aspects of components such as padding and color. From Bootstrap 2, a customization wizard creates a tailored version of Bootstrap, as per the desired components and its different settings.

Each Bootstrap component comprises an HTML structure, CSS code, and even associated JavaScript, if required. In addition to the standard HTML components, Bootstrap offers other widely used interface components. Web developers implement these components as CSS classes that are applicable to some HTML elements in a Webpage.

Bootstrap also provides a variety of JavaScript components, which are available as jQuery plugins. These components offer additional elements such as carousels, modals, tooltips, alerts, popovers, drop-downs, and dialog boxes. They also extend the functionality of a few prevalent elements. For example, an input field is now available with an auto-complete function.

Bootstrap 3 supports the latest Firefox, Opera, Chrome, Safari (not on Windows), and Internet Explorer (IE) versions. It also supports the IE8 and the latest Extended Support Release (ESR) of Firefox. In all these browsers, the Bootstrap components render themselves smoothly without affecting consistency in appearance.

2.2 Layout Components

In Bootstrap, there are dozens of layout components all of which are reusable. They include glyphicons, drop-downs, button groups, navbar, breadcrumbs, labels, badges, pagination, input groups, progress bars, and panels.

2.2.1 Glyphicons

There are more than 250 glyphs or icon fonts in Bootstrap. They are available in the font format in **Glyphicons Halflings** files of the **font** folder. Although not usually available free of cost, the set is available for free usage through Bootstrap. Any of these icons needs a base as well as an individual icon class. Following is the syntax for inserting a glyphicon:

```
<span class="glyphicon glyphicon-<name>"></span>
```

where,

- `glyphicon`: Is the base class.
- `glyphicon-name`: Is the individual icon class. It uses the Unicode font reference in Glyphicons Halflings for rendering the icon on a Webpage. For example, the Unicode font for envelope icon is 2709.
- `name`: Is to be replaced with the exact glyphicon name.

A Web developer should use an icon class only on elements that have neither text nor child elements. Similarly, it is impossible to combine icon classes directly with other components. Further, Web developers should not use them with other classes for the same element. Rather, it is recommended using the icon classes within a nested `<span>` tag.

Figure 2.1 shows the code for displaying envelope and search glyphicons.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Bootstrap Example</title>
5    <meta charset="utf-8">
6    <meta name="viewport" content="width=device-width, initial-scale=1">
7    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
8    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
9    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10 </head>
11 <body>
12
13 <div class="container">
14   <h2>Some Glyphicons</h2>
15   <p>Envelope icon: <span class="glyphicon glyphicon-envelope"></span></p>
16   <p>Hyperlinked Envelope:
17     <a href="#"><span class="glyphicon glyphicon-envelope"></span></a>
18   </p>
19   <p>Search: <span class="glyphicon glyphicon-search"></span></p>
20 </div>
21
22 </body>
23 </html>
```

Figure 2.1: Code for Envelope and Search Glyphicons

In figure 2.1, there is a space between the text and `span` tag specifying the appropriate glyphicon. This is essential for putting space between the text and its glyphicon while displaying them in the browser. Figure 2.2 shows these glyphicons.

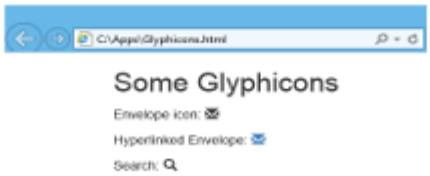


Figure 2.2: Envelope and Search Glyphicons

For the users of assistive technologies, there are a few things to consider. The `aria-hidden` attribute should be set to `true` to prevent any kind of unwanted or confusing output by a screen reader. This is essential if the icons are just for decorative purpose. This is because the latest assistive technology version of a reader tends to read both Unicode characters and CSS generated content. Therefore, it can result in a confusing output due to which the attribute is used to hide the characters.

In case of a component using only an icon and nothing else, it is rational to offer alternative content to state the purpose of that component for the screen reader users. To do so, just add the `arial-label` attribute to the element itself.

Example 2.1 shows the usage of `aria-label` and `aria-hidden` attributes.

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

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

Example 2.1: Usage of `aria-label` and `aria-hidden` Attributes

If there is an icon that reflects the desired text graphically, it is essential to ensure that a screen reader conveys this text. For example, a Web developer can consider using the `.sr-only` class for hiding the additional content on a Webpage but making it available for a screen reader to read.

Figure 2.3 shows the code for using the `sr-only` class to convey an error message.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4     <title>Bootstrap Example</title>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
8     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
9     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10 </head>
11 <body>
12
13 <div class="container">
14  <div class="alert alert-danger" role="alert">
15    <span class="glyphicon glyphicon-exclamation-sign" aria-hidden="true"></span>
16    <span class="sr-only">Error:</span>
17    Enter digits only
18 </div>
19
20 </body>
21 </html>
```

Figure 2.3: Code for Indicating an Error Message via the `.sr-only` Class

In figure 2.3, the icon of exclamation sign is used to indicate an error message. The `sr-only` class conveys this error to the users of screen readers.

Figure 2.4 shows the output.

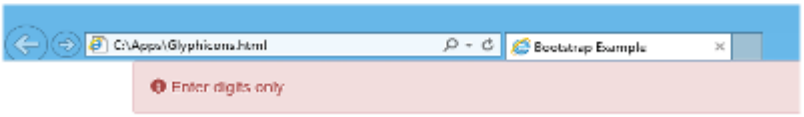


Figure 2.4: Error Message

2.2.2 Navbars

A navbar or a navigation bar is a responsive navigation header visible on the top of a Webpage. In Bootstrap, it is collapsible or extensible, as per the size of the targeted screen. A navbar can contain links, drop-downs, buttons, icons, or a combination of all. A Web developer can align these elements to left or right. A navbar is set horizontally when the existing width of the viewport increases.

Figure 2.5 shows the code for a navbar having three links namely, Home, About us, and Contact us.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4     <title>Bootstrap Case</title>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device
7     <link rel="stylesheet" href="https://maxcdn
8     <script src="https://ajax.googleapis.com/a
9     <script src="https://maxcdn.bootstrapcdn.co
10 </head>
11 <body>
12
13 <nav class="navbar navbar-default">
14   <div class="container-fluid">
15     <div class="navbar-header">
16       <a class="navbar-brand" href="#">WebSite
17     </div>
18     <ul class="nav navbar-nav">
19       <li class="active"><a href="#">Home</a>
20       <li><a href="#">Home</a></li>
21       <li><a href="#">About us</a></li>
22       <li><a href="#">Contact us</a></li>
23     </ul>
24   </div>
25 </nav>
26
27 <div class="container">
28    <h3>Basic Navbar</h3>
29    <p>A navigation bar is a navigation header
30 </div>
31
32 </body>
33 </html>
```

Note: If JavaScript is disabled in the user's browser and the viewport is so narrow that a navbar collapses, it is unfeasible to expand the navbar or see the content in its collapsed state.

Figure 2.5: Navbar Code

In figure 2.5, `<nav class="navbar navbar-default">` creates a standard navigation bar. The `.navbar-brand` class is used to display your organization's name. The `.navbar-nav` class is used for a light and full-height navigation and provides classes for adding links and drop-downs to an unordered list. The `active` class indicates that the **Home** page is the current Webpage displayed in the browser.

Figure 2.6 shows the output.

Figure 2.6: Default Navbar

Bootstrap cannot recognize the amount of space the content requires in a navbar. Therefore, it might trigger some content wrapping issues in the second row. To avoid them, a Web developer can reduce the width or number of navbar items.

Note: It is impossible to add several form controls in an input group.

For the users of assistive tools, Web developers must use `<nav>` or `role="navigation"` for each navbar in case of a generic element such as `<div>`. This explicitly recognizes that section for those users.

2.2.3 Input Groups

Bootstrap supports input, textarea, radio, select (even with multiple select), and check box form controls. It also supports all input types of HTML5, which are:

- Text
- Number
- Password
- E-mail
- Datetime, datetime-local, week, time, month, and date
- URL
- Search
- Color

Example 2.2 shows how to create text and password input controls using the `.form-control` class.

```
<div class="form-group">
<label for="user">Name:</label>
<input type="text" class="form-control" id="user">
</div>
<div class="form-group">
<label for="pwd">Password:</label>
<input type="password" class="form-control" id="pwd">
</div>
```

Example 2.2: Using the `form-control` Class

In example 2.2, the `form-group` class adds some space between controls.

Input groups are referred to as extended form controls using which it is easy to append or prefix icons, buttons, or text to an input field. For example, you can prefix an input field taking price in Dollars with a Dollar icon.

Bootstrap provides the `.input-group` container class for improving an input by adding help text, button, or an icon in front or at the rear. It also offers the `.input-group-addon` class that affixes text, icon, or button to the input field.

Figure 2.7 shows the code for using these input group classes.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>
<form>
    <div class="input-group">
      <span class="input-group-addon">
      <i class="glyphicon glyphicon-user"></i></span>
      <input id="email" type="text" class="form-control"
     name="email"
      placeholder="Enter E-mail.">
    </div>
    <div class="input-group">
      <span class="input-group-addon">
      <i class="glyphicon glyphicon-lock"></i></span>
      <input id="password" type="password" class="form-control"
     name="password"
      placeholder="Enter password.">
    </div>
    <br>
    <div class="input-group">
      <span class="input-group-addon">Text</span>
      <input id="msg" type="text" class="form-control" name="msg"
      placeholder="Enter more details, if any.">
    </div>
    <br>
    <div class="input-group">
    <input type="text" class="form-control" placeholder="Enter what
     you wish to search.">
    <div class="input-group-btn">
      <button class="btn btn-default" type="submit">
       <i class="glyphicon glyphicon-search"></i>
      </button>
    </div>
  </form>
</body>
</html>
```

Figure 2.7: Code for Input Groups

In the example, the code for user and lock glyphicons is added before the `<input>` element so that the icons become visible before the input element. However, the code for search glyphicon exists after the element so that the search icon appears after the element.

Figure 2.8 shows these input controls.

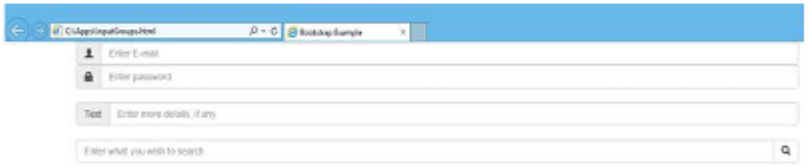


Figure 2.8: Input Groups

2.2.4 Pagination

In case of a Website with many pages, adding blocks of page numbers or pagination component is useful. To add this component, a Web developer should use the `.pagination` class in the `<ul>` element.

Figure 2.9 shows the code for adding the default pagination component.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <h2>Pagination</h2>
  <p>The .pagination class provides pagination links:</p>
  <ul class="pagination">
    <li><a href="#">1</a></li>
    <li><a href="#">2</a></li>
    <li><a href="#">3</a></li>
    <li><a href="#">4</a></li>
    <li><a href="#">5</a></li>
    <li><a href="#">6</a></li>
  </ul>
</div>

</body>
</html>
```

Figure 2.9: Pagination Code

Figure 2.10 shows the output.

Figure 2.10: Pagination

2.2.5 Page Headers

The page header component is similar to the `<H1>` tag. It acts as a section splitter and adds proper spacing to the Webpage headings. This is useful for distinctly showing each title on a Webpage in case there are many titles. To add a page header, a Web developer needs to wrap the title within `<div>` and use the `.page-header` class.

Figure 2.11 shows the code for including a page header.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="page-header">
  <h1>Example page header <small>Subtext for header</small></h1>
</div>

</body>
</html>
```

Figure 2.11: Page Header Code

In figure 2.11, the component uses the `small` element (default) of `<H1>` to display text in small case next to the header text.

Figure 2.12 shows the output.

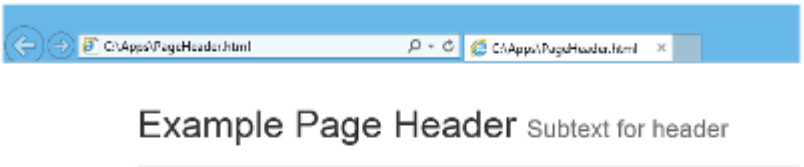


Figure 2.12: Page Header

2.2.6 Badges

In Bootstrap, badges are special labels indicating new items, which are similar to number of unread e-mails near the Inbox tab in an e-mail account. They show numerical values that indicate the number of items related to a link. For creating badges, a Web developer should use the `.badge` class within `<span>` or other elements such as buttons and list groups.

Figure 2.13 shows the code for creating badges, wherein `2`, `11`, and `12` are badges.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4     <title>Bootstrap Example</title>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
8     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
9     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10 </head>
11 <body>
12
13 <div class="container">
14    <h2>Badges</h2>
15    <a href="#">Hello <span class="badge">2</span></a><br>
16    <a href="#">World <span class="badge">11</span></a><br>
17    <a href="#">Folks <span class="badge">12</span></a>
18 </div>
19
20 </body>
21 </html>
```

Figure 2.13: Badges Code

Figure 2.14 shows the output.

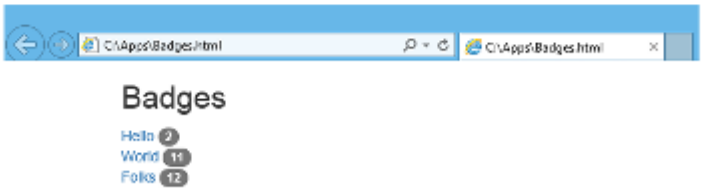


Figure 2.14: Badges

2.2.7 Panels

A panel refers to a container that can hold other components such as lists and tables with proper padding. The `.panel` base class is used to create a panel with standard padding and border, while the `.panel-body` class is used to place the content inside it.

Figure 2.15 shows the code for creating a panel.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Bootstrap Case</title>
5    <meta charset="utf-8">
6    <meta name="viewport" content="width=device-width, initial-scale=1">
7    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
8    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
9    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10 </head>
11 <body>
12
13 <div class="container">
14   <h2>Demo Panel</h2>
15   <div class="panel panel-default">
16     <div class="panel-body">A Basic Panel</div>
17   </div>
18 </div>
19
20 </body>
21 </html>
```

Figure 2.15: Panel Code

In figure 2.15, the `.panel-default` sub-class retains the default color style for the panel.

Figure 2.16 shows the output.

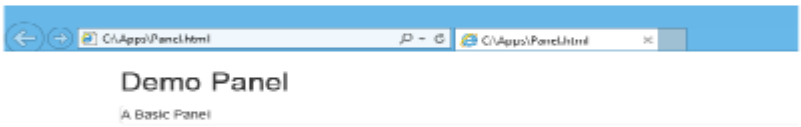


Figure 2.16: Panel

2.2.8 List Groups

A list group renders an unordered list of items. Apart from this standard use, the component also renders complex and tailored content. To create a standard list group, a Web developer should include the `.list-group` class within `<ul>` and the `.list-group-item` class within `<li>`.

Figure 2.17 shows the code for creating a list group with badges on the right.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <h2>List Group With Badges</h2>
  <ul class="list-group">
    <li class="list-group-item">Hello <span class="badge">2</span></li>
    <li class="list-group-item">World <span class="badge">15</span></li>
    <li class="list-group-item">Folks <span class="badge">13</span></li>
  </ul>
</div>

</body>
</html>
```

Figure 2.17: List Group Code

Figure 2.18 shows the output.

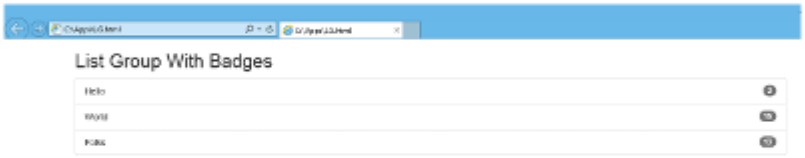


Figure 2.18: List Group with Badges

2.2.9 Progress Bars

A progress bar shows the status of an action or a process in progress, such as downloading and uploading a file. It can even indicate a feedback or convey that an action regarding the Webpage elements is occurring. Bootstrap offers different types of progress bars, that are as follows:

- Default
- Striped
- Animated
- Stacked

Figure 2.19 shows some of these progress bars.

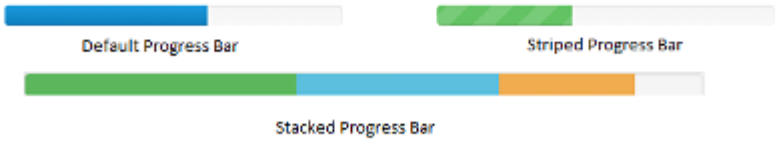


Figure 2.19: Different Progress Bars in Bootstrap

To generate a progress bar, a Web developer needs to use the `.progress` class in `<div>`.

Figure 2.20 shows the code for including a progress bar with a label showing the percentage of progress.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Bootstrap Example</title>
5    <meta charset="utf-8">
6    <meta name="viewport" content="width=device-wi
7    <link rel="stylesheet" href="https://maxcdn.boo
8    <script src="https://ajax.googleapis.com/ajax/l
9    <script src="https://maxcdn.bootstrapcdn.com/bo
10 </head>
11 <body>
12
13 <div class="container">
14   <h2>Progress Bar With Label</h2>
15   <div class="progress">
16     <div class="progress-bar" role="progressbar"
17      aria-valuemax="100" style="width:89%">
18       89%
19     </div>
20   </div>
21 </div>
22 </body>
23 </html>
```

Note: JavaScript can also be a part of the head section. However, it is ideal to do so. A browser loads a Webpage step wise from top to bottom. If JavaScript is present in the head section, it may slow down the loading process.

Figure 2.20: Progress Bar Code

Figure 2.21 shows the output.

Figure 2.21: Progress Bar with a Percentage Label

Progress bars in Bootstrap implement CSS3 animations as well as transitions for a few effects.

2.3 Programming in Bootstrap

After setting up the environment for Bootstrap, a Web developer can start programming in Bootstrap. CSS forms the core of Bootstrap and always exists in the head section, while JavaScript comes at the end of the body section. In Bootstrap, JavaScript also depends on jQuery. Thus, programming in Bootstrap consists of the following three main parts:

- Bootstrap CSS in the head section
- Bootstrap JavaScript at the end of the body
- jQuery at the end of the body

Note: While Opera 12 has no support for animations at all, older Firefox versions and Internet Explorer 9 and below do not support these effects.

Figure 2.22 shows the basic template, which acts as a foundation for writing Bootstrap code.

```
1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <meta charset="utf-8">
5       <meta http-equiv="X-UA-Compatible" content="I
6       <meta name="viewport" content="width=device-
7       <!-- The above 3 meta tags *must* come first
8        any other head content must come *after* the
9       <title>Bootstrap Basic Template</title>
10      <!-- Bootstrap -->
11      <link href="css/bootstrap.min.css" rel="style
12
13      <!-- HTML5 shim and Respond.js for IE8 suppo
14      <!-- WARNING: Respond.js doesn't work if you
15      <!--[if lt IE 9]>
16        <script src="https://oss.maxcdn.com/html5s
17        <script src="https://oss.maxcdn.com/respond
18      <![endif]-->
19    </head>
20    <body>
21      <h1>Hello, world!</h1>
22
23      <!-- jQuery (necessary for Bootstrap's JavaS
24      <script src="https://ajax.googleapis.com/aja
25      <!-- Include all compiled plugins (below), o
26      <script src="js/bootstrap.min.js"></script>
27    </body>
28  </html>
```

Figure 2.22: Basic Bootstrap Template for Programming

Note: : For an <a> element, a Web developer should use the `href` attribute rather than `data-target`.

In figure 2.22, the **html5shiv.js** and **respond.min.js** files are added so that IE 8 can render the HTML5 elements. A Web developer needs to modify this template by adding the desired CSS and JavaScript code for creating the intended Website.

To start programming in Bootstrap, let's modify this template to add a collapsible. A collapsible is useful for showing or hiding a huge amount of content.

Figure 2.23 shows the code for adding collapsible content through Bootstrap code.

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4     <meta name="viewport" content="width=device-width, initial-scale=1">
5     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
6     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
7     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
8   </head>
9   <body>
10
11  <div class="container">
12    <h2>Collapsible Example</h2>
13    <button type="button" class="btn btn-info" data-toggle="collapse" id="myBtn" data-target="#demo">Collapsible</button>
14    <div id="demo" class="collapse">
15      Collapsibles are useful when you want to hide and show large amount of content
16    </div>
17  </div>
18
19  </body>
20  </html>
```

Figure 2.23: Bootstrap Code for Adding Collapsible Content

In figure 2.23, `class="collapse"` marks a collapsible content, which is . This content is displayed or hidden when a user clicks the button of type `info`, which is created using the `btn-`

`info` class. To set the suitable toggle behavior for showing and hiding the content, the value of the `data-toggle` attribute is set to `collapse`. The `data-target` attribute links the button with the corresponding collapsible element.

Figure 2.24 shows the output.

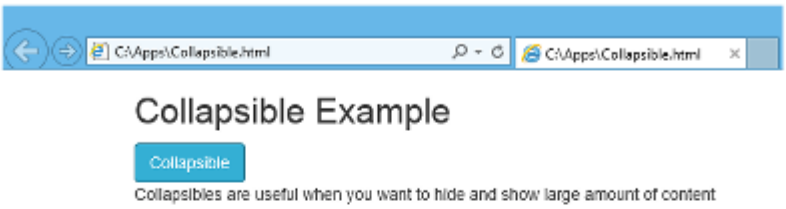


Figure 2.24: Collapsible Content

By default, the collapsible content is hidden. When a user clicks the button, the content becomes visible. Upon clicking the button once more, the content hides, as the `<div>` tag collapses. To show the content by default, a Web developer should use `class=collapse.in` in the `<div>` tag.

Figure 2.25 shows how to use JavaScript to create collapsible content.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4     <meta name="viewport" content="width=device-width, initia
5     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.
6     <script src="https://ajax.googleapis.com/ajax/libs/jquer
7     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.
8  </head>
9  <body>
10
11 <div class="container">
12    <h2>Collapsible Example</h2>
13    <button type="button" class="btn btn-info" id="myBtn">Coll
14    <div id="demo" class="collapse">
15      Collapsibles are useful when you want to hide and show
16    </div>
17 </div>
18
19 <script>
20 $(document).ready(function(){
21    $("#myBtn").click(function(){
22      $('.collapse').collapse();
23    });
24 });
25 </script>
26 </body>
27 </html>
```

Note: The Bootstrap version in use should include the transition.js (plugin) for proper functioning of collapsible.

Figure 2.25: JavaScript Code for Collapsible Content

In figure 2.25, appropriate JavaScript code in the `<body>` tag specifies what should happen on the click of the button. The `$('.collapse').collapse();` statement internally uses the collapse plugin to automatically provide the control over `<div>`.

2.4 Demo on Bootstrap Functionality

Let's see a sample Bootstrap application by designing an HTML form. Styling the controls in an HTML form one by one with manual CSS code is often tedious and subject to errors. Bootstrap eliminates tediousness and risk of errors by offering a predefined collection of classes for styling and aligning the form controls.

For forms, Bootstrap offers three layouts, which are as follows:

- Vertical
- Horizontal
- Inline

Irrespective of a layout, a Web developer needs to wrap each element in `<div>` that uses the `.form-group` class. This is required for optimal spacing. Further, it is necessary to use the `.form-control` class in the `<input>`, `<textarea>`, and `<select>` elements.

2.4.1 Vertical Forms

A vertical layout is the default layout in Bootstrap for forms. In this layout, styles are specified for controls without a base class. The controls are piled with labels that are alighed to the left.

Figure 2.26 shows the code for creating a vertical form having two input fields, two labels, a check box, and a button. The vertical layout is the default form layout in Bootstrap.

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4     <title>Bootstrap Example</title>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
8     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
9     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10  </head>
11  <body>
12
13  <div class="container">
14    <form>
15      <div class="form-group">
16        <label for="email">User Email:</label>
17        <input type="email" class="form-control" id="email" placeholder="Enter email">
18      </div>
19      <div class="form-group">
20        <label for="pwd">Password:</label>
21        <input type="password" class="form-control" id="pwd" placeholder="Enter password">
22      </div>
23      <div class="checkbox">
24        <label><input type="checkbox"> Remember me</label>
25      </div>
26      <button type="submit" class="btn btn-default">Submit</button>
27    </form>
28  </div>
29
30  </body>
31  </html>
```

Figure 2.26: Code for Vertical Form with Controls

Figure 2.27 shows the output.

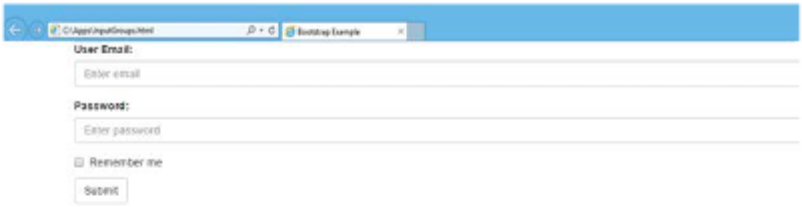


Figure 2.27: Vertical Form with Controls

At times, a Web developer may need to arrange the controls side-by-side for squeezing the layout vertically. This is when an inline form layout is useful, which is likely to suit the device's screen. In an inline form, all elements are in line, one after the other and are left-aligned. This is relevant to forms displayed within a viewport whose width is minimum 768 pixels. To create an inline form, a Web developer needs to include the `.form-inline` class in the `<form>` element.

Figure 2.28 shows the code for an inline form containing two labels, two input fields, a check box and a button.

```
1  <body>
2
3  <div class="container">
4    <h2>Inline form</h2>
5    <form class="form-inline">
6      <div class="form-group">
7        <label for="email">Email:</label>
8        <input type="email" class="form-contro
9      </div>
10     <div class="form-group">
11       <label for="pwd">Password:</label>
12       <input type="password" class="form-con
13     </div>
14     <div class="checkbox">
15       <label><input type="checkbox"> Remembe
16     </div>
17     <button type="submit" class="btn btn-def
18   </form>
19 </div>
20
21 </body>
```

Note: The Bootstrap version in use should include the tooltip.js (plugin) for proper functioning of popovers.

Figure 2.28: Code for Inline Form with Controls

Figure 2.29 shows the output.

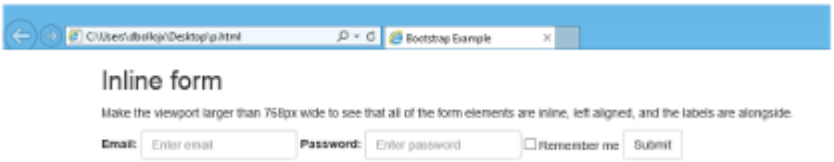


Figure 2.29: Inline Form with Controls

Now, let's see how to implement jQuery in Bootstrap for creating a popover. A popover is analogous to a tooltip. It refers to a pop-up dialog box appearing on clicking an element. However, unlike a tooltip, a popover can display much more content.

Figure 2.30 shows the CSS and jQuery code for creating a popover.

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4     <title>Bootstrap Example</title>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
8     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
9     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10  </head>
11  <body>
12
13  <div class="container">
14    <h3>Javascript/JQuery Example</h3>
15    <a href="#" data-toggle="popover" title="Popover Header" data-content="Hi there!!">Toggle popover</a>
16  </div>
17
18  <script>
19  $(document).ready(function(){
20      $('[data-toggle="popover"]').popover();
21  });
22  </script>
23
24  </body>
25  </html>
```

Figure 2.30: Popover Code

In figure 2.30, `data-toggle="popover"` initializes a popover for the `Toggle popover` link. The title attribute specifies the popover's header text, while the `data-content` attribute holds the text that the popover should display in its body.

A popover is not CSS-only plugin due to which it is essential to initialize it through jQuery. So, in figure 2.30, the jQuery statement, `$('[data-toggle="popover"]').popover()`, selects the element that has the `data-toggle` property set to `popover` and invokes the `popover()` method.

Figure 2.31 shows the output when a user clicks the `Toggle popover` link.

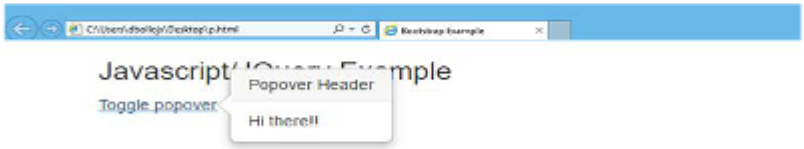


Figure 2.31: Popover Display

Quick Test 2.1

1. A Web developer can use an icon class on the label element.
   a. True
   b. False

2. It is ideal to have JavaScript in the `body` section.
   a. True
   b. False

Summary

- Each Bootstrap component is reusable and comprises an HTML structure, CSS code, and even associated JavaScript.
- Bootstrap provides a variety of JavaScript components, which are available as jQuery plugins.
- There are more than 250 glyphs in Bootstrap, which are available in **Glyphicons Halflings** files.
- An icon class is applicable to elements that have neither text nor child elements.
- A navbar is a responsive navigation header visible on the top of a Webpage and is collapsible or extensible.
- Pagination refers to a block of numbers, each indicating a different page of a Website.
- Badges are special labels indicating new items, which is similar to the number of unread e-mails near the Inbox tab in an e-mail account.
- A panel refers to a container that can hold other components such as lists and tables with proper padding.
- A list group renders an unordered list of items as well as complex and tailored content.
- Bootstrap offers four main types of progress bars namely, default, striped, animated, and stacked.
- Programming in Bootstrap includes CSS in the `head` section and JavaScript/jQuery in the `body` section.
- The **html5shiv.js** and **respond.min.js** files are included in the Bootstrapped template so that IE 8 can render the HTML5 elements.
- Bootstrap offers three form layouts namely, vertical (default), horizontal, and inline.

Check Your Progress

1. **Which of the following components is a jQuery plugin?**
   a. Page header
   b. Popover
   c. Badge
   d. None of these
2. **Which of the following classes is a base class for adding an icon in Bootstrap?**
   a. glyph
   b. glyphicon-<name>
   c. glyphicon
   d. glyph-name
3. **Which of the following classes adds spacing between two controls in a form?**
   a. .form-group
   b. .form-control
   c. .form-control-space
   d. .form-group-space
4. **Which of the following components aids in showing each title distinctly on a Webpage?**
   a. Pagination
   b. Badges
   c. Page header
   d. None of these
5. **Which of the following progress bars Bootstrap offers?**
   a. Animated
   b. Vibrant
   c. Horizontal
   d. Inline

Answers to Check Your Progress

1. Popover
2. `glyphicon`
3. `.form-group`
4. Page header
5. Animated

Answers to Quick Test

Quick Test 2.1

1. False
2. True

SESSION 3 - Basics of jQuery

Session Objectives:

This session introduces the jQuery library along with its basic concepts and implementation in Bootstrap. It also describes the different jQuery selectors and various ways of creating functions. The session also explains JavaScript Object Notation (JSON) along with its usage in jQuery. It further explains conversions to JSON and event delegation in jQuery.

At the end of this session, you will be able to:

- *Explain the basics of jQuery and its usage with Bootstrap*
- *Outline selectors and functions in jQuery*
- *Describe JSON and its usage in jQuery*
- *Explain conversions to JSON*
- *Describe the concept of event delegation and its usage in jQuery*

3.1 Introduction to jQuery and Integrating with Bootstrap

jQuery refers to a lightweight and concise JavaScript (JS) library that aims to make JS programming easier and swifter. It is a feature-rich and open-source library that simplifies several complicated tasks such as manipulating and traversing an HTML/CSS page, handling events, animating objects, and making Asynchronous JavaScript and XML (AJAX) calls.

The library offers an easy-to-use Application Programming Interface (API) that is compatible across several browsers. It wraps several common tasks into built-in methods. A Web developer can invoke the desired method by writing just a single line of JS code instead of writing several lines of it. This is how jQuery accomplishes the purpose of 'get more by writing less'.

3.1.1 Benefits of Using jQuery over JS

There are many JS frameworks available but jQuery is perhaps the most extendable one. Big brands such as Google, IBM, and Microsoft are using it. Following are the benefits of using jQuery over JS or other JS libraries:

- **Unobtrusive JS**: jQuery helps in designing Webpages through unobtrusive JS, an approach that separates functionality between the presentation and structure of a Webpage. In other words, inserting an event code in the script tag keeps the behavior separate from the Webpage's structure or content.
- **Lightweight**: The reduced library size is in Kilobytes (KB), which is perhaps smaller than that of a digital photo file. The jQuery version 2.x is smaller than the version 1.x. The former version is more widely used, as it supports Internet Explorer (IE) 6/7/8. The browser needs to download it once and cache it for using it in all Webpages of a site. In case a Web developer relies upon a Content Delivery Network (CDN), the browser tends to cache the library across several Websites instead of downloading for each.
- **Ease of Use**: This is due to the simple yet robust syntax and significantly reduced lines of JS code. The syntax makes it simple to select the page elements that need a change. For this, the library uses CSS3 selector specification, which eliminates the need to learn the jQuery syntax.
- **Big and Focused Library**: Unlike other JS libraries, Web developers can perform a variety of functions. Further, to make JS tasks easier, jQuery offers several utility functions.

With these functions, Web developers can parse data, trim strings, and search for array elements based on a filter. However, the library is focused, which means it contains only essential features. A Webpage downloads only those features that it requires.

- **Extensibility**: jQuery comes with a plugin framework due to which it is simple to extend jQuery. The framework includes both third-party and official plugins. So, if a feature does not exist in the library, a Web developer may include it through a plugin.
- **Browser Compatibility**: Just as in few other JS libraries, jQuery flawlessly resolves cross-browser issues. JS implementations tend to vary amongst browsers, which can take much time of Web developers to get the code working across browsers. However, the makers of jQuery have already implemented the code to display Webpages seamlessly across all major browsers.
- **Strong Community**: Although a new library, jQuery has an exclusive community through which the dedicated developers tend to improve its functionality. Through this community, there are several plugins available to accelerate the Web development process. Most of the plugins are secured and efficient.
- **AJAX Support**: jQuery easily allows creating AJAX templates for gaining the benefits of smoother interfaces and efficient loading. A user can perform an action on a Webpage without reloading the whole page.
- **Comprehensive Documentation**: Official Website of jQuery has in-depth documentation and tutorials, which helps even a beginner in programming to start easily.

3.1.2 Using Bootstrap and jQuery Together

Web developers can also jQuery with Bootstrap through the jQuery UI Bootstrap theme, as shown in Figure 3.1.

Figure 3.1: jQuery UI Bootstrap Theme

***Image Courtesy**: https://webresourcesdepot.com*

The theme allows using Bootstrap-themed widgets without breaking the appearance of components. Originally, Bootstrap and jQuery cannot exist simultaneously due to which using them together triggers conflicts with JS and CSS styles and classes. However, the new theme comes with the relevant JS and CSS for designing a Web application using jQuery and Bootstrap. It also offers the Bootstrap theme for several third-party jQuery widgets for designing user interfaces. Following are the steps to use the theme:

1. Download the archive from the Internet.
2. Unzip it.
3. Look for **custom-theme** folder within the **css** folder of the unzipped package.

4. Move the **custom-theme** folder to the **css** folder of the Web application for which this theme is to be used.
5. Add the relevant .css files to the Website's layout.

3.1.3 Using jQuery in Web Applications

The jQuery library is a JS file that a Web developer uses for programming in jQuery for developing a Web application. There are two ways to use the file, which are as follows:

- Download the library file from jQuery.com and refer it in the HTML code just as a Web developer would do for any other JS file. Save the downloaded file in the directory of Webpages.
- Include the library into the code by referring to a CDN, such as Google and Microsoft. Code Snippet 1 shows how to refer to the Microsoft CDN.

  Code Snippet 1:

```
<head>
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-
3.2.0.min.js"></script>
</head>
```

3.1.4 Understanding the jQuery Syntax

With jQuery, a Web developer can quickly find and select an HTML element and trigger the desired action or actions on it. For this purpose, the makers of jQuery have offered a customized syntax, which is as follows:

```
$(selector).action()
```

where,

- *$*: Is the sign that indicates the use of jQuery and is the jQuery identifier.
- *(selector)*: Is used to find and select the HTML element or elements.
- *action()*: Is an action that jQuery should perform on the selected element or elements.

Following are a few samples that indicate how the syntax is used:

- *$(this).hide()* for hiding the element that is selected at present
- *$("div").fadeIn()* for fading in all * * elements
- `$("#info").hide()` for hiding an element whose `id` is set as `info`
- `$(".info").show()` for showing an element whose `class` attribute as the value, `info`

From these examples, it is clear that the selector can be an id, class, element, or `this`. Any jQuery code exists within the `document ready` event. This helps in executing the code only after the Document Object Model (DOM) loads completely or is ready. Following is the syntax of the event:

Syntax

```
$(document).ready(function()
{
   // Code jQuery functions here.
});
```

```
OR
$(function()//Alternate syntax
{
   // Code jQuery functions here.
});
```

Where,

- $: Is the jQuery identifier.
- document: Refers to the DOM of the HTML page.
- ready: Is an event that is raised when the fully loaded DOM is ready for manipulation through JavaScript. At this point, the script can safely find and access the HTML elements in the DOM. For example, it can find the <a> element for associating it with a click event handler.
- function: Refers to an anonymous function, as it has no name and it contains an action to be taken.

It is rational to wait for the DOM to load fully before the HTML page handles any requests and responses. This is because it is not sensible to hide an element or resize an image, which is not loaded completely. This enables a Web developer to include JavaScript code inside the <head> tag.

3.2 Functions in jQuery

In jQuery, a Web developer can define a function in four different ways. It is also possible to define a custom function.

3.2.1 Basic Declaration

This is the easiest way of declaring a function in jQuery. Code Snippet 2 shows the basic declaration to define a function in jQuery.

Code Snippet 2:

```
function Multiply(var1, var2, var3)
{
    return (var1 * var2 * var3);
}
document.write("Basic Way:"+Multiply(10, 10, 8));
```

In Code Snippet 2, the Multiply function is defined to return the product of three numbers. This way of declaring a function is useful for testing something quickly. It is not a good choice for supporting code reuse.

3.2.2 Declaration as a Variable Name

A Web developer can define a function through an expression, which is assigned to a variable. After storing the expression in a variable, this variable is then used as a function. Code Snippet 3 uses a variable as a function.

Code Snippet 3:

```
var Multiply = function (var1, var2, var3)
{
    return (var1 * var2 * var3);
}
```

```
document.write("Function as Variable Name "+Multiply(10, 10, 8));
```

In Code Snippet 3, the code declares the variable `Multiply` and assigns an anonymous function to it. This function returns the product of three numbers. In the `write()` method, the variable itself is used as a function to return the product.

3.2.3 Self-invoking Function Declaration

Apart from assigning an anonymous function to a variable, a Web developer can pass it directly to a method. This argument acts as a self-invoking function, which is typically an expression that the script automatically invokes without being called. It has no name and is impossible to call it explicitly.

To declare such a function expression, a Web developer should add parentheses around the function. Code Snippet 4 defines a self-invoking function.

Code Snippet 4:

```
(function () {document.write("Self Invoking Function, Hello! I
called myself");})();
```

In Code Snippet 4, the self-invoking function is within parentheses and calls itself on its own. The string is the output that is displayed in the browser.

3.2.4 User-defined Function Declaration

Web developers can create their own jQuery functions, which are known as user-defined functions. The `jQuery.fn` or `$.fn` object allows creating a user-defined function. It is equal to `jQuery.prototype`. Code Snippet 5 creates a user-defined function.

Code Snippet 5:

```
$(document).ready(function(){
    $.fn.myFunction = function(){
        alert('You have successfully defined the function!');
    }
    $(".call-btn").click(function(){
        $.fn.myFunction();
    });
});
```

In Code Snippet 5, `$.fn` object allows creating a user-defined function, `myFunction`, which shows an alert message. It makes the function available for all objects of jQuery. This is evident when the click property of the button whose `class` attribute is set to `call-btn` which is used to invoke the newly created function.

Figure 3.2 shows a code that defines four functions, each with a different declaration method.

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
<script>
//Basic Way
function Multiply(var1, var2, var3)
{
    return (var1 * var2 * var3);
}
document.write("Basic Way:"+Multiply(10, 10, 8));
document.write("<br>");
//Function as Variable
var Multiply = function (var1, var2, var3)
{
    return (var1 * var2 * var3);
}
document.write("Function as Variable Name "+Multiply(10, 10, 8));
document.write("<br>");
//Self Invoking Function
(function () {
    document.write("Self Invoking Function, Hello! I called myself");
})();
//Creating own functions
$(document).ready(function(){
    $.fn.myFunction = function(){
        alert("You have successfully defined the function!");
    }
    $(".call-btn").click(function(){
        $.fn.myFunction();
    });
});
</script>
</head>
<body>
<div id="div1"><h2>Funtions in jQuery</h2></div>
 <button type="button" class="call-btn">Click Me</button>
</body>
</html>
```

Figure 3.2: Code for Declaring Functions in Different Ways

In figure 3.2, the code specifies the Google CDN reference. In the `<script>` tag, the code is executed to invoke the `Multiply` function, which is written in the basic as well as in the variable forms. The jQuery library invokes and executes the self-invoking function without writing any code for doing the same. Finally, when a user clicks the `Click Me` button, its `click` property invokes the user-defined function, `myFunction` and displays an alert box.

Figure 3.3 shows the output.

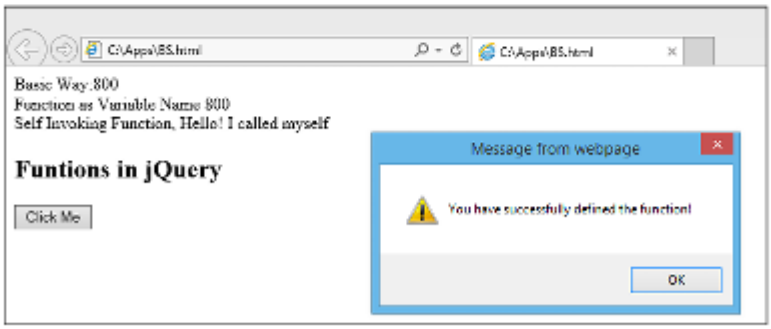


Figure 3.3: Output of Code for Declaring Functions in Different Ways

3.3 Selectors in jQuery

The jQuery library allows using CSS and its own custom selectors for swiftly accessing and manipulating the HTML elements in the DOM. A jQuery Selector refers to a function in the form of an expression for searching the matching element in the DOM. In other words, selectors allow selecting single or multiple HTML elements for triggering different actions on the selected ones. All selectors begin with the jQuery identifier ($), which is followed by parentheses. The library allows accessing the elements by:

- ID
- Class name
- Tag name
- Attribute
- Attribute values
- Other selectors

3.3.1 ID Selector

ID selector refers to the `id` attribute of a tag for searching the corresponding element. As the ID of each element is unique, the selector is useful only for finding a single tag at a time. It works by relying on the `document.getElementById()` function.

To find a tag or an element through its ID, a Web developer needs to specify ID selector, by beginning it with the hash (#) character and following it by the `id` value of the element.

Syntax

```
$('#<id>')
```

where,

`id`: Is the value of `id` attribute of the required HTML element.

Figure 3.4 shows how to use an ID selector in jQuery.

```
1  <!DOCTYPE html>
2  <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4      <meta charset="utf-8" />
5      <title></title>
6      <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.1.min.js"></script>
7      <script type="text/javascript">
8          $(document).ready(function () {
9              $('#button1').click(function () {
10                 alert('jQuery Selector');
11             });
12         });
13     </script>
14 </head>
15 <body>
16 <input id="button1" type="button" value="Click Me" />
17 </body>
18 </html>
```

Figure 3.4: Code for Using an ID Selector

In figure 3.4, the `click` event is associated with the `Click Me` button whose `id` is `button1`. When a user clicks this button, the event is raised to execute an anonymous function that displays the specified alert message.

Figure 3.5 shows the output.

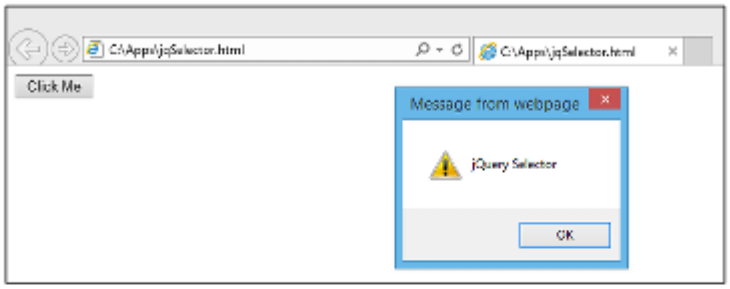


Figure 3.5: Output of Code Using an ID Selector

Following are some points to remember about ID selector:

- It is the most efficient of all selectors.
- It accesses only the first element, in case of two or more tags with the same ID.
- It is not the same as `document.getElementById()`, as the latter returns a DOM object and the former returns a jQuery object that includes the DOM object and provides different methods. Therefore, a Web developer can invoke jQuery methods such as `click()` on the returned object. To refer to the underlying DOM object using a jQuery object, a Web developer should use, `$('#btnidentifier')[0]`.
- It is slower in execution than `document.getElementById()`. Still, it is in use for harnessing the power of the extra functionality or methods of a jQuery object.
- It does not throw an error unlike `document.getElementById()`, which raises an error if the element is not found. The `length` property of ID selector allows checking whether an element was found and returned or not.

3.3.2 The Class Selector

The class selector looks for HTML elements that have the specified class. It works by using the `getElementsByClassName()` native function. However, the user's browser should support this function. To find a tag or an element through its class, a Web developer needs to specify the class selector by beginning it with the dot (.) character and following it by the `class` name of the element.

Syntax

```
$('.<class>')
```

where,

`class`: Is the value of `class` attribute of the required HTML element.

Code Snippet 6 shows some examples of class selectors.

Code Snippet 6:

```
$('.small') // Selects all elements with class small
$('.small,.big') // Selects all elements with class small or big
```

Figure 3.6 shows how to use a class selector in jQuery.

```
1  <!DOCTYPE html>
2  <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4      <meta charset="utf-8" />
5      <title></title>
6      <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.3.min.js"></script>
7      <script type="text/javascript">
8          $(document).ready(function () {
9              $('.small').css('border', '5px solid green');
10         });
11     </script>
12 </head>
13 <body>
14 <span class="small">
15         Span 1
16     </span>
17     <br /><br />
18     <div class="small">
19         Div 1
20     </div>
21     <br />
22     <span class="big">
23         Span 2
24     </span>
25     <div class="smaller">
26         Div 2
27     </div>
28 </html>
29
```

Figure 3.6: Code Using a Class Selector in jQuery

In figure 3.6, jQuery fetches all the elements whose class name is `small`. Then, it changes their style to apply a green border, which is 5-pixel thick by invoking the jQuery object's `css()` method.

Figure 3.7 shows the output.

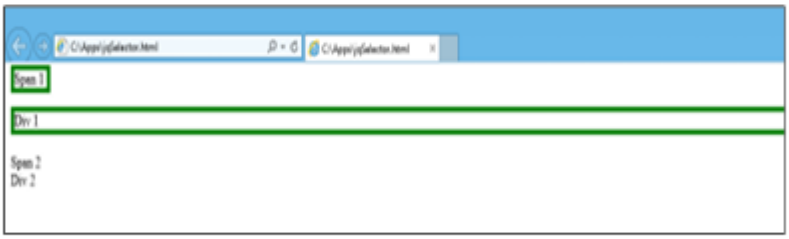


Figure 3.7: Output of Code Using a Class Selector

3.3.3 The Element Selector

The element selector access elements as per their name. To find an element by its name, a Web developer needs to specify the element selector within the parentheses following the $ character.

Syntax

```
$(<element>)
```

where,

`element`: Is the name of the element.

Code Snippet 7 shows some examples of element selectors.

Code Snippet 7:

```
$('td') // Selects all <td> elements
$('div a') // Selects all anchor elements that are inside the div element
```

Figure 3.8 shows the code for using an element selector.

```
1  <!DOCTYPE html>
2  <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4      <meta charset="utf-8" />
5      <title></title>
6      <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.3.min.js"></script>
7      <script type="text/javascript">
8          $(document).ready(function () {
9              alert($('td').length);
10         });
11     </script>
12 </head>
13 <body>
14 <table style="border:1px solid black; border-collapse:collapse">
15         <tr>
16             <td>India</td>
17             <td>Germany</td>
18             <td>France</td>
19         </tr>
20         <tr>
21             <td>Europe</td>
22             <td>Asia</td>
23             <td>Australia</td>
24         </tr>
25     </table>
26     </body>
27 </html>
```

Figure 3.8: Code for Using an Element Selector

In figure 3.8, the jQuery code displays the total number of `<td>` elements by triggering an alert box. To find the total number of `<td>` tags, it uses the `length` property of the element selector, `$('td')`.

Figure 3.9 shows the output, which returns the count as 6.

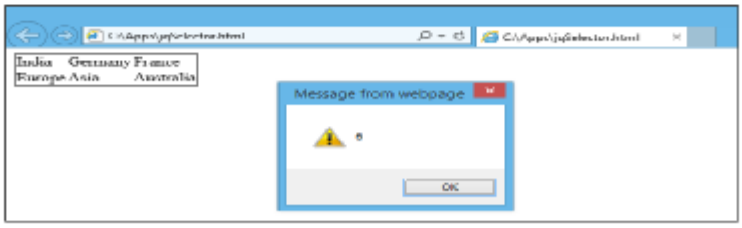


Figure 3.9: Output of Code for Using an Element Selector

3.3.4 The Checked Selector

The checked selector accesses and selects all selected radio buttons and check boxes. To access these input controls, a Web developer needs to include: `checked` inside double quotes, which should be within the parentheses following the `$` character.

Syntax

```
$(":checked")
```

Code Snippet 8 shows how to access all radio buttons at once that a user has selected.

Code Snippet 8:

```
$('input[type="radio"]:checked')
```

Figure 3.10 shows how to use the checked selector.

```
1   <!DOCTYPE html>
2   <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
3   <head>
4       <meta charset="utf-8" />
5       <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.1.min.js"></script>
6       <script type="text/javascript">
7           $(document).ready(function () {
8               $('#btnSubmit').click(function () {
9                   var result = $('input[type="checkbox"]:checked');
10                  if (result.length > 0) {
11                      var resultString = result.length + " checkbox(s) checked<br/>";
12                      result.each(function () {
13                          resultString += $(this).val() + "<br/>";
14                      });
15                      $('#divResult').html(resultString);
16                  }
17                  else {
18                      $('#divResult').html("No checkbox checked");
19                  }
20              });
21          });
22      </script>
23  </head>
24  <body>
25  Countries :
26      <input type="checkbox" name="skills" value="India" />India
27      <input type="checkbox" name="skills" value="Germany" />Germany
28      <input type="checkbox" name="skills" value="Croatia"/>Croatia
29      <input type="checkbox" name="skills" value="Russia" />Russia
30      <br /><br />
31      <input id="btnSubmit" type="submit" value="submit" />
32      <br /><br />
33      <div id="divResult">
34      </div>
35      </body>
36  </html>
```

Figure 3.10: Code for Using the Checked Selector

In figure 3.10, the jQuery code uses the checked selector to find out the number of selected check-boxes and stores it in the `result` variable. This variable then is recognized as the returned jQuery object. The object's `length` property helps in checking whether the user has selected any check boxes or not. If yes, the code under the `if` statement is executed, which stores the total count of selected check boxes in the `resultString` variable.

Next, the `each()` method of `result` object is used to refer to each selected check box for fetching the value of the corresponding `value` attribute.

After identifying the total count along with the check box names, ID selector is used to display the result by invoking its `html()` method. ID selector is referring to the `<div>` tag with the id, `divResult`. This whole process of finding the total count and check box names is triggered as soon as a user clicks the `submit` button.

Figure 3.11 shows the output.

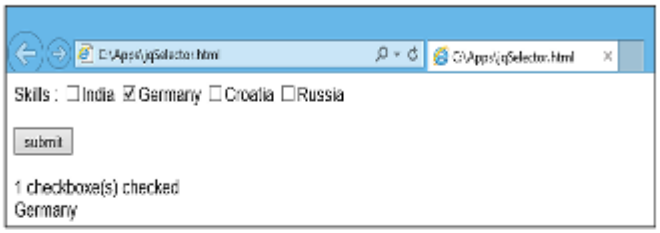


Figure 3.11: Output of Code Using the Checked Selector

3.4 JSON and Conversions to JSON

JSON refers to a lightweight format for exchanging and storing data. It is an easier alternative to eExtensible Markup Language (XML) in terms of usage. This is because it is a self-describing format due to which it is simple to comprehend the data exchanged between a server and a client (browser).

Exchanging data between these two nodes happens only in text format. JSON is a simple text format to use in this scenario, although it uses JS syntax. It also works independent of any programming language.

A Web developer can convert an object into JSON and then send it to the server. It is also possible to convert data obtained from the server into JSON objects and arrays. Both the processes are free from complex parsing.

3.4.1 Creating a JSON Object

The format of JSON is almost similar to JS objects. So, it is written in the form of key/value or name/value pairs. However, unlike in JS, the key or name in JSON is always a string due to which it must be within double quotes. Further, in JS, strings can appear in single or double quotes.

In JSON, a name in double quotes is separated by the value with a colon.

Syntax

```
{"name":value}
```

In a name/value pair, the value needs to a valid data type, which can be number, string, boolean, array, null, or object. JSON do not support date data type. So, the only way to represent a date in JSON is to make it a string. Code Snippet 9 shows some examples of name/value pairs in JSON.

Code Snippet 9:

```
{"city":"Moscow"}
{"age":40}
{"marriage":"1980-12-23"}
```

In Code Snippet 9, `city` is the key or name in double quotes and `Moscow` is its string value. Similarly, `age` is the key, while `40` is its numeric value. For the `marriage` key, the value is a string showing the wedding date. In jQuery, Web developers usually work with JSON objects. Code Snippet 10 shows how to store employee data in a JSON object, `employeeJSON`.

Code Snippet 10:

```
var employeeJSON =
{
    "firstName": "Philips",
    "lastName": "LG",
    "gender": "Male",
    "salary": 5000,
    "bike": null
};
```

In Code Snippet 10, the employee data is included inside the curly braces. Further, comma separates each name/value pair. There is no comma at the end of the last name/value pair.

Each name/value pair is referred to as a property. A Web developer can specify any number of properties in a JSON object. Code Snippet 11 represents the same data using XML.

Code Snippet 11:

```
<Employee>
  <firstName>Philips</firstName>
  <lastName>LG</lastName>
  <gender>Male</gender>
  <salary>5000</salary>
  <bike>null</bike>
</Employee>
```

From Code Snippet 11, it is clear that the JSON format is simpler than XML.

3.4.2 Accessing Data from a JSON Object

The dot (.) notation and the corresponding property name enable accessing data from a JSON object.

Syntax

```
<Object>.<propertyName>;
```

Code Snippet 12 shows how to fetch the first name of the employee from the `employeeJSON` object created in Code Snippet 10.

Code Snippet 12:

```
var firstName = employeeJSON.firstName;
```

Figure 3.12 shows the code for accessing all data from the `employeeJSON` object using jQuery.

```
1  <!DOCTYPE html>
2  <html lang="en" xmlns="http://www.w3.org/2000/xhtml">
3  <head>
4      <meta charset="utf-8" />
5      <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.1.min.js"></script>
6      <script type="text/javascript">
7          $(document).ready(function () {
8
9              // Creating a JSON object
10             var employeeJSON = {
11                 "firstName": "Philips",
12                 "lastName": "LG",
13                 "gender": "Male",
14                 "salary": 5000
15             };
16
17             // Accessing data from a JSON object
18             var result = '';
19             result += 'First Name = ' + employeeJSON.firstName + '<br/>';
20             result += 'Last Name = ' + employeeJSON.lastName + '<br/>';
21             result += 'Gender = ' + employeeJSON.gender + '<br/>';
22             result += 'Salary = ' + employeeJSON.salary + '<br/>';
23
24             $('#resultDiv').html(result);
25         });
26     </script>
27 </head>
28 <body style="font-family:Arial">
29     <div id="resultDiv"></div>
30 </body>
31 </html>
```

Figure 3.12: Code for Fetching Data from a JSON Object

In figure 3.12, the code inside the anonymous function creates the required JSON object as soon as the document is loaded. It then declares a variable named `result` for accessing the data from the object. The variable is set to a concatenated string, which uses the individual property name of the JSON object for fetching all its data. The result is a JSON string that is displayed through the `<div>` tag.

Figure 3.13 shows the output.

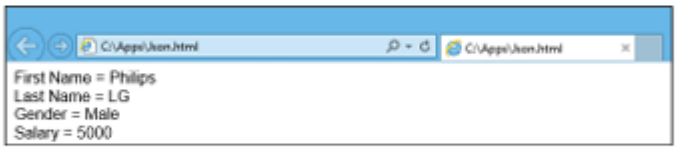


Figure 3.13: Output of Code for Fetching Data from a JSON Object

3.4.3 Using JSON Arrays

Let's imagine a scenario wherein a Web developer needs to store the data of more than one employee in a JSON object. This is where a JSON array is useful. A JSON array can hold several objects. To create a JSON array, include the objects in square brackets and separate each by a comma. Code Snippet 13 shows how to create a JSON array indicating the colors in a rainbow.

Code Snippet 13:

```
[ "Red", "Blue", "Green", "Yellow", "Violet", "Indigo", "Orange" ]
```

A JSON array is almost similar to a JS array. However, its values or elements can be number, string, boolean, array, null, or object. It cannot be a valid expression, unlike in JS. Code Snippet 14 shows how to create a JSON object holding the data of two employees.

Code Snippet 14:

```
var employeesJSON = [
{
    "firstName": "Philips",
    "lastName": "LG",
    "gender": "Male",
    "salary": 5000
},
{
    "firstName": "Samsung",
    "lastName": "Sam",
    "gender": "Female",
    "salary": 4000
}];
```

In Code Snippet 14, both elements in the array are employee objects. To access these from the JSON array, `employeeJSON`; a Web developer should use the object's index position. Code Snippet 15 shows how to retrieve the last name from both the objects of `employeesJSON` array.

Code Snippet 15:

```
var result = employeesJSON[0].lastName;
var result = employeesJSON[1].lastName;
```

3.4.4 Converting an Object or Array into a JSON String

When sending data to a server, it should be in the string format. So, for converting a JS object into a string, a Web developer should use `JSON.stringify()`. This method is used on both arrays and objects.

Figure 3.14 shows how to convert the `employeesJSON` array into a JSON string using `JSON.stringify()`.

```
1   <!DOCTYPE html>
2   <html lang "en" xmlns "http://www.w3.org/1999/xhtml">
3   <head>
4       <meta charset="utf 8" />
5       <script src="http://ajax.googleapis.com/ajax/libs/jquery/jquery-2.1.3.min.js"></script>
6       <script type "text/javascript">
7           $(document).ready(function () {
8               var employeesJSON = [
9                   {
10                      "firstName": "Philips",
11                      "lastName": "LG",
12                      "gender": "Male",
13                      "salary": 5000
14                  },
15                  {
16                      "firstName": "Samsung",
17                      "lastNum": "Sam",
18                      "gender": "Female",
19                      "salary": 4000
20                  }
21              ];
22
23              var JSONString = JSON.stringify(employeesJSON);
24              $('#resultDiv').html(JSONString);
25          });
26      </script>
27  </head>
28  <body style="font-family:Arial">
29      <div id="resultDiv"></div>
30  </body>
31  </html>
```

Figure 3.14: Code for Converting an Array into a JSON String

In figure 3.14, `JSON.stringify()` takes the array as an input and converts it into a string in JSON notation. This string is stored in `JSONString` variable and is displayed on the Webpage through the `<div>` tag. This string is now ready to be sent to the server.

Figure 3.15 shows the output.

Figure 3.15: Output of Code Converting an Array into a JSON String

3.4.5 Converting a String into a JSON Object or Array

A Web developer can parse a string using `JSON.parse()` to convert it into a JSON object or an array. To avoid a syntax error, the string should be in JSON format.

Figure 3.16 shows how to convert a string into a JSON array using `JSON.parse()`.

```
1   <!DOCTYPE html>
2   <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
3   <head>
4       <meta charset="utf-8" />
5       <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.3.min.js"></script>
6       <script type="text/javascript">
7           $(document).ready(function () {
8               var JSONString = '[{ "firstName": "Philips", "lastName": "LG", "gender": "Male", "salary": 5000 },
9               { "firstName": "Samsung", "lastName": "Sam", "gender": "Female", "salary": 4000 }]';
10
11              var employeesJSON = JSON.parse(JSONString);
12
13              var result = '';
14
15              $.each(employeesJSON, function (i, item) {
16                  result += 'First Name = ' + item.firstName + '<br/>';
17                  result += 'Last Name = ' + item.lastName + '<br/>';
18                  result += 'Gender = ' + item.gender + '<br/>';
19                  result += 'Salary = ' + item.salary + '<br/><br/>';
20              });
21
22              $('#resultDiv').html(result);
23          });
24      </script>
25  </head>
26  <body style="font-family:Arial">
27      <div id="resultDiv"></div>
28  </body>
29  </html>
```

Figure 3.16: Code for Converting a String into a JSON Array

In figure 3.16, a string variable, `JSONString` is set to a string in JSON syntax. The `JSON.parse()` function takes this string as a parameter and converts it into a JSON array, `employeesJSON`. Then, the `each()` method of jQuery allows looping through each employee element or object in the array to retrieve its corresponding property values.

Figure 3.17 shows the output.

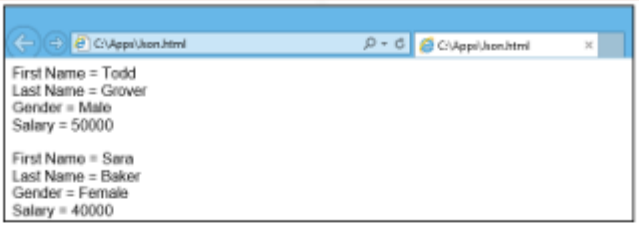


Figure 3.17: Output of Code Converting a String into a JSON Array

3.5 Event Delegation in jQuery

jQuery is customized for reacting to events occuring on a Webpage. An event denotes the exact moment when an action from a user occurs. A few examples of events include loading a Webpage, clicking an element, selecting a check box, and hovering the mouse over an element.

With events, Web developers can design dynamic Webpages. When an event is raised, an event listener, which is associated with source of event, listens to it and notifies the handler. An event handler, which is associated with the event, is usually a custom function that conveys what to do for handling the event.

In jQuery, event delegation enables a Web developer to assign a single listener to a parent element, which will listen for all descendants corresponding to a selector.

This is irrespective of whether these descendants are present now or are included later. The descendants of an element encompasses all its children elements including the great grand-child ones.

The `on()` and `delegate()` methods facilitate delegating an event in jQuery. Delegating an event eliminates the need to add event listeners to particular tags. This is because a Web developer can now add a listener to a parent element, which can analyze the bubbled events to look for a match on child elements.

For example, let's assume that there is a parent `<ul>` element with 50 child elements. Something should happen when a user clicks any child element. For this, a Web developer may add a distinct event listener to each `<li>` element. However, a problem arises when the developer frequently tends to add and remove these child elements. In this case, adding and removing the listeners would be a tedious job and may lead to performance issues. So, it would be much better to add a listener to the parent element.

However, an important question here is that how would the developer find out which child element was clicked? The solution here is to use the `target` property of the event object to obtain a reference to the clicked node while the event bubbles up to the parent element.

When the click event triggers on any child element of the `<UL>`, it also triggers on all ancestors of that element. This is called event bubbling in which an event bubbles upwards from the source element in the DOM tree. As a result, it activates each of its parent click handlers, which includes `<ul>`, `<div>`, `<p>`, `<body>`, `<html>`, and root. In other words, clicking an `<li>` tag indicates clicking the whole body of the document.

Event bubbling and `target` property are the main features of event delegation in jQuery. They boost performance when there are hundreds of child elements on which a specific event should be listened and handled.

The `target` property of any event refers to the element that originated the event. Through event delegation, a Web developer can add a handler to an element, wait for event bubbling, and quickly identify the event's source, which is the element where the event occurred. Following are the benefits of event delegation:

- Less handlers to take up memory
- Better performance
- Reduced risk of crashing
- No re-assigning the handlers if the child elements keep changing

Figure 3.18 shows how to delegate and undelegate a click event for an unordered list using the `on()` method of jQuery.

```
1   <!DOCTYPE html>
2   <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
3   <head>
4       <meta charset="utf-8" />
5       <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.3.min.js"></script>
6       <script type="text/javascript">
7           $(document).ready(function () {
8               $('ul').on('click', 'li', function () {
9                   $(this).fadeOut(500);
10              });
11
12              $('#btnAdd').on('click', function () {
13                  $('ul').append('<li>New List Item</li>');
14              });
15
16              $('#btnUndelegate').on('click', function () {
17                  $('ul').off('click', 'li');
18              });
19          });
20      </script>
21  </head>
22  <body style="font-family:Arial">
23      <input id="btnAdd" type="button" value="Add a New List Item" />
24      <input id="btnUndelegate" type="button" value="Undelegate" />
25      <ul>
26          <li>List Item1</li>
27          <li>List Item2</li>
28      </ul>
29  </body>
30  </html>
```

Figure 3.18: Code for Delegating and Undelegating an Event

In figure 3.18:

- Clicking any `<li>` tag triggers event bubbling in which the click event bubbles up to its parent element, `<ul>`. This is because there is no dedicated event handler attached to the clicked list item. The parent element handles the bubbled event, as it has the click event handler. The `on()` method of the jQuery object directly binds the `click` event to the `<ul>` tag and the anonymous function acts as an event handler. It takes three parameters namely, the event, selector parameter, and the anonymous function.

- The selector parameter informs the handler to listen to the click event. It also tells the handler to check whether the element triggering the click event matches with the selector parameter at the time of hearing. In this case, the source of the event is `<li>`, which matches with the second parameter. Thus, the anonymous function executes to fade out the clicked `<li>` element in `3000` milliseconds.
- When a user clicks the button whose `id` is `btnAdd` for dynamically adding a new list item, the item is added to `<ul>` using the `append()` method. There is no need to specify the button's click event handler explicitly in the `<input>` tag. As the new item is added to the same parent element `<ul>`, the click event of the list item also gets bubbled to the same parent.
- Upon clicking the `Undelegate` button, the `undelegate()` method of the `<ul>` element selector is invoked to stop event delegation.

Figure 3.19 shows the output.

Figure 3.19: Output of Code Delegating and Undelegating an Event

Quick Test 3.1

1. The checked selector allows finding which radio buttons are selected.
   a. True
   b. False

2. It is impossible to call a self-invoking function explicitly.
   a. True
   b. False

Summary

- jQuery is a lightweight, open-source JS library that makes JS programming easier and more efficient.
- jQuery uses unobtrusive JS, offers a variety of reusable functions, is extensible, and is compatible with modern browsers.
- Web developers can either download the jQuery file or include it by referring to a CDN for using jQuery in their Web applications.
- The customized syntax of jQuery is `$(selector).action()`.
- There are four ways of declaring a function in jQuery, which are basic, as a variable, self-invoking, and user-defined.
- A jQuery Selector refers to a function in the form of an expression for searching the matching element in the DOM.
- The jQuery library allows selecting the HTML elements by ID, class name, tag name, attribute, and attributed values.
- JSON is a lightweight format for exchanging and storing data as well as an easier alternative to XML.
- Unlike in JS, the key or name in JSON is always a string.
- Each name/value pair in JSON format is referred to as a property.
- A JSON object is a collection of properties, each separated by a colon.
- The `JSON.stringify()` method converts a JSON object into a JSON string.
- The `JSON.parse()` method converts a JSON string into a JSON object or array.
- Event delegation allows assining a single listener to a parent element, which will listen for all descendants corresponding to a selector.
- The `on()` and `delegate()` methods facilitate delegating an event in jQuery.
- Event bubbling in which an event bubbles upwards from the source element in the DOM tree.
- Event bubbling and `target` property are the main features of event delegation.

Check Your Progress

1. **Which of the following statements about jQuery is true?**
   a. It has a small but vibrant community
   b. It uses obtrusive JS
   c. It supports AJAX
   d. None of these
2. **_____ is a valid jQuery selector.**
   a. `$(this)`
   b. `$(p)`
   c. `$(*)`
   d. `$(#btn)`
3. **The _____ object allows declaring a user-defined function.**
   a. `.fn`
   b. `$.fn`
   c. `Jquery.prototype`
   d. `$.fn.prototype`
4. **Which of the following methods converts an object into a JSON string?**
   a. `JSON.string()`
   b. `JSON.convertostring()`
   c. `JSON.stringify()`
   d. `JSON.tostring()`
5. **Which of the following statements about event delegation is true?**
   a. It features event bubbling
   b. It assigns an event handler to a parent element
   c. It boosts performance
   d. All of these

Answers to Check Your Progress

1. It supports AJAX
2. `$(this)`
3. `$.fn`
4. `JSON.stringify()`
5. All of these

Answers to Quick Test

Quick Test 3.1

1. True
2. True

SESSION 4 – Functions, Widgets, Plugins in jQuery

Session Objectives:

This session introduces the built-in methods in jQuery and elaborates briefly on the `live()` method for event delegation. It also describes the different Asynchronous JavaScript and XML (AJAX) methods that jQuery supports and the callback concept. The session also briefly introduces the jQuery UI JavaScript (JS) library, its different widgets including autocomplete, and the datatable dynamic control.

At the end of this session, you will be able to:

- *Describe various types of functions in jQuery*
- *Explain use of AJAX in jQuery*
- *Elucidate the callback concept in jQuery*
- *Describe implementation and use of various controls such as widgets and datatables in jQuery*

4.1 Built-in Methods in jQuery

The jQuery framework features different capabilities of JavaScript (JS). As JS has a handy set of built-in functions, all of them are also available through this framework. This is one of the examples proving that functions are reusable blocks. The JS built-in functions are methods with which Web developers can quickly manipulate data values such as dates, numbers, and strings.

Table 4.1 shows some of the built-in jQuery methods.

Method	Description
`charAt()`	Fetches the character at the mentioned index.
`concat()`	Returns a new string by combining two strings.
`indexOf()`	Fetches the index inside the requesting string object of the initial occurrence of the mentioned value. If nothing is found, it returns `-1`.
`length()`	Returns the number of characters in a string.
`forEach()`	Invokes a function for each array element.
`pop()`	Removes and returns the last array element.
`push()`	Appends single or multiple elements at the end of an array and fetches its modified length.
`substr()`	Extracts the string characters starting at the mentioned index location until the specified index length.
`reverse()`	Changes the order of array elements by making the first one last and the last one first.
`sort()`	Arranges the array elements in an ascending or descending order.
`toLowerCase()`	Converts a string value into lower case.
`toUpperCase()`	Converts a string value into upper case.
`tostring()`	Converts a given value into a string.

Table 4.1: Built-in Methods in jQuery

Figure 4.1 shows the code using some of the built-in methods.

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
<script>
function myFunction() {
    var str = "HELLO WORLD";
    var res = str.charAt(0)
    document.write("charAT Function:" +" " + res);
    document.write("<br>");
    var str1 = " FOLKS"
    document.write("concat Function:" + " " + str.concat(str1));
    document.write("<br>");
    document.write("Index Of Function:" +" "+ str.indexOf("WORLD"));
    document.write("<br>");
    document.write("Length of string:" +" "+ str.length);
    document.write("<br>");
    document.write("Converting to Lower case:" + " "+ str.toLowerCase());
}
</script>
</head>
<body>

<p>Click here to execute builtin functions in jQuery.</p>

<button onclick="myFunction()">Click me</button>

</body>
</html>
```

Figure 4.1: Code Using Built-in Methods

In figure 4.1, the code uses different built-in functions of jQuery for manipulating two strings. The `charAt()` built-in function returns the character at 0th position in the string, which is precisely the first character. The `concat()` function adds two strings namely, `str` and `str1`. The `indexOf()` function retrieves the index position from where the string `WORLD` starts in the string, `str`. The `length()` function retrieves the total number of letters in the `str` string. Finally, the `toLowerCase()` function converts the upper case letters in the string to lower case.

Figure 4.2 shows the output.

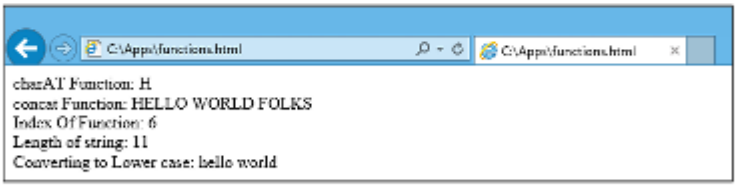


Figure 4.2: Output of Code Using Built-in Methods

4.1.1 Live Function in JQuery

The built-in `live()` method delegates single or multiple event handlers for all elements that match with the selector and defines a function to run once the event occur. Event handlers attached in this manner work for both existing and upcoming elements that match the selector. The `live()` method is just another way to perform event delegation in jQuery.

The on() and delegate() methods facilitate bubbling up the event to the source's parent element. On the other hand, the live() method bubbles up to the document object. A Web developer can remove the associated event handler using the die() method.

Syntax

```
$(selector).live(event,data,function);
```

where,

- event: Refers to single or multiple events to delegate to the elements, with multiple event values parted by space.
- data: Is optional and represents extra data to be passed to the function.
- function: Indicates the function to execute once the event takes place.

In the latest version of jQuery, the on() method takes care of all event delegation tasks. It can handle all tasks of the live() method.

Note: jQuery recommends using on() and off() methods instead of live() and die() methods to the users of latest version of jQuery.

Figure 4.3 shows the code for using live(), on(), and die() methods.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7/jquery.min.js">
5  </script>
6  <script type="text/javascript">
7          $(document).ready(function () {
8              $('li').live('click', function () {
9                  alert("live event triggered")
10
11             });
12
13             $('#btnAdd').on('click', function () {
14                 $('ul').append('<li>New List Item</li>');
15             });
16
17             $('#btnUndelegate').on('click', function () {
18                 $('li').die('click');
19             });
20         });
21     </script>
22 </head>
23 <body style="font-family:Arial">
24     <input id="btnAdd" type="button" value="Add a New List Item" />
25     <input id="btnUndelegate" type="button" value="Undelegate" />
26     <ul>
27         <li>List Item</li>
28         <li>List Item</li>
29     </ul>
30 </body>
31
32 </html>
```

Figure 4.3: Code Using live(), on(), and die() Built-in Methods

In figure 4.3, the code refers to an older version of jQuery framework to demonstrate the use of live() method. The live() and on() methods attach the event handler to all <li> elements

and btnAdd button, respectively. Both of them take the event and an anonymous function as parameters. When a user clicks any list item, the live() method executes the function and displays the alert box. When a user clicks the button, the on() method runs the function to add a new list item to the existing unordered list. Clicking the btnUndelegate button invokes the on() method, which then calls the die() method to stop event delegation.

Figure 4.4 shows the output.

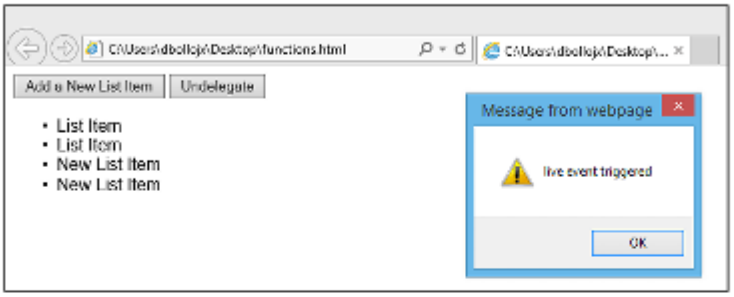


Figure 4.4: Output of Code Using live(), on(), and die() Built-in Methods

4.2 jQuery AJAX

AJAX allows updating different parts of a Webpage by exchanging data with the Web server. However, it does so without reloading or refreshing the whole Webpage. Some examples of AJAX Websites are Gmail and YouTube.

Coding regular AJAX without jQuery can be a bit complicated, as its syntax for implementation tends to vary from one browser to another. This means a Web developer need to write additional code for testing their Webpages on each of the major browsers.

However, with jQuery, this extra effort is no longer required, as the makers have already coded for it. Web developers can now code implement an AJAX functionality in just one or two lines. So, in jQuery, it is easier to write AJAX code.

jQuery provides many AJAX methods for developing Web applications. Through the jQuery AJAX methods, Web developers can request data from a distant Web server via GET and POST. They can request data in any of the four different formats namely, text, JavaScript Object Notation (JSON), eXtensible Markup Language (XML), and HTML. Further, they can easily load the remote data quickly into desired HTML elements.

4.2.1 AJAX jQuery Load() Method

The jQuery load() method is an easy and robust AJAX functionality. As the name suggests, it loads the requested data from a Web server and inserts it into the desired HTML element.

Syntax

```
$(selector).load(URL,data,callback);
```

where,

- URL: Indicates the Uniform Resource Locator (URL) of a resource on a server, a Web developer intends to load.

- `data`: Is optional and indicates a set of key/value pairs as a query string in the form of JSON object or string that goes along with the request. The default request method used here is GET.
- `callback`: Is optional and indicates the name of a callback function that will run after the `load()` method completes executing the request.

Figure 4.5 shows how to fetch data from an HTML file on a server and load it in the body of a `<div>` element.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
5  <script>
6  $(document).ready(function(){
7      $("button").click(function(){
8          $("#div1").load("test.txt");
9      });
10 });
11 </script>
12 </head>
13 <body>
14
15 <div id="div1"><h2>Ajax load demo</h2></div>
16
17 <button>Get test file Content</button>
18
19 </body>
20 </html>
```

Figure 4.5: Code Using the `load()` Built-in Method

In figure 4.5, once the document is ready for manipulation and when a user clicks the button, the click event is raised and the `load()` method triggers. The method triggers an AJAX request to the URL.

The `load()` method takes the URL as the parameter and fetches its contents. Here, the text file is in the same folder as this HTML file. The method finally loads the fetched content in the body of `<div>` by assigning the content to it through the id selector.

Figure 4.6 shows the output.

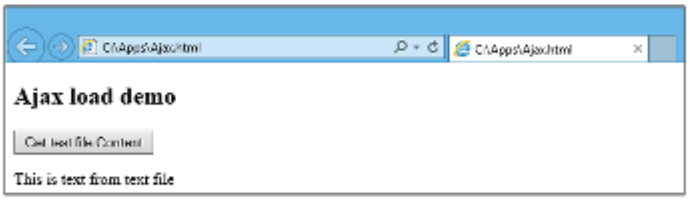


Figure 4.6: Output of Code Using the `load()` Built-in Method

4.3 Callbacks in jQuery

A callback function runs once the current jQuery effect such as hide, fade, or slide, is over. Usually, in JS, execution occurs line by line. Nevertheless, when there is an effect to display, the next code line can execute even if the former is not over. This can lead to errors. To keep this risk away, a Web developer defines a callback function.

Syntax for Hide Effect

```
$(selector).hide(speed,callback);
```

where,

- `speed`: Is optional and indicates the speed of hiding an element in milliseconds or as `fast` or `slow`.
- `callback`: Is optional and indicates the name of a callback function to run after `hide()` executes completely.

Figure 4.7 shows the code for hiding a paragraph without using a callback function.

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
5   <script>
6   $(document).ready(function(){
7       $("button").click(function(){
8           $("p").hide(1000);
9           alert("The sentence is now hidden");
10      });
11  });
12  </script>
13  </head>
14  <body>
15
16   <button>Hide</button>
17
18   <p>This is a sentence with little content.</p>
19
20  </body>
21  </html>
```

Figure 4.7: Code Hiding an Element without a Callback Function

In figure 4.7, the code invokes the anonymous function once a user clicks the button. The function executes to hide the sentence in the `<p>` element in `1000` milliseconds and show the alert dialog box.

Figure 4.8 shows the output.

Figure 4.8: Output of Code Hiding an Element without a Callback Function

In figure 4.8, it is clear that the alert box is displayed without waiting for the hide effect to get over. Now let's see what happens when a callback fucntion is present.

Figure 4.9 shows the code for hiding a paragraph using a callback function.

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
5   <script>
6   $(document).ready(function(){
7       $("button").click(function(){
8           $("p").hide("slow", function(){
9               alert("The sentence is now hidden");
10          });
11      });
12  });
13  </script>
14  </head>
15  <body>
16
17  <button>Hide</button>
18
19  <p>This is a sentence with little content.</p>
20
21  </body>
22  </html>
```

Figure 4.9: Code Hiding an Element with a Callback Function

In figure 4.9, the code invokes the anonymous function once a user clicks the button. The function executes to hide the sentence in the `<p>` element slowly and then, invoke a callback function to display the alert dialog box.

Figure 4.10 shows the output.

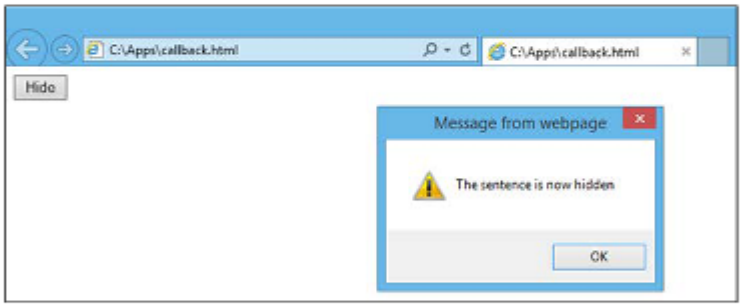


Figure 4.10: Output of Code Hiding an Element with a Callback Function

In figure 4.10, it is clear that the alert box is displayed only after the sentence in the paragraph hides. completely. This means that the callback function is called only after the effect is completed.

In the `load()` method, the optional callback parameter indicates a callback function. This executes only when the desired content loads into intended section of a Webpage. This callback function can take three parameters.

Syntax

```
$(selector).load(URL,data,function(response,status,XHR));
```

where,

- `response`: Holds the fetched content in case loading is successful.

- status: Has the status of the loading action.
- XHR: Possesses the XMLHttpRequest object with response text.

Note: XHR has all contents of the browser's native XMLHttpRequest object.

Code Snippet 1 shows how to pass a callback function as a parameter to the load() method.

Code Snippet 1:

```
$("btnLoad").click(function(){
    $("#divLoad").load("test.txt", function(response, status, xhr){
        if(status == "success")
            alert("Content loaded!");
        if(status == "error")
            alert("Error in loading: " + xhr.status);
    });
});
```

In code snippet 1, the code displays the alert box once the load() method's execution is over. If the method succeeds, it displays the corresponding alert message. Otherwise, it shows an error message using the status property of XMLHttpRequest object.

4.3.1 jQuery $.get() Method

The $.get() method asks data of a resource from a Web server through a GET request.

Syntax

```
$.get(URL,callback);
```

where,

- URL: Indicates the URL a Web developer intends to request.
- callback: Is optional and is a function to be executed when the request is successful.

Figure 4.11 shows how to use $.get() method with a callback function to display the contents of an ASP file into an HTML page.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
5  <script>
6  $(document).ready(function(){
7      $("button").click(function(){
8          $.get("demo_test.asp", function(data, status){
9              alert("Data: " + data + "\nStatus: " + status);
10         });
11     });
12 });
13 </script>
14 </head>
15 <body>
16
17 <button>Get response from concerning asp page</button>
18 </body>
19 </html>
```

Figure 4.11: Code for Using $.get() with a Callback Function

In figure 4.11, the occurrence of click event invokes the `$.get()` method, which fetches the data from the ASP file at the specified URL. An ASP file is a server side script. The method takes two parameters, of which the second one is the callback function. This function contains the requested data and status of the request. Following is the content existing in the ASP file:

```
<%
response.write("Response from asp file")
%>
```

`Response.write()` works in the same way as `document.write()` in HTML. It shows the message on the HTML page.

Figure 4.12 shows the output.

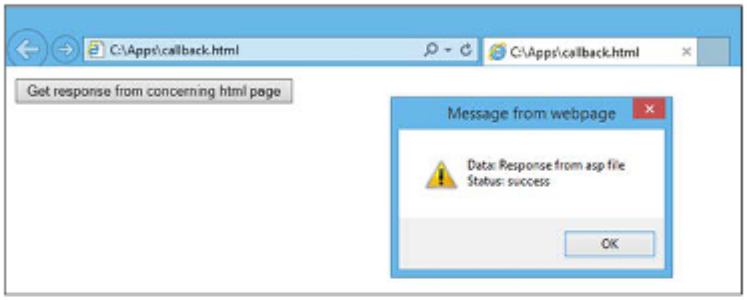


Figure 4.12: Code for Using `$.get()` with a Callback Function

Web developers might think that the `$.get()` and `load()` methods are same. However, there is a difference between these two methods of jQuery. Table 4.2 distinguishes them.

Point of Distinction	Load()	$.get()
Functionality	Loads only HTML data after retrieving it from the server.	Loads any type of data, such as JSON and script.
Type of HTTP Request	Sends a GET or POST request, but it depends on whether the specified data parameter. If data parameter is specified, POST is used. Otherwise, GET is used.	Sends a GET request always.
Selector Usage	Follows a selector for inserting the retrieved content into the element that matches with the selector. For example: `$('selector').load ('url', 'data');`	Does not follow a selector while coding. Instead, it invokes a callback function to insert the retrieved data from the server into an HTML element. For example: `$.get('url', 'data', function (response) { $('selector').html(response); });`

Table 4.2: Load() versus Get()

4.3.2 The jQuery $.post() Method

The `$.post()` method asks data of a resource from a Web server through a POST request.

Syntax

```
$.post(URL,data,callback);
```

where,

- URL: Indicates the URL a Web developer intends to request.
- data: Is optional and represents extra data to be passed to the function.
- callback: Is optional and is a function to be executed when the request is successful.

Figure 4.13 shows how to use $.post() method with a callback function to send data and display the contents of an ASP file into an HTML page.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
5  <script>
6  $(document).ready(function(){
7      $("button").click(function(){
8          $.post("demo_test_post.asp",
9          {
10           name: "Bruce Wayne",
11           city: "Gotham"
12         },
13         function(data,status){
14             alert("Data: " + data + "\nStatus: " + status);
15         });
16     });
17 });
18 </script>
19 </head>
20 <body>
21
22 <button>Send an HTTP POST request and get the result back</button>
23
24 </body>
25 </html>
```

Figure 4.13: Code for Using $.post() with a Callback Function

In figure 4.13, the jQuery $.post() method takes three parameters: First being the ASP file URL, second being the data of name and city, and third being the callback function. The code passes this data along with the request. In the callback function, the first parameter possesses the requested page's content, while the second one indicates the request's status.

By sending data as its second parameter, we are writing it to the demo_test_post.asp file and fetching the same for displaying on the HTML page. The ASP file has the following script:

```
<%
dim fname,city
fname=Request.Form("name")
city=Request.Form("city")
Response.Write("Hello " & fname & ", where are you from" & city &
". ")
%>
```

The ASP script here reads the parameters, assigns them to variables, and returns a result. The script declares two variables namely, fname and city. Then, it assigns the values fetched from the HTML page to them using request.form().

Figure 4.14 shows the output.

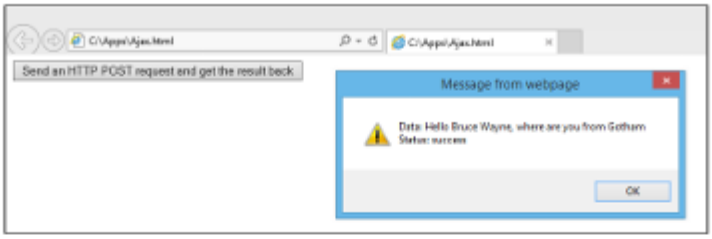


Figure 4.14: Output of Code Using $.post() with a Callback Function

Web developers may consider that the $.get() and $.post() methods are same. However, there is a difference between the two. Table 4.3 distinguishes them.

Point of Distinction	$.get()	$.post()
Functionality	Is mainly for obtaining data from a Web server although it can send data.	Is mainly for changing data on a Web server although it can send data.
Caching	Is capable of caching the requested data and returning it.	Does not cache data.
Method of Sending Data	Appends data to the URL as a query string.	Includes the data in the message body.
Limitation on Data to Be Sent	Puts a limit on the amount of data to be sent because of the limit on the length of the data passed to an URL.	Has no such limitation.

Table 4.3: $.get() versus $.post()

4.4 Autocomplete Function in jQuery

While filling a form, it is common to see a drop-down menu giving a list of words that best matches with the typed characters. In short, the browser tries to predict the exact value a user wishes to enter in the text field. This is commonly evident when a user types just two or more characters of the word or phrase in the search box of Google after which a drop-down.

Auto completion is a comonly used technique in modern Websites to offer a list of suggestions for selecting the exact value instead of typing it. In jQuery, the autocomplete widget in JQuery UI provides this functionality. It is a control that functions similar to a drop-down list. However, it filters the options for showing the ones that match with what the user is entering into a control.

JQuery UI stands for jQuery User Interface and is a robust open-source JS library based on jQuery framework. It offers a collection of jQuery plugins for easily adding new and complex functionalities to the library of jQuery. It comprises the following:

- interactive plugins or interactions such as resize and drag
- jQuery plugins or widgets for inserting visual components such as menus, autocomplete, and progress bars
- Effects encompassing transitions and custom animations, including toggle, show, and hide

- Utilities as modular functions or tools such as position and toggle for internal use by jQuery UI

Just as jQuery, Web developers can either download the jQuery UI files from the its Official Website or use a CDN to refer to them. Code Snippet 2 shows the CDN links to include in the <head> tag.

Code Snippet 2:

```
<link href = https://code.jquery.com/ui/1.10.4/themes/ui-
lightness/jquery-ui.css rel = "stylesheet">
<script src = "https://code.jquery.com/jquery-1.10.2.js"></script>
<script src = "https://code.jquery.com/ui/1.10.4/jquery-
ui.js"></script>
```

In code snippet 2, the first link includes the theme of jQuery UI, ui-lightness, through CSS to give a stylish look to the user interface. The second link includes the jQuery library. It is essential to add this link, as the new library is built based on the jQuery library. Finally, the third link includes the jQuery UI library to enable it for the Webpage in which these three links exist.

To use the autocomplete plugin, the library offers the autocomplete() method. Using this method, a Web developer can specify a list of predicted values as a drop-down for any input field.

Syntax

```
$ (selector, context).autocomplete (opt);
```

This syntax states that an <input> element should be handled as an input field and displayed just atop the drop-down list. The opt parameter refers to an object that conveys the behavior of the list. A Web developer can include more than one option, which they can separate using a comma.

Syntax

```
$(selector, context).autocomplete({opt1: val1, opt2: val2..... });
```

Table 4.4 lists the different options.

Option	Description
appendTo	Appends an element to the list. The value is null by default.
source	Indicates how the data will be obtained, which will then match with the input data. It is essential to specify a value for using the autocomplete widget. The default value is none.
delay	Refers to the waiting time in milliseconds before fetching the matching values, as per the value of the source option. The default value is 300. This option alleviates thrashing, especially while fetching non-local data by giving time for typing more characters prior to auto search.
miniLength	Refers to the number of characters to be typed before fetching the matching values. This option averts generating a

	too big list. The default value is `1`.
`Autofocus`	Focuses the first menu item when its value is `true`. The default value is `false`.
`Disabled`	Inactivates the widget if the value is `true`. The default value is `false`.
`Position`	Specifies the position of the list with regards to input element. The default value is `{ my: "left top", at: "left bottom", collision: "none" }`.

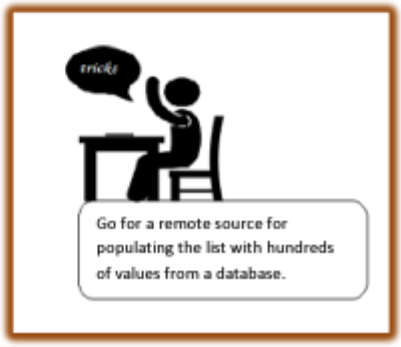
Table 4.4: Autocomplete Options

Figure 4.15 shows how to use the `autocomplete()` method in jQuery.

```
1  <!doctype html>
2  <html lang = "en">
3     <head>
4        <meta charset = "utf-8">
5        <title>jQuery UI Autocomplete function</title>
6        <link href = "https://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css"
7         rel = "stylesheet">
8        <script src = "https://code.jquery.com/jquery-1.10.2.js"></script>
9        <script src = "https://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
10       <script>
11        $(function() {
12           var countries = [
13              "Germany",
14              "Ghana",
15              "Croatia",
16              "Cambodia",
17           ];
18           $( "#autocomplete-1" ).autocomplete({
19              source: countries
20           })
21        });
22       </script>
23    </head>
24    <body>
25       <!-- HTML -->
26       <div class = "ui-widget">
27          <p>Type "g" or "c"</p>
28          <label for = "autocomplete-1">Tags: </label>
29          <input id = "autocomplete-1">
30       </div>
31    </body>
32 </html>
```

Figure 4.15: Code for Using the `autocomplete()` Method in jQuery

In figure 4.15, the CDN links for jQuery UI and jQuery exists under the `<head>` tag. Next, the code defines an array named `countries` holding the names of four countries. Then, the `autocomplete()` method is associated with the input field whose id is `automplete-1`. It takes the `source` option whose value is set to `countries`, which is an array. This populates the autocomplete drop-down list with the elements of the array. This list appears when the user enters '`g`' or '`c`' in the input field. The type of source mentioned here is local, as it is defined in the code. A local source is preferable for having a fewer values in the autocomplete drop-down list.



In the HTML body, the `div` tag contains the `.ui-widget` class. This class is applicable to the autocomplete widget's outer container for setting the font family and its size.

Figure 4.16 shows the output.

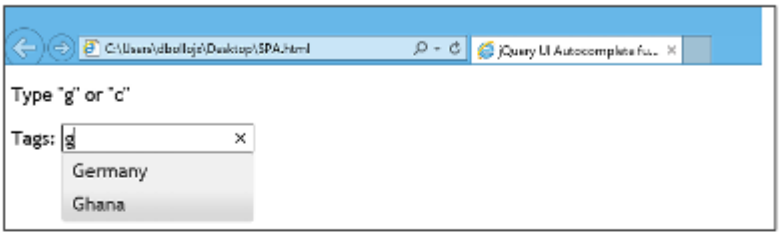


Figure 4.16: Output of Code Using the `autocomplete()` Method in jQuery

4.5 Widgets in jQuery

A few visual controls or elements on a Webpage are likely to accept the intended state, roles, and responsibilities naturally at the time of development. Adding more such elements to a Webpage usually increases the level of complexity. This triggers a need for a maintainable design solution. Such a solution features modularity, purposeful design, and unit tests that strengthen the major parts of the environment, coding language, and the platform.

In case of jQuery, the platform is the browser, JS is the language, and jQuery and jQuery UI libraries denote the environment's major parts. The libraries tend to make the code simple by:

- Reducing the number of lines.
- Resolving browser compatibility issues.
- Ensuring consistency in AJAX interactions.
- Implementing modularity.

At the core of jQuery UI's visual section is a widget. A widget refers to an object associated with an element on a Webpage for handling many aspects with other JS objects or widgets. These aspects are:

- Lifespan
- Interaction
- State
- Inheritance

Widgets in jQuery come from jQuery UI and are handy for designing Webpages that are highly interactive. The jQuery UI widgets are special plugins that simplify the task of applying a functionality to their associated elements. The standard jQuery plugins do not possess a few built-in capabilities, such as regulating the plugin's lifespan, revealing methods, and associating data with its elements. However, widgets come with these capabilities. Following are the jQuery UI widgets:

- Accordions
- Buttons
- Tabs
- Sliders
- Autocomplete

- Dialogs
- Datepickers
- Progress bars
- Menus
- Spinners
- Tooltips

4.5.1 Using the Tooltip Widget

Replacing the native tooltip, the Tooltip widget comes with fresh themes and facilitates customization. It allows display other content instead of just the `title` attribute, such as AJAX-retrieved data and footnotes. The widget also customizes the alignment of tooltips, such as aligning it in the middle. It also applies additional styling to personlize the appearance. This is useful for indicating a warning or error.

Web developers can apply a tooltip to any HTML element. In HTML without jQuery, a Web developer simply needs to specify the `title` attribute to the desired element for showing a tooltip. The browser displays the value of this attribute as a tooltip when a user hovers over the element. Code Snippet 3 illustrates the use of both the `title` attribute and Tooltip widget of jQuery UI.

Code Snippet 3:

```
HTML
<label id="lName" for="tName" title="Full Name">Name</label>
<input id="tName" type="text" title="Your full name as it appears in
memo"/>

jQuery
$('#tName').tooltip();
```

In code snippet 3, the `label` element shows the native tooltip, while the text field uses the widget's `tooltip()` method to display the tooltip. Code Snippet 4 shows how to apply the widget to all the elements in a document, which have the title attribute.

Code Snippet 4:

```
$(document).tooltip();
```

jQuery UI offers the `tooltip()` method for applying a tooltip to any HTML element. It applies a fading effect by default to hide or show the tooltip instead of simply toggling the tooltip.

Syntax

```
$(selector, context).tooltip(opt);
$(selector, context).tooltip({opt1: val1, opt2: val2..... });
```

Table 4.5 lists the different options that a Web developer can pass as parameters to the `tooltip()` method.

Option	Description
`content`	Indicates the tooltip content. The default value is a function that returns the value of the title attribute.
`items`	Denotes the items that can display the tooltip. The default value is `title`.

disabled	Inactivates the widget if the value is true. The default value is false.
hide	Denotes the effect of animation while hiding the widget. The default value is true.
show	Conveys the manner in which the display of the tooltip shall be animated. The default value is true.
track	Allows the tooltip to follow the mouse if the value is true. The default value is false.
position	Sets the tooltip's position with regards to its target element. The values can be at, within, of, using, collision, and my. The default value is a function returning the value of title attribute.
tooltipClass	Is useful for showing an error or warning via a tooltip. The default value is null.

Table 4.5: Tooltip Options

Figure 4.17 shows how to use the tooltip() method in jQuery.

```
1  <!doctype html>
2  <html lang = "en">
3     <head>
4        <meta charset = "utf-8">
5        <title>jQuery UI Tooltip</title>
6        <link href = "https://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css"
7         rel = "stylesheet">
8        <script src = "https://code.jquery.com/jquery-1.10.2.js"></script>
9        <script src = "https://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
10       <script>
11        $(function() {
12           $("myTooltip").tooltip();
13        });
14       </script>
15    </head>
16
17    <body>
18       <!-- HTML -->
19       <label for = "name">FirstName:</label>
20       <input id = "myTooltip" title = "Enter Your first name">
21    </body>
22 </html>
```

Figure 4.17: Code for Using the tooltip() Method in jQuery

In figure 4.17, the tooltip() method is invoked to display the value of title attribute near the text field. Figure 4.18 shows the output.

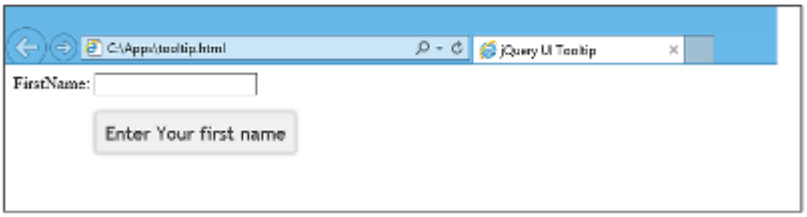


Figure 4.18: Output of Code Using the tooltip() Method in jQuery

4.5.2 Using the Dialog Widget

The Dialog widget provides a floating window that has a title and a body with content. The window is moveable via dragging and resizable through stretching. A user can even close the dialog by clicking the **x** icon present in its top-right corner. The window is dynamic enough to display a scrollbar if the content exceeds its maximum height. For customizing it, a Web developer can add a button to its bottom and modal in its body area.

jQuery UI offers the `dialog()` method for showing a dialog box.

Syntax

```
$(selector, context).dialog(opt);
$(selector, context).dialog({opt1: val1, opt2: val2..... });
```

Table 4.6 lists a few options that a Web developer can pass as parameters to the `dialog()` method.

Option	Description
`title`	Indicates the title text. The default value is null.
`autoOpen`	Opens the dialog box if the value is `true`, which is the default value. If false, it opens the dialog box only when a Web developer invokes dialog('open').
`closeOnEscape`	Closes the dialog box once a user presses the Esc key when the focus in on the dialog box if the value is `true`. This is the default value.
`buttons`	Includes buttons in the dialog box.
`draggable`	Allows dragging the dialog box by dragging its title bar if the value is `true`. This is the default value.
`resizable`	Allows resizing the dialog box in any direction if the value is `true`. This is the default value.
`modal`	Applies the behavior of a modal to the dialog box if the value is `true`. Other items on the page are disabled. The default value is `false`.
`position`	Sets the dialog box's initial position, which can be top, center, left, right, or bottom.

Table 4.6: Dialog Box Options

Figure 4.19 shows how to use the `dialog()` method in jQuery.

```html
<!doctype html>
<html lang = "en">
   <head>
      <meta charset = "utf-8">
      <link href = "https://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css"
       rel = "stylesheet">
      <script src = "https://code.jquery.com/jquery-1.10.2.js"></script>
      <script src = "https://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
      <script>
       $(function() {
          $( "#myDialog" ).dialog({
             autoOpen: false,
          });
          $( "#open" ).click(function() {
             $( "#myDialog" ).dialog( "open" );
          });
       });
    </script>
   </head>
   <body>
      <div id = "myDialog"
       title = "Dialog">jQuery Dialog!</div>
      <button id = "open">Open Dialog</button>
   </body>
</html>
```

Figure 4.19: Code for Using the `dialog()` Method in jQuery

In figure 4.19, the `<div>` and `<button>` elements have their respective dialogs. For `<div>`, the `dialog()` method has the `autoOpen` option set to `false`. This means that the dialog box will not open when the page is loading. When a user clicks the button, the `dialog()` method with the `open` parameter invokes the dialog box for the `<div>` element. This means that clicking the button will create and display the dialog box with the content, `jQuery Dialog!`.

Figure 4.20 shows the output upon clicking the button.

Figure 4.20: Output of Code Using the `dialog()` Method in jQuery

4.6 Datatable Plugin in jQuery

Datatable is an intuitive jQuery plugin that adds more functionality to the `<table>` element. It makes a plain HTML table dynamic by easily including the pagination, sorting, and searching utilities in it. This enables a developer to manage the display of huge quantity of data. Similarly, the plugin also enables a user to access the desired data quickly. The plugin is quite flexible to facilitate user-friendly interaction for an HTML table.

Following are the salient feature of datatables:

- Free and open-source
- Support for any data source including AJAX
- Built-in utilities such as instant search, pagination, and column-based ordering
- Built-in extensions such as fixed columns and buttons
- Built-in options such as `scrollY` and `order` along with callback functions
- Completely internationalizable
- Comprehensive documentation
- Hundreds of examples

For using the plugin, a Web developer needs to include only two additional files in the intended Webpage. They are the DataTables JS file and the DataTables CSS file. Web developers can either download the files from the Official Website or refer to a CDN. Code Snippet 5 shows the CDN links to these files, which a Web developer needs to add along with jQuery CDN link.

Code Snippet 5:

```
<link rel="stylesheet" type="text/css" href="//cdn.datatables.net/1.
10.13/css/jquery.dataTables.min.css">
<script type="text/javascript" charset="utf8" src="//cdn.datatables.
net/1.10.13/js/jquery.dataTables.min.js"></script>
```

Web developers can use datatables with a variety of data sources. A datatable can accept on the data of an array, an HTML table, or AJAX-retrieved data.

Syntax

```
$(selector).dataTable();
```

In this syntax, the jQuery selector is usually a reference to an HTML table, which a Web developer intends to enhance. The `datatable()` method can accept options as optional parameters, which form a configuration object of the dataTable constructor. Code Snippet 6 shows the usage of some of these options.

Code Snippet 6:

```
$('table').dataTable( {
  paginate: true,
  scrollY: 400
} );
```

In code snippet 6, pagination and scrolling both are enabled for the datatable.

Figure 4.21 shows how to use the `datatable()` method in jQuery.

```
1  <!DOCTYPE html>
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head runat="server">
4      <title></title>
5      <script
6    src="https://code.jquery.com/jquery-2.2.4.min.js"
7    integrity="sha256-ahbd1vqf/xrr9qja0nqlnivqPRLnCRrrXzXnHutelr44="
8    crossorigin="anonymous"></script>
9     <script
10   src="https://code.jquery.com/ui/1.12.1/jquery-ui.min.js"
11   integrity="sha256-VazP97ZCwtekAsvqPHPkNvyJ5UrPPRzrvDJuuf5GVYrahUgu="
12   crossorigin="anonymous"></script>
13     <link rel="stylesheet" type="text/css"
14       href="https://cdn.datatables.net/1.10.13/css/jquery.dataTables.min.css" />
15     <script src="https://cdn.datatables.net/1.10.13/js/jquery.dataTables.min.js">
16     </script>
17     <script type="text/javascript">
18       $(document).ready(function () {
19           $('#datatable').dataTable();
20       });
21     </script>
22 </head>
23 <body style="font-family: Arial">
24     <form id="form1" runat="server">
25         <div style="width: 500px; border: 1px solid black; padding: 5px">
26             <table id="datatable">
27                 <thead>
28                     <tr>
29                         <th>ID
30                         </th>
31                         <th>City
32                         </th>
33                          <th>Country
34                          </th>
35                          <th>Continent
36                          </th>
37                     </tr>
38                 </thead>
39                 <tbody>
40                     <tr>
41                         <td>1</td>
42                         <td>Monteria</td>
43                         <td>Colombia</td>
44                         <td>South America</td>
45                     </tr>
46                     <tr>
47                         <td>2</td>
48                         <td>Birmingham</td>
49                         <td>United Kingdom</td>
50                         <td>Europe</td>
51                     </tr>
52                     <tr>
53                         <td>3</td>
54                         <td>Bangalore</td>
55                         <td>India</td>
56                         <td>Asia</td>
57                     </tr>
58                 </tbody>
59             </table>
60         </div>
61     </form>
62 </body>
63 </html>
```

Figure 4.21: Code for Using the `datatable()` Method in jQuery

In figure 4.21, the `<head>` tag includes CDN references to jQuery, jQuery UI, and dataTables files. Within the `<script>` tag, the `datatable()` method is associated with the HTML table whose id is `datatable`. This table has three rows.

Figure 4.22 shows the output.

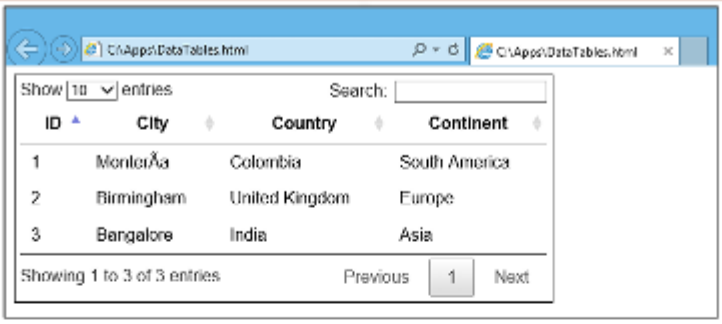


Figure 4.22: Output of Code Using the `datatable()` Method in jQuery

In figure 4.22, the HTML table features the search, pagination, and scrolling utilities. It is no longer a plain HTML table.

Quick Test 4.1

1. The autocomplete functionality is added directly in jQuery.

   True
   False

2. A callback function waits for an effect to occur completely.

   True
   False

Summary

- Built-in functions are methods that allow manipulating data values such as dates, numbers, and strings.
- The `live()` method delegates event handlers to existing and future elements that match with the selector.
- The jQuery AJAX methods allow requesting data from a distant Web server via LOAD, GET, and POST in different formats and load the same into the desired HTML element.
- A callback function runs once the current jQuery effect is over.
- The `$.get()` method asks data of a resource from a Web server through a GET request.
- The `$.post()` method asks data of a resource from a Web server through a POST request.
- The jQuery UI library offers the `autocomplete()` method to implment the autocomplete plugin.
- A widget refers to an object associated with an element on a Webpage for handling the lifespan, interaction, state, and inheritance and other aspects with other JS objects or widgets.
- The jQuery UI widgets are special plugins that simplify the task of applying functionality to their associated elements and offering some capabilities lacking in standard jQuery plugins.
- The Datatable jQuery plugin makes a plain HTML table dynamic by easily including the pagination, sorting, and searching utilities in it.

Check Your Progress

1. **_____ is a built-in jQuery function.**
   a. sortstr()
   b. pop()
   c. pushstr()
   d. tostr()
2. **Which of the following parameters are of the live() method?**
   a. Event listener
   b. data
   c. Callback function
   d. element
3. **The _____ object is a parameter of the load() function.**
   a. $.fn
   b. HttpRequest
   c. XMLHttpRequest
   d. html()
4. **Which of the following statements is true about $.get() method?**
   a. It can load any type of data.
   b. It always uses GET.
   c. It can cache the requested data.
   d. All of these.
5. **Which of the following statements about jQuery UI is true?**
   a. It is built upon AJAX.
   b. It offers the datatable functionality.
   c. It provides widgets.
   d. None of these.

Answers to Check Your Progress

1. `pop()`.
2. `data`.
3. `XMLHttpRequest`.
4. All of these.
5. It provides widgets.

Answers to Quick Test

Quick Test 4.1

1. False
2. True

SESSION 5 – Creating Bootstrap Responsive Layout

Session Objectives:

This session introduces the Bootstrap responsive layout along with the elements that contribute to it. It explains the responsive grid system and shows how to use it to create simple responsive Webpages. The session also describes the usage of jumbotron component and various typography classes in Bootstrap for transforming and styling text in HTML elements.

At the end of this session, you will be able to:

- *Describe how bootstrap helps in achieving responsiveness*
- *Explain the grid system*
- *Outline the use of Jumbotron component*
- *Describe the use of various typography classes*
- *Describe creation of simple responsiveness pages with Bootstrap*

5.1 Bootstrap Responsive Layout

The increased usage of mobile devices such as smartphones and tablets has made it indispensable to optimize Websites for them. In simple words, a Website should adjust to the screen size of any device to ensure proper viewing of the page. This is exactly what a responsive Web design offers.

It refers to a design process for providing ease of access and smooth viewing experience to the users regardless of the device in use. Such a design concept is desirable and an efficient way to display a Website on a variety of devices with least efforts.

A responsive layout adapts its user interface components and perhaps even functionality to any screen size on its own. This screen can be of a laptop, desktop, or a mobile phone. The width of the browser does not matter, as the layout responds according to it, as shown in figure 5.1.

Figure 5.1: Responsive Layout on Different Screens

In figure 5.1, there are four tabs in the middle in all devices, but its location differs as per the size of the device's screen. Changing of the layout as per size of the target screen facilitates an optimal viewing experience. This encompasses effortless reading and navigation on the target devices with minimum resizing, scrolling, and rotating.

A responsive Website adapts its layout to the viewing screen using three elements:

- Fluid, proportion-based grids
- Flexible or fluid images other multimedia
- CSS3 media queries (extension of `@media` rule) for applying suitable breaks to the page's content

It is challenging to design a Website that can change its environment as per its requested Webpage. This means that the grids and multimedia should be fluid and flexible. At this point, using media queries becomes essential to display the right assets in the right manner on the targeted devices.

A fluid grid handles sizing of a page element in relative units such as percentages, instead of absolute units such as points or pixels. Such relative sizes are also set for flexible multimedia so that they do not move outside their container. Media queries apply different CSS style rules as per targeted device's characteristics, more commonly being the browser's width.

Bootstrap 3 handles all these elements to give a solid foundation for designing and developing responsive Web applications. With its mobile first grid system, it has become easier for Web developers to build responsive and mobile-friendly applications.

There is no need of extra style sheet for activating the responsive functionality in Bootstrap 3.

Being mobile first means that the Bootstrap code begins by focusing first on smaller screens of devices such as tablets and smartphones. It then expands its components including the grids for larger screens such as desktops and laptops.

Version 3 comes with four classes of grids. These classes offer better control over the layout and over the manner in which the grid will render on different devices. Bootstrap media queries allow showing, hiding, and moving content based on the size of the viewport.

5.2 Working with Grids in Bootstrap

Bootstrap 3 comes with a robust mobile-first grid system for designing a responsive layout of any size and shape. A grid in Bootstrap is a collection of HTML/CSS components. This collection enables a Web developer to structure a Website and insert its contents easily in the intended locations.

An analogy of this grid system is a graph paper consisting of a series of vertical and horizontal lines. These lines intersect each other such that they create rectangular or square boxes. The same concept applies to the grid system that allows defining rows and columns to put the desired content in the resulting boxes.

A Web developer can create only up to 12 columns for a Webpage regardless of the viewport, but can have unlimited rows. This is why it is also known as a 12-grid system or 12-column layout, as shown in figure 5.2.

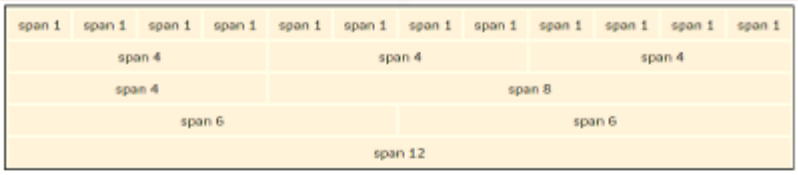


Figure 5.2: 12-column Grid Layout

In figure 5.2, it is clear that each row can only have up to 12 columns. Each grid column holds the number out of 12 available columns across which it should span. For example, each column in the second row of figure 5.2 spans up to four columns. This splits the row into a total of three columns, each holding four columns.

Note: If the number of columns exceeds 12, the additional columns shall stack irrespective of the viewport.

The grid system in Bootstrap is responsive due to which the columns tend to re-arrange as per the size of the screen. A layout with three columns looks better on a big screen. Similarly, a layout with stacked items looks better on a small screen.

The grid system consists of a series of containers, rows, and columns for designing a layout and aligning its content. The container element covers the content by setting the margins on the left and right. In case of a fixed container, the width differs as per the targeted device, which is shown in table 5.1.

Width of Device's Screen	Width of Fixed Container
1200 pixels and beyond	1170 pixels
992 to 1199 pixels	970 pixels
768 to 991 pixels	750 pixels
Under 768 pixels	Auto

Table 5.1: Different Fixed Widths

In case of a fluid container, its width is always equal to that of the targeted device. However, on the right and left sides, both fixed and fluid containers feature padding of 15 pixels.

A row extends from left to right edge of a container and wraps content around the columns. It eliminates the container's padding through a margin value of -15px on both sides. In simple words, the text in a row touches the container's left edge. A Web developer can create a row by specifying the `.row` class in a block-level element. For example, this element can be `<div>` within a container. For columns, there are various class prefixes for different screen sizes.

5.2.1 Predefined Grid Classes

Bootstrap 3 provides predefined grid classes for designing responsive layouts quickly for different devices. There are four such classes that form the four tiers of the grid system, which are as follows:

- `xs`, which is for extra small screens
- `sm`, which is for small screens
- `md`, which is for medium-sized screens
- `lg`, which is for large screens

A Web developer can user one or more of these classes to design a more dynamic and fluid layout. A Web developer can now show a Website with a different grid as per the varying browser size. Table 5.2 shows the column specifications for varying browser sizes.

Column Definition	Browser of Target Device	Browser Size
`.col-lg-*`	Large devices (desktops)	1200 pixels and up
`.col-md-*`	Medium devices (desktops)	992 pixels and up
`.col-sm-*`	Small devices (tablets)	768 pixels and up
`.col-xs-*`	Extra small (mobile phones)	< 768 pixels

Table 5.2: Column Specifications for Different Browser Sizes

In table 5.2, the * symbol represents the number of columns the resulting column in a row should cover. For example, to cover four columns out of 14 in just one column, a Web developer needs to specify `.col-xs-4`. Technically, each class cascades upwards. In simple words, if the widths for `xs` and `sm` are same, the Web developer needs to mention only `xs`.

5.2.2 Rules of Bootstrap Grid System

Bootstrap 3 has few rules to follow while using its grid system. Following are these rules:

- Containers act as a means to middle-align the contents of your site.
- Rows are horizontal series of columns for properly aligning the columns. For aligning and ensuring proper padding of rows, it is essential to place them within a container.
- Columns hold the content and only they can be the direct children of rows.
- Predefined grid classes for columns indicate the type of device and number of columns to include out of 12 for each row. For example, for having four columns of equal width in a row, a Web developer should specify `<div>` tags four times, each with `.col-sm-3` class. This means that each column in this row shall span up to 3 columns to cover all 12.
- Grid tiers depend on minimum widths, which means they are applicable to all tiers lying above the specified one. For example, applying `.col-sm-*` to an element affects its styling even on medium and large devices even if their classes are not included. This means that a column for smaller devices shall work in any device.
- Predefined grid classes for both rows and columns make grid layouts quickly.
- Columns have gaps between its content, which is known as gutters. They do so through padding, which is neutralized for the first and last column in a row via negative margin.

Table 5.3 shows the different grid options that work across several devices.

Options/Devices	Extra Small Devices	Small Devices	Medium Devices	Large Devices
Grid Behavior	Always horizontal	Collapsed, horizontal over the media query breakpoints	Collapsed, horizontal over the media query breakpoints	Collapsed, horizontal over the media query breakpoints
Container Width in Pixels	Auto	750	970	1170
Class	`.col-xs-*`	`.col-sm-*`	`.col-md-*`	`.col-lg-*`

Column Width in Pixels	Auto	62	81	97
Gutter Width in Pixels	30 (15 on each side)	30 (15 on each side)	30 (15 on each side)	30 (15 on each side)
Nestable	Yes	Yes	Yes	Yes

Table 5.3: Grid Options

5.2.3 Grid's Basic Structure

A grid structure for any layout has one or more containers, rows, and columns. A Web developer should start by defining a container that contains rows and columns. Then, within the container, the code should define rows using the .row class. Next, within a row, the Web developer should define columns using `.col-xs-*`, `.col-sm-*`, `.col-md-*`, and `.col-lg-*` classes. Finally, these columns are where the Web developer can place the content. Code Snippet 1 shows the basic grid structure in Bootstrap.

Code Snippet 1:

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

In code snippet 1, the code defines three rows within a container through `<div class="row">`. To each row, it adds the desired number of columns through suitable `.col-*-*` classes. The numbers in `.col-*-*` for a row should total up to 12.

5.2.4 Grids for Different Devices

A grid system differs as per the devices in terms of layout. In simple words, for each type of device, the grid system has a distinct split percentage for columns in a row. Following are the four types of grids, each suitable for different devices:

- Stacked or horizontal
- Small
- Medium
- Large

Stacked or Horizontal Grid

This grid system loads as a stack on small devices such as mobile phones prior to being horizontal on medium or large devices.

Figure 5.3 shows the code for a stacked-to-horizontal layout featuring two columns. These columns will split in the percentage ratio of 50:50 on all screens. However, on extra small screens, it will stack.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <h1>Grid System</h1>
  <p>col-sm-6 divides the page in to two equal halves</p>
  <p>Rescale the browser to see the changes</p>
  <div class="row">
    <div class="col-sm-6" style="background-color:green;">
      Bootstrap includes a powerful mobile-first flexbox grid system
      for building layouts of all shapes and sizes.
    </div>
    <div class="col-sm-6" style="background-color:red;">
      Responsive Web design is a Web design approach aimed at crafting sites
      to provide an optimal viewing experience
    </div>
  </div>
</div>

</body>
</html>
```

Figure 5.3: Code for Stacked-to-Horizontal Layout

Figure 5.4 shows the output wherein both the columns in a row are split in a percentage ratio of 50/50.

Figure 5.4: Output of Code for Stacked-to-Horizontal Layout

Grid Layout for Small Devices

Small devices possess a screen whose width is between 768 and 991 pixels. For these devices, Web developers should use the `.col-sm-*` classes. Assuming a layout having only two columns, a Web developer can split them into the percentage ratio of 75/25 or 25/75. Figure 5.5 shows 75/25 split.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <h1>Grid System</h1>
  <p>col-sm-9 col-sm-3 divides the page in to two 75:25 ratio</p>
  <p>Rescale the browser to see the changes</p>
  <div class="row">
    <div class="col-sm-9" style="background-color:green;">
      Bootstrap includes a powerful mobile-first flexbox grid system
     for building layouts of all shapes and sizes.
    </div>
    <div class="col-sm-3" style="background-color:red;">
      Responsive Web design is a Web design approach aimed at crafting sites
     to provide an optimal viewing experience
    </div>
  </div>
</div>

</body>
</html>
```

Figure 5.5: Code for Grid Layout for Small Devices

Figure 5.6 shows the output.

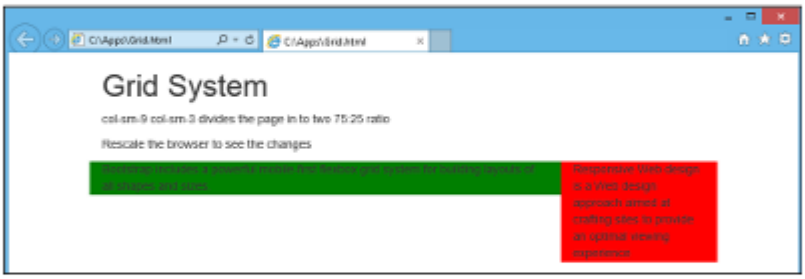


Figure 5.6: Output of Code for Grid Layout for Small Devices

Grid Layout for Medium Devices

Although a split of 25/75 for two columns looks fine for small devices, it is not an ideal choice for medium devices. For them, a split of 50/50 results in a better look, as they possess a screen whose width is between 992 and 1199 pixels. For these devices, a Web developer should specify the `.col-md-*` classes. Code Snippet 2 shows how to specify column widths for medium devices.

Code Snippet 2:

```
<div class="container-fluid">
  <h1>Hello World!</h1>
  <div class="row">
    <div class="col-sm-3 col-md-6" style="background-color:yellow;">
      <p>Medium Devices left..</p>
    </div>
    <div class="col-sm-9 col-md-6" style="background-color:pink;">
      <p>Medium Devices Right.</p>
    </div>
  </div>
```

```
</div>
```

In code snippet 2, for a two-column layout on medium devices, the code specifies `<div class="col-sm-3 col-md-6">` and `<div class="col-sm-9 col-md-6">`. These lines convey that `sm` is for display on small devices and `md` is for display on medium devices. The snippet creates a percentage split of 25/75 on small devices, as shown in figure 5.7 and 50/50 split on both large and medium ones. This layout stacks automatically on extra small devices.

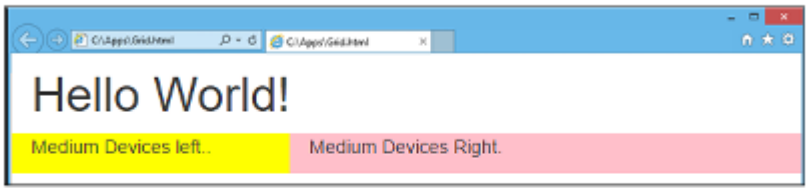


Figure 5.7: Output of Medium Grid Split on Small Devices

If a Web developer specifies only `.col-md-6` in both tags, the two columns will split in the ratio of 1:1 on medium and large devices. This is due to the scaling effect. On the other hand, on extra small devices, the layout shall vertically stack to have 100% width, as shown in figure 5.8.

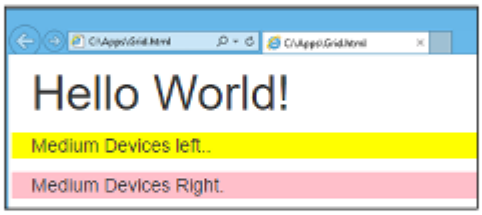


Figure 5.8: Output of Medium Grid Split on Extra Small Devices

Grid Layout for Large Devices

While 25/75 percentage split is for small devices and 50/50 split is for medium devices, a split of 33/66 is ideal for large devices. This is perhaps because the width of their screens is at least 1200 pixels. For these devices, a Web developer should specify the `.col-lg-*` classes. Code Snippet 3 shows how to specify column widths for large devices.

Code Snippet 3:

```
<div class="container-fluid">
  <h1>Hello World!</h1>
  <div class="row">
  <div class="col-sm-3 col-md-6 col-lg-4" style="background-
color:yellow;">
      <p>Large Devices</p>
    </div>
    <div class="col-sm-9 col-md-6 col-lg-8" style="background-
color:pink;">
      <p>Large Devices</p>
```

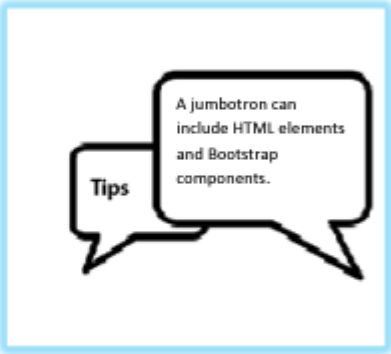
```
    </div>
  </div>
</div>
```

In code snippet 3, for a two-column layout on large devices, the `<div>` tags contain three classes, `col-sm-3`, `col-md-6`, and `col-lg-4`. The presence of these classes conveys that the page is set for display on all three types of devices. The snippet creates a percentage split of 25/75 on small devices, 50/50 on medium ones, and 33/66 on large ones.

In case a Web developer specifies only `.col-lg-6` class in both `<div>` tags, the two columns will split in the ratio of 1:1 on large devices. However, on small and medium devices, the layout shall vertically stack to have 100% width.

5.3 Working with Headers in Bootstrap

The page header layout component adds a header to a page. It separates it from the rest of the content by inserting a horizontal line under the heading. However, using this component would be insufficient for highlighting the heading text to grab the attention of visitors. In this situation, using the jumbotron component is a better option.



As the name suggest, this layout component allows increasing the size of heading text. It also allows adding much space to the content of a landing page. The component displays content in a big grey box featuring corners in the round shape for grabbing attention to some special details.

Jumbotron is a flexible and lightweight component that can extend over the whole viewport to highlight key informative content on a site.

To create jumbotron, a Web developer should specify the `.jumbotron` class in a `<div>` element. A Web developer simply needs to wrap the special content such as a description or a heading in `<div>` and apply this class to it. A `<div>` tag should contain the `.container` or `.container-fluid` class for appropriate padding as well as alignment.

There are two ways to use this component:

- Inserting it inside `<div class="container">` for preventing it to extend up to the edges of the screen
- Inserting it outside `<div class="container">` for extending it up to the edges of the screen or for having a full-width jumbotron

Figure 5.9 shows the code for preventing the jumbotron to extend up to the screen's edges.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
    <div class="jumbotron">
      <h1>Bootstrap headers</h1>
      <p>Bootstrap Header text.</p>
    </div>
    <p>Normal paratag1</p>
    <p>Noraml paratag2</p>
</div>

</body>
</html>
```

Figure 5.9: Code for Placing Jumbotron inside `<div>`

In figure 5.9, the `<div>` element holding the `.jumbotron` class is placed inside the `<div>` element that holds the `.container` class. This means that the jumbotron will be displayed inside this fixed container.

Figure 5.10 shows the output.

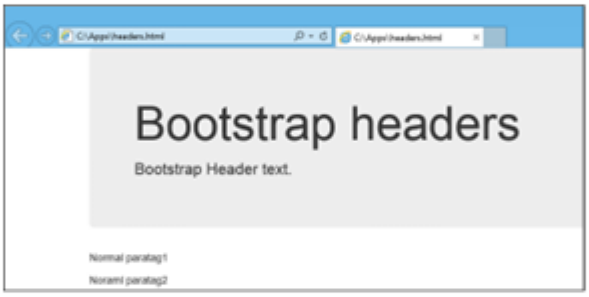


Figure 5.10: Output of Code Placing Jumbotron inside `<div>`

In figure 5.10, the `<div>` element within the container aligns the jumbotron to the Webpage. The jumbotron is inside the fixed container due to which it does not extend up to the edges of the screen. This is evident from the gap between the left side of the browser and jumbotron.

Figure 5.11 shows the code for extending the jumbotron up to the screen's edges.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="jumbotron">
    <h1>Bootstrap Header outside container</h1>
    <p>Bootstrap Header text.</p>
</div>
<div class="container">
    <p>Normal para tag1</p>
    <p>Normal para tag2</p>
</div>


</body>
</html>
```

Figure 5.11: Code for Placing Jumbotron Outside `<div>`

In figure 5.11, the jumbotron is placed outside and before the container. Thus, it will extend up to the edges. Figure 5.12 shows the output.

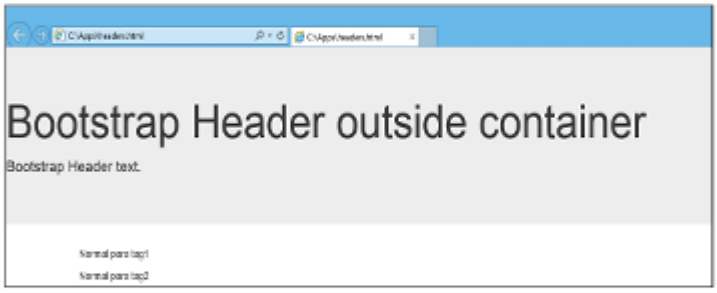


Figure 5.12: Output of Code Placing Jumbotron Outside `<div>`

5.4 Typography in Bootstrap

Bootstrap offers typography classes and elements that style a few HTML elements in a slightly different way from the browsers. These are customized typography options available for making text in headings, body, and lists more stylish.

By default, the size of font in Bootstrap is 14 pixels with 1.428 as the height of a line. These global settings are applicable to `<body>` and `<p>` elements. Moreover, all paragraphs possess a bottom margin that is half of their line height. Their bottom margin comes to 10 pixels by default.

5.4.1 Headings

For all headings of HTML, Bootstrap has corresponding `.h1` to `.h6` classes. These classes are useful for matching a heading's font style and accommodate inline text besides the heading text.

Figure 5.13 shows how Bootstrap styles the HTML headings.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
   <h1>h1 Text Transformation (36px)</h1>
   <h2>h2 Text Transformation (30px)</h2>
   <h3>h3 Text Transformation (24px)</h3>
   <h4>h4 Text Transformation (18px)</h4>
   <h5>h5 Text Transformation (14px)</h5>
   <h6>h6 Text Transformation (12px)</h6>
</div>


</body>
</html>
```

Figure 5.13: Code for Styling HTML Headings

In figure 5.13, the code includes all H1 to H6 headings just as a Web developer specifies in HTML. Each of these headings is set as per its pixel size.

Figure 5.14 shows the output.

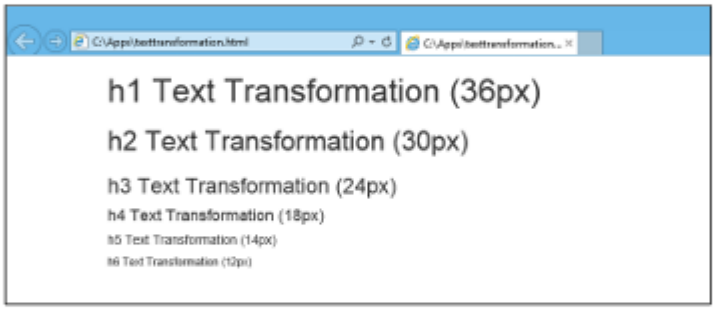


Figure 5.14: Output of Code for Styling HTML Headings

In any of these headings, a Web developer can insert an inline sub-heading using the `.small` class. An inline sub-heading allows displaying secondary text in smaller size and lighter color. Code Snippet 4 shows the use of `.small` class.

Code Snippet 4:

```
<div class="container">
  <h1>h1 Text Transformation <small>subordinate text</small></h1>
  <h2>h2 Text Transformation <small>subordinate text</small></h2>
  <h3>h3 Text Transformation <small>subordinate text</small></h3>
  <h4>h4 Text Transformation <small>subordinate text</small></h4>
  <h5>h5 Text Transformation <small>subordinate text</small></h5>
  <h6>h6 Text Transformation <small>subordinate text</small></h6>
</div>
```

Figure 5.15 shows how `.small` class renders its style.

Figure 5.15: Usage of `.small` Class

5.4.2 Highlighting, Abbreviating, and Referencing Text

Bootstrap also styles the `<mark>`, `<abbr>`, and `<blockquote>` elements of HTML. Figure 5.16 shows how it styles these elements.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>
<mark>
<div class="container">
  <h1>Highlighting Text</h1>
  <p>Mark element is used to <mark>point out</mark> specific section</p>
</div>
<abbr>
<div class="container">
  <h1>Abbreviations</h1>
  <p>The abbr element is used to mark up an abbreviation or acronym:</p>
  <p>The <abbr title="Indian Space Research Organisation">ISRO</abbr> was founded in 1969.</p>
</div>
<blockquote>
<div class="container">
  <h1>Blockquotes</h1>
  <p>The blockquote element is used to present content from another source:</p>
  <blockquote>
    <p>The Indian Space Research Organisation is the space agency of the Government of India
    headquartered in the city of Bangalore</p>
    <footer>From ISRO's website</footer>
  </blockquote>
</div>
</body>
</html>
```

Figure 5.16: Code for Highlighting, Abbreviating, and Referencing Text in Bootstrap

In figure 5.16, the codes uses the <mark>, <abbr>, and <blockquote> as Web developers use them in a standard HTML file. Figure 5.17 shows the output.

Figure 5.17: Output of Code for Highlighting, Abbreviating, and Referencing Text in Bootstrap

In figure 5.17, the output shows the highlighted text that is in the <mark> tag. It highlights the text using a background color, which is in a different shade than in HTML.

Similarly, the text in <abbr> is displayed with a dotted underline, unlike in HTML. When the user scrolls over the abbreviation, the full form given in the title attribute is displayed. Unlike in HTML, Bootstrap shows the help cursor when the user points it to the underlined text. The help cursor is a standard cursor with a small icon of question mark.

The text in <blockquote> is referenced by the text mentioned in <footer>. This is different from the standard HTML output wherein there is no footer.

5.4.3 Styling Description Lists, Inline Code Snippets, and Keyboard Input Text

Bootstrap also styles the <dl>, <code>, and <kbd> elements of HTML. Figure 5.18 shows how it styles these elements.

Figure 5.18: Code for Styling a Description List, Inline Snippet, and a Keyboard Input

Figure 5.19 shows the output.

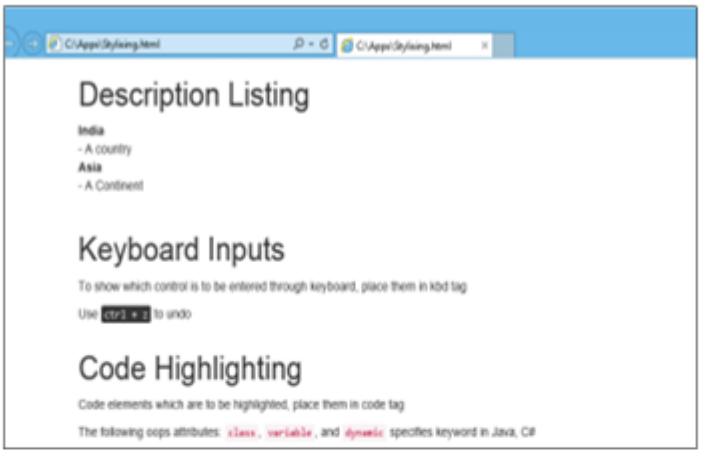


Figure 5.19: Output of Code for Styling a Description List, Inline Snippet, and a Keyboard Input

In figure 5.19, Bootstrap styles the `<dl>` element and its content by applying the bold style to the content in `<dt>`. This is something that HTML does not do. Similarly, unlike in HTML, Bootstrap highlights the text in `<code>` by giving it a background color. It also changes its font, which is visible even in a standard HTML output. Similarly, the keyboard input text in `<kbd>` gets a new font, black background color, and white text color. In HTML output, only the font is changed.

5.4.4 Transforming Text Through Text Transformation Classes

Bootstrap provides three text transformation classes for changing the case of text in components, which are as follows:

- `.text-uppercase`: Renders each letter in a text in upper case.
- `.text-lowercase`: Renders each letter in a text in lower case.
- `.text-capitalize`: Renders a text in which the first letter of each word is in upper case.

Figure 5.20 shows how to use these classes in Bootstrap.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Bootstrap Example</title>
5    <meta charset="utf-8">
6    <meta name="viewport" content="width=device-width, initial-scale=1">
7    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
8    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
9    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10 </head>
11 <body>
12
13 <div class="container">
14   <p class="text-lowercase">CONVERT IN TO LOWERCASE.</p>
15 <p class="text-uppercase">convert in to uppercase</p>
16 <p class="text-capitalize">capitalise me.</p>
17 </div>
18
19 </body>
20 </html>
```

Figure 5.20: Code for Using Text Transformation Classes

In figure 5.20, the text transformation classes are used within <p> tags.

Figure 5.21 shows the output.

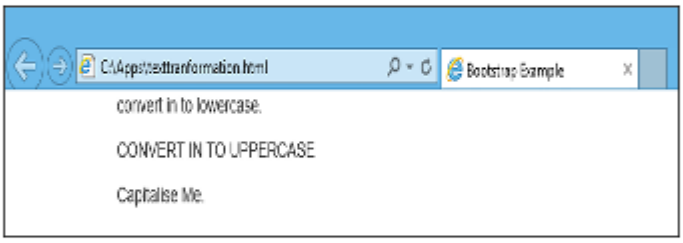


Figure 5.21: Output of Code for Using Text Transformation Classes

5.5 Responsive Web Designing in Bootstrap

Bootstrap uses the concept of grids to fit a Webpage properly on different devices for ensuring a consistent browsing experience. Let's now use the grid system along with some components to create a simple and responsive Webpage in Bootstrap.

Figure 5.22 shows a responsive Webpage that uses predefined grid classes and components such as navbar, jumbotron inside a container, and footer.

```
<!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">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity=
"sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity=
"sha384-rHyoN1iZaVOfW+2/kP+S7qzz/I3UqTq2r0Q/BvRKI9ZfyXqkMNpdaZ1ArA9U7ZC+WF9Wscg4W+Ng+rr4Ig" crossorigin="anonymous">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity=
"sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUfjllgmyghQksz74MpoVhUqNK6ZqhJ6YZ" crossorigin="anonymous"></script>
</head>
<body>
<nav id="myNavbar" class="navbar navbar-default navbar-inverse navbar-fixed-top" role="navigation">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#navbarCollapse">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">Website created with Bootstrap</a>
        </div>
        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse" id="navbarCollapse">
            <ul class="nav navbar-nav">
                <li class="active"><a href="" target="_blank">Home</a></li>
                <li><a href="" target="_blank">About</a></li>
                <li><a href="" target="_blank">Contact</a></li>
            </ul>
        </div>
    </div>
</nav>
<div class="container">
    <div class="jumbotron">
        <h1>Responsive Website</h1>
        <p><a href="" target="_blank"></a>Responsive web design is a process of designing and building websites to provide better
        accessibility and optimal viewing experiences to the user. </p>
        <p><a href="" target="_blank" class="btn btn-success btn-lg">Learn More</a></p>
    </div>
    <div class="row">
        <div class="col-sm-6 col-md-4 col-lg-2">
            <h2>Jumbotron</h2>
            <p>A jumbotron indicates a big box for calling extra attention to some special content or information. A jumbotron is displayed
            as a grey box with rounded corners. It also enlarges the font sizes of the text inside it.</p>
        </div>
        <div class="col-sm-6 col-md-4 col-lg-2">
            <h2>Navigation</h2>
            <p>A navigation bar is a navigation header that is placed at the top of the page</p>
        </div>
        <div class="clearfix visible-sm-block"></div>
        <div class="col-sm-6 col-md-4 col-lg-2">
            <h2>Header</h2>
            <p>A page header is like a section divider. The .page-header class adds a horizontal line under the heading (+ adds some extra
            space around the element)</p>
        </div>
        <div class="clearfix visible-md-block"></div>
        <div class="col-sm-6 col-md-4 col-lg-2">
            <h2>Badges</h2>
            <p>Badges are canonical indicators of how many items are associated with a link</p>
        </div>
    </a>
    <div class="row">
        <div class="col-sm-12">
            <footer>
                <p>Footer Text</p>
            </footer>
        </div>
    </div>
</div>
</body>
</html>
```

Figure 5.22: Bootstrap Code for Responsive Webpage

In figure 5.22, the code creates a default navbar containing `Home`, `About`, and `Contact` links. Then, it defines a container holding the jumbotron component with a large, successful-style button. These button attributes are as per the `btn-success` and `btn-lg` classes. Inside the container, `.row` class creates a row that is split into four columns. The value `col-sm-6 col-md-4 col-lg-2` for each of the four columns spans across 12 columns (6 + 4 + 2). These numbers represent a portion of the total width of a `div`. In other words, `col-sm-6` spans up to 6 columns out of 12, `col-md-2` spans up to 4, out of 12, and `col-lg-2` spans up to 2 columns out of 12. Each row has a header and a paragraph. One more row spanning across all 12 columns defines a footer text.

Figure 5.23 shows the output.

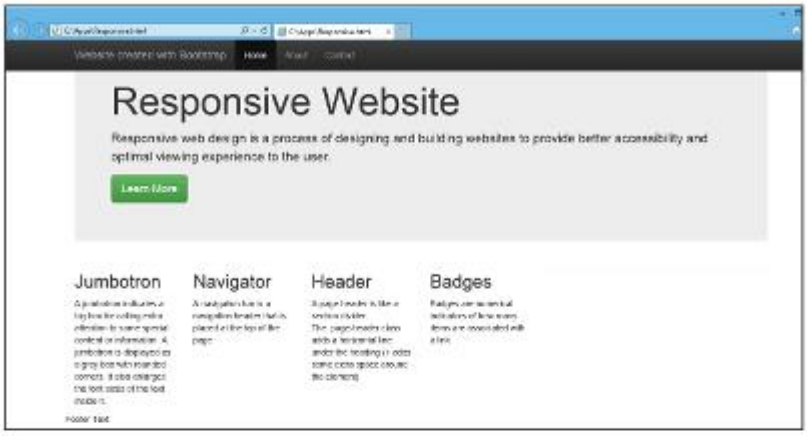


Figure 5.23: Output of Code for Responsive Webpage

In figure 5.23, out of four columns, the third and fourth one in the row use the `.clearfix` class to prevent uneven wrapping on medium devices. Figure 5.24 shows uneven wrapping on medium devices.

Figure 5.24: Output of Code for Responsive Webpage

Figure 5.25 shows how `.clearfix` class organizes the content on a medium-sized device.

Figure 5.25: Output of Using `.clearfix` on Medium Devices

In figure 5.22, the third and fourth columns in the first row also use the `.visible-sm-block` and `.visible-md-block` classes. Well, these are responsive utilities useful for showing the content through media queries as per the targeted device. The `.visible-sm` responsive utility makes the content visible on small devices. On the other hand, `.visible-md` ensures the same on the screens of medium devices. The block part displays the style of `display:block` (CSS property). Similarly, there are `.visible-xs` and `.visible-lg` classes. A Web developer can use one or more of these utilities for toggling content across different viewport breakpoints defined by media queries.

Figure 5.26 shows the output on a small device in absence of `.visible-sm-block`.

Figure 5.26: Output on a Small Device in Absence of `.visible-sm-block`

Figure 5.27 shows the output on a medium device in absence of `.visible-md-block`.

Figure 5.27: Output on a Medium Device in Absence of `.visible-md-block`

Quick Test 5.1

1. There can be unlimited rows as well as column in a Bootstrap layout.

   True
   False

2. Bootstrap allows capitalizing all letters in a text.

   True
   False

Summary

- A responsive layout adapts its user interface components and perhaps even functionality to any screen size on its own.
- A responsive Website adapts its layout to the viewing screen using three elements namely, fluid grids, flexible multimedia, and CSS3 media queries.
- Bootstrap 3 mobile first technology allows the code to focus first on smaller screens and then, expand for fitting on larger screens.
- A grid in Bootstrap is a collection of HTML/CSS components that enable a Web developer to structure a Website and insert its contents easily in the intended locations.
- Bootstrap provides only up to 12 columns for a Webpage regardless of the viewport, but can have unlimited rows.
- The grid system consists of a series of containers, rows, and columns for designing a layout and aligning its content. The container element covers the content by setting the margins on the left and right.
- Bootstrap offers four predefined classes namely, `xs`, `sm`, `md`, and `lg` for designing responsive layouts quickly as per the targeted devices.
- The jumbotron component allows increasing the size of heading and other text for grabbing quick attention of visitors.
- Inserting jumbotron inside a `<div>` tag prevents it to extend up to the edges of the screen, while placing it outside extends it up to the edges.
- Bootstrap offers `.h1` to `.h6` classes for styling heading text and accommodating inline text.
- Bootstrap provides `.text-uppercase`, `.text-lowercase`, and `.text-capitalize` classes for transforming the case of words in any piece of text.

Check Your Progress

1. **Which of the following values of the class attribute in div class will result in three columns spanning up to 12 columns in a row?**
   a. `col-xs-5, .col-xs-4, .col-xs-5`
   b. `col-xs-5, .col-xs-5, .col-xs-2`
   c. `col-xs-4, .col-xs-4, .col-xs-2`
   d. None of these
2. **Which of the following statements is true about a responsive layout in Bootstrap?**
   a. It has predefined classes.
   b. Columns tend to hold the content.
   c. It comes with media queries for hiding content as per the viewport's size.
   d. All of these.
3. **The `.col-md-*` class targets _____.**
   a. Medium devices of size 1000 pixels and up
   b. Medium devices of size 950 pixels and up
   c. Medium devices of size 900 pixels and up
4. **The jumbotron component can extend over _____.**
   a. Any element
   b. whole viewport
   c. Any text
   d. Modal
5. **Which of the following elements allows adding an inline text in a .h1 header?**
   a. `<small>`
   b. `<abbr>`
   c. `<dl>`
   d. None of these

Answers to Check Your Progress

1. `col-xs-5, .col-xs-5, .col-xs-2`
2. All of these
3. Medium devices of size 992 pixels and up
4. whole viewport
5. `<small>`

Answers to Quick Test

Quick Test 5.1

1. False
2. True

SESSION 6- Creating Responsive Webpages Using Bootstrap and jQuery

Session Objectives:

This session introduces the role of media queries in making a Bootstrap Webpage responsive. It also explains how to use Bootstrap tables and wells for organizing the content with the desired custom styles. The session also describes how to customize Bootstrap by overriding its default CSS and how to use both jQuery and Bootstrap for making dynamic Webpages.

At the end of this session, you will be able to:

- *Describe the role of media queries in achieving responsiveness in Bootstrap*
- *Understand the use tables in Bootstrap*
- *Understand the use tabs in a well in Bootstrap*
- *Describe how to customize Bootstrap*
- *Explain how to create a Website using Bootstrap and jQuery*

6.1 Role of Media Queries in Responsive Layouts

A media query defines a style rule for displaying a Webpage on a specific type of device (media) without changing the markups. It helps in adding responsive CSS to a Bootstrap application. In CSS2, it allows specifying styles for computer screens, TV-type gadgets, handheld devices, and printers. In CSS3, media queries extend the concept of CSS2 media types. They now focus on the characteristics of the targeted media instead of focusing on its type. The queries are useful in determining the device type displaying a Webpage as well as its physical properties, which are:

- Viewport's and device's width and height
- Screen resolution
- Orientation such as portrait or landscape display

Using media queries allows rendering a custom style sheet on a tablet, Android phone, or iPhone. In short, they help in rendering a device-specific style sheet. Visibly, media queries define the breakpoint for the media's width from where the layout changes on a specific screen. The breakpoint is the pixel width at which the Webpage's layout needs to adjust for a better display on the targeted screen, as shown in figure 6.1.

Figure 6.1: Effect of a Breakpoint on a Mobile Screen

In figure 6.1, when the screen is smaller such as on a mobile device whose size is less than 768 pixels, each column expands to have a width of 100%.

Bootstrap defines its media queries in **bootstrap-responsive.css** file located in the **css** folder under **assets**. It specifies breakpoints in a grid system. Code Snippet 1 shows how bootstrap media queries implement the mobile-first technology.

Note: A Web developer can insert several breakpoints as per the requirements.

Code Snippet 1:

```
/* Portrait phones and smaller */
@media (max-width: 480px) {}

/* Landscape phones and portrait tablets */
@media (max-width: 767px) {}

/* Portrait tablets and small desktops */
@media (min-width: 768px) and (max-width: 991px) {}

/* Landscape tablets and medium desktops */
@media (min-width: 992px) and (max-width: 1199px) {}

/* Large desktops and laptops */
@media (min-width: 1200px) {}
```

In Code Snippet 1, the media queries start by targeting the small devices or media before any other devices. This is how mobile-first technology is implemented to display Webpages faster on smaller screens.

Here, a media query contains the `@media` attribute and one or more width-based logical expressions. The expression, when matches with the screen resolution of a specific device, returns `true` or else it returns `false`. A Web developer can specify a block of CSS properties or rules, which should apply only if the width condition is true.

The line stating `@media (max-width: 480px)` allows applying styles to layouts on devices having 480 pixels as maximum width. Similarly, `@media (min-width: 768px) and (max-width: 991px)` allows applying styles to layouts on devices whose screens have a width that is equal to or more than 768 pixels and less than or equal to 991 pixels. In short, the style sheet shown in Code Snippet 1 stores styles as per the minimum and maximum widths of the targeted screens using two properties namely, `min-width` and `max-width`.

The benefit of this mechanism is that it facilitates targeting the different ranges of screen size instead of a specific type of device. This means that when a new model of tablet or smartphone is released, Web developers are sure that their Websites shall appear precisely on the new screen. This is because the device's screen size exists in the covered range.

Figure 6.2 shows how to use a mobile-first media query within a Bootstrap application to deliver device-specific styles.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<style>
body {
    background-color: lightyellow;
}

@media only screen and (max-width: 768px) {
    body {
        background-color: lightpink;
    }
}
</style>
</head>
<body>
<p>Rescale the browser window. Lightpink is displayed when width of the screen is 768px and lightyellow is displayed in other cases.</p>
</body>
</html>
```

Figure 6.2: Code for Using a Mobile-first Media Query in Bootstrap

In figure 6.2, the `@media` attribute contains a media type value set to `screen` and a media query or expression within parentheses. This query is a logical expression for checking whether the width of the user's current screen is less than or equal to `768` pixels. It displays the background in `light yellow` when the width is less than `768` pixels and `light pink` when it is `768` pixels or beyond.

Figure 6.3 shows the output on a small device.

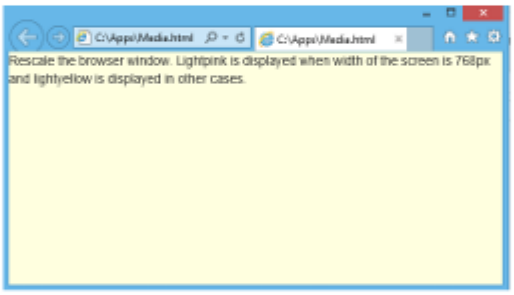


Figure 6.3: Output of Mobile-first Media Query on a Small Device

Figure 6.4 shows the output on a large device.

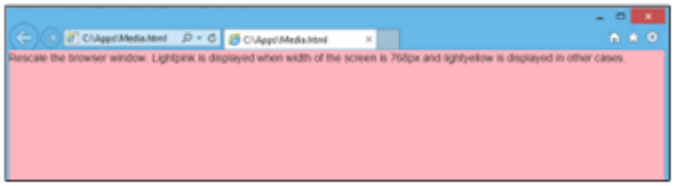


Figure 6.4: Output of Mobile-first Media Query on a Large Device

Instead of changing styles, once the width becomes smaller than 768 pixels, Web developers change the layout and once the width becomes bigger than 768 pixels. This is how this Webpage renders quickly on a mobile device, as the processing of media query takes place only on wider screens.

A Web developer can include styles for different devices within a single Webpage. Each of these styles has its own media query for displaying the Webpage differently as per the changing resolutions.

Usually, specifying three blocks of `@media` attribute is common: One for mobile phones and tablets (sm), one for lower-resolution desktops (md), and one for higher-resolution screens (lg). The code in figure 6.2 would apply the light pink background color to all devices having a resolution greater than 768 pixels. So, this is an ideal example of how to have a single block of style for devices with bigger screens.

A Web developer can also deliver a device-specific style sheet using a media query. Code Snippet 2 shows how to do so.

Code Snippet 2:

```
<link rel="stylesheet" media="screen and (max-width: 768px)" href="mobile-devices.css">
<link rel="stylesheet" media="screen and (min-width: 768px)" href="large-devices.css">
```

This is how you can add a style sheet link for each device type to a Webpage, each with its own media query. The style sheet in Code Snippet 2 would be used on all devices having a resolution smaller than 768 pixels. Thus, this is an ideal example of how to target mobile and tablet devices with a single style sheet, and large devices such as desktops with a different style sheet.

6.2 Tables in Bootstrap

An HTML table displays data in a grid layout that consists of rows and columns. Bootstrap helps in improving the appearance of an HTML table by providing built-in CSS classes. A basic Bootstrap table features distinct appearance in terms of only horizontal splitters and slight padding of 8 pixels. For applying this basic style, Bootstrap offers the `.table` class.

Figure 6.5 shows the basic Bootstrap table.

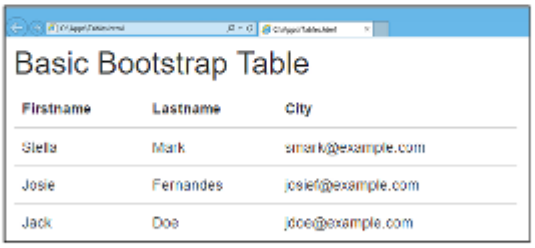


Figure 6.5: Basic Bootstrap Layout

Bootstrap provides a clean layout for creating a table. Table 6.1 lists the table elements that Bootstrap supports.

Element	Description
`<table>`	Is a wrapper for displaying data in rows and columns.
`<thead>`	Is a container holding the table's column headers in single or

Element	Description
	multiple <tr> tags.
<tbody>	Is a container holding the table's body in single or multiple <tr> tags.
<tr>	Is a container defining a row, which holds one or more table cells created through <td> or <th>.
<td>	Is a standard table cell with text that is left-aligned.
<th>	Is a special cell having header data that is middle-aligned and bold. It exists only within <thead>.
<caption>	Describes a caption for a table or a summary of what the table has. It must come immediately after <table>. Only one caption can exist in a table.

Table 6.1: Supported Table Elements

The standard order of table tags is <table> → <tbody> → <tr> → <td>. In case of column headers, the order shall be <thead> → <tr> → <th>.

6.2.1 Striped Rows in a Bootstrap Table

A Web developer can create a table with alternate rows having a background color differing from its previous one. It gives an appearance of zebra stripes. To give this effect to a table, Bootstrap offers the .table-striped class, which a Web developer needs to add to the .table base class.

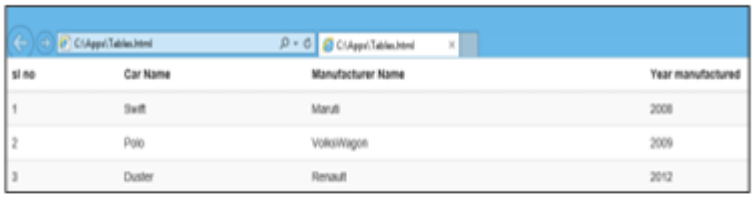
Figure 6.6 shows the code for creating a table with striped rows.

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1">
6       <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
7       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
8       <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
9   </head>
10  <body>
11  <table class="table table-striped">
12      <thead>
13          <tr>
14              <th>sl no</th>
15              <th>Car Name</th>
16              <th>Manufacturer Name</th>
17              <th>Year manufactured</th>
18          </tr>
19      </thead>
20     <tbody>
21          <tr>
22              <td>1</td>
23              <td>Swift</td>
24              <td>Maruti</td>
25              <td>2008</td>
26          </tr>
27          <tr>
28              <td>2</td>
29              <td>Polo</td>
30              <td>VolksWagon</td>
31              <td>2009</td>
32          </tr>
33          <tr>
34              <td>3</td>
35              <td>Duster</td>
36              <td>Renault</td>
37              <td>2013</td>
38          </tr>
39      </tbody>
40  </table>
41  </body>
42  </html>
```

Figure 6.6: Code for Table with Striped Rows

In figure 6.6, the value of class in `<table>` is set to `table table-striped`. This transforms a basic Bootstrap table into a striped format.

Figure 6.7 shows the output.



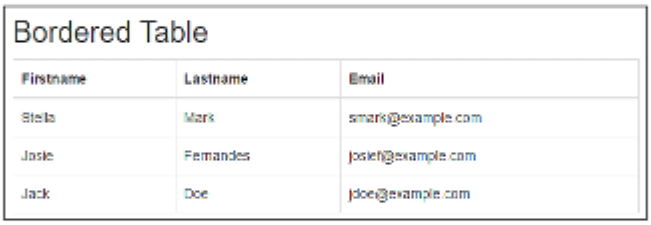
si no	Car Name	Manufacturer Name	Year manufactured
1	Swift	Maruti	2008
2	Polo	Volkswagon	2009
3	Duster	Renault	2012

Figure 6.7: Output of Code for Table with Striped Rows

6.2.2 Bordered Tables

By default, a Bootstrap table only has horizontal dividers. This means that there are only horizontal borders. However, Bootstrap enables adding borders on all sides of a table. It does so by offering the `.table-bordered` class, which should be added to the base class.

Figure 6.8 shows a table with a border on all sides.



Bordered Table

Firstname	Lastname	Email
Stella	Mark	smark@example.com
Josie	Fernandes	josiefr@example.com
Jack	Doe	jdoe@example.com

Figure 6.8: Table with a Border on All Sides

6.2.3 Hover Effect on Rows

Bootstrap allows Web developers to apply a hover effect to table rows. The row on which a user moves the mouse shows a gray background. To apply a hover effect, Bootstrap offers the `.table-hover` class, which should be added to the base class.

Figure 6.9 shows a table with rows showing the hovering effect.

Hover Effect on a Table

Firstname	Lastname	Email
Stella	Mark	smark@example.com
Josie	Fernandes	josiefr@example.com
Jack	Doe	jdoe@example.com

Figure 6.9: Table with Hovering Effect on a Row

6.2.4 Condensed or Compact Tables

Bootstrap allows making a table more condensed by reducing cell padding by half. This saves space around the cells. To make a table more compact, Bootstrap offers the `.table-condensed` class, which should be added to the base class.

Figure 6.10 shows a compact table.

Compact Table

Firstname	Lastname	Email
Stella	Mark	smark@example.com
Josie	Fernandes	jcsief@example.com
Jack	Doe	jdoe@example.com

Figure 6.10: Compact Table

6.2.5 Emphasis on Table Rows

Bootstrap offers a few contextual classes for emphasizing a cell or a complete row. They indicate success, warning, info, or danger by changing the color of the background of a row or a cell. Table 6.2 lists the contextual classes, which are also known as `<tr>`, `<th>`, and `<td>` classes, as they color the cells or rows.

Contextual Class	Background Color	Description
`.active`	Gray	Applies the hover color to a cell or row.
`.success`	Green	Denotes a successful action.
`.info`	Blue	Denotes informative text or neutral action.
`.warning`	Yellow	Denotes a warning to grab quick attention.
`.danger`	Red	Denotes a harmful action.

Table 6.2: Contextual Classes for Tables

Figure 6.11 shows how to use contextual classes in a table.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>
<div class="bs-example">
    <table class="table">
        <thead>
            <tr>
                <th>Traffic Signals sl no:</th>
                <th>Color</th>
                <th>Status</th>
            </tr>
        </thead>
        <tbody>
          <tr class="success">
                <td>1</td>
                <td>Green</td>
                <td>Free to go</td>
            </tr>
            <tr class="warning">
                <td>2</td>
                <td>Yellow</td>
                <td>Wait for Green</td>
            </tr>
            <tr class="danger">
                <td>3</td>
                <td>Red</td>
                <td>Stop</td>
            </tr>
        </tbody>
    </table>
</div>
</body>
</html>
```

Figure 6.11: Code for Using Contextual Classes in a Table

Figure 6.12 shows the output.

Traffic Signals sl no:	Color	Status
1	Green	Free to go
2	Yellow	Wait for Green
3	Red	Stop

Figure 6.12: Output of Code Using Contextual Classes in a Table

6.2.6 Responsive Tables

In Bootstrap 3, a Web developer can quickly create a responsive table. In other words, Bootstrap 3 activates horizontal scrolling on a small device whose screen width is not more than 768 pixels. Nevertheless, the same table on other devices with a larger screen will not show any difference.

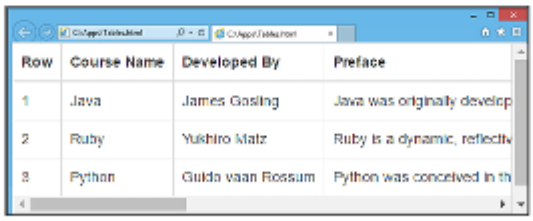
Bootstrap offers the `.table-responsive` class for making a table responsive. Figure 6.13 shows how to create a responsive table.

```
<!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">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity=
"sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity=
"sha384-rHyoN1iZ4vnfsx0QkqkQvZyjk3BqnUW2iJ0GxUvub05OAcQ0QMxtPrVYdd47iI2x" crossorigin="anonymous">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity=
"sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUfHRa4GziQtgbK7WWeKWTwp6RMUVf5Per" crossorigin="anonymous"></script>
</head>
<body>
<div class="bs-example">
    <div class="table-responsive">
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th>Row</th>
                    <th>Course Name</th>
                    <th>Developed By</th>
                    <th>Preface</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>1</td>
                    <td>Java</td>
                    <td>James Gosling</td>
                    <td>Java was originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun
                    Microsystems' Java platform.</td>
                <tr>
                    <td>2</td>
                    <td>Ruby</td>
                    <td>Yukihiro Matz</td>
                    <td>Ruby is a dynamic, reflective, object-oriented, general-purpose programming language. It was designed and developed
                    in the mid-1990s by Yukihiro "Matz" Matsumoto in Japan.</td>
                </tr>
                <tr>
                    <td>3</td>
                    <td>Python</td>
                    <td>Guido vaan Rossum</td>
                    <td>Python was conceived in the Late 1980s, and its implementation began in December 1989 by Guido van Rossum at Centrum
                    Wiskunde & Informatica (CWI)</td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
</body>
</html>
```

Figure 6.13: Code for Creating a Responsive Table

In figure 6.13, the `<div>` tag has `.table-responsive` class to make the table responsive. With it, another `<div>` tag is defined with the `.table-bordered` class, which is added to the base class. This gives a border on all sides of the responsive table.

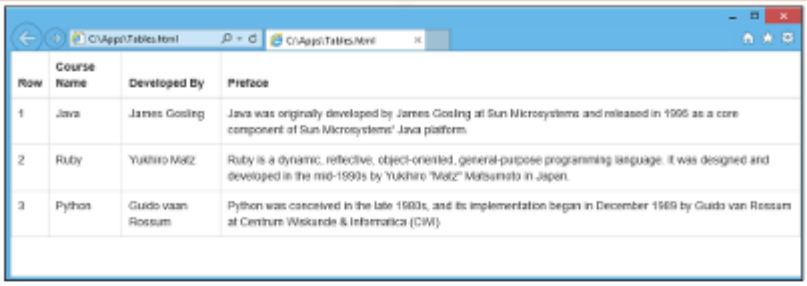
Figure 6.14 shows the output on a small screen having a horizontal scroll bar.



Row	Course Name	Developed By	Preface
1	Java	James Gosling	Java was originally develop
2	Ruby	Yukihiro Matz	Ruby is a dynamic, reflectiv
3	Python	Guido vaan Rossum	Python was conceived in th

Figure 6.14: Output of a Responsive Table on a Small Screen

Figure 6.15 shows the output on a big screen.



Row	Course Name	Developed By	Preface
1	Java	James Gosling	Java was originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform.
2	Ruby	Yukihiro Matz	Ruby is a dynamic, reflective, object-oriented, general-purpose programming language. It was designed and developed in the mid-1990s by Yukihiro "Matz" Matsumoto in Japan.
3	Python	Guido vaan Rossum	Python was conceived in the late 1980s, and its implementation began in December 1989 by Guido van Rossum at Centrum Wiskunde & Informatica (CWI).

Figure 6.15: Output of a Responsive Table on a Big Screen

6.3 Customizing Bootstrap

Twitter Bootstrap is now a popular CSS framework for developing Web applications, as it saves much time of Web developers. However, most developers do not make adequate efforts to customize their Bootstrap sites. This results in similar looking Bootstrap Websites.

In fact, it is easy to make a Bootstrap Website stand out from using a few premium themes. These themes do not retain the defaults; they contain custom components. Instead of paying for these premium themes, why not put some effort to customize a Website and make it stand apart?

The Official Website of Bootstrap offers a custom builder for generating a personal Bootstrap package. It also offers the LESS and SASS versions for compiling the CSS file as per the requirements. However, these options do not suit all Web developers, as it involves tracking all possible changes. Well, this is an additional as well as a tedious job, especially while upgrading to the latest version.

Instead of messing up with the core files, it is more efficient to override the CSS file to code the desired styles. This is how a Web developer can also prevent customizations from blending with the original code. This mechanism of customizing Bootstrap not only prevents deviation from the normal workflow but also facilitates hassle-free upgradation.

Figure 6.16 shows how to customize Bootstrap by overriding CSS.

```
<!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">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity=
"sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg8Va+Pmd9iz/b5Ft+bdN/1iqg" crossorigin="anonymous">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity=
"sha384-rHyoN1iZaVX9GabO3snlcGzrC2oZ7ueJ6WfKB7W1xD9Kkz+0W0Ny7q1Rd7gwpJgj6qG2A" crossorigin="anonymous">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity=
"sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUfHRa4Hq+MVR9iqdX2+2ZmeW2vR0QqVVhxS8SzK+Me5KA" crossorigin="anonymous"></script>
<style type="text/css">
    body {
    padding: 15px;
}

table {
    border-collapse: separate;
    border-spacing: 0 5px;
}

body {
    background-color: lightblue;
}

p {
    font-family: calibri;
    font-weight: bold;
}

thead th {
    background-color: #006DCC;
    color: yellow;
}

tbody td {
    background-color: #EDEDED;
}

tr td:first-child,
tr th:first-child {
    border-top-left-radius: 6px;
    border-bottom-left-radius: 6px;
}

tr td:last-child,
tr th:last-child {
    border-top-right-radius: 6px;
    border-bottom-right-radius: 6px;
}
    }
</style>
</head>
<body>
<div class="bs-example">
    <div class="table-responsive">
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Course Name</th>
                    <th>Developed By</th>
                    <th>Platform</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>1</td>
                    <td>Java</td>
                    <td><p>James Gosling</p></td>
                    <td><p>Java was originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of
                    Sun Microsystems' Java platform.</p></td>
                </tr>
                <tr>
                    <td>2</td>
                    <td>Ruby</td>
                    <td><p>Yukihiro Matsu</p></td>
                    <td><p>Ruby is a dynamic, reflective, object-oriented, general-purpose programming language. It was designed and
                    developed in the mid-1990s by Yukihiro "Matz" Matsumoto in Japan.</p></td>
                </tr>
                <tr>
                    <td>3</td>
                    <td>Python</td>
                    <td><p>Guido van Rossum</p></td>
                    <td><p>Python was conceived in the late 1980s, and its implementation began in December 1989 by Guido van Rossum at
                    Centrum Wiskunde & Informatica (CWI)</p></td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
</body>
</html>
```

Figure 6.16: Code for Customizing Bootstrap

In figure 6.16, the code overrides the default Bootstrap CSS by defining custom styles for `body`, `p`, and `table` elements. The `:first-child` selector selects the mentioned selector, only if it is the first child of its parent. Similarly, the `:last-child` selector matches each element that is the last child of its parent. By retrieving the first and last child elements, we are assigning specific CSS attributes to different elements.

Figure 6.17 shows the output.

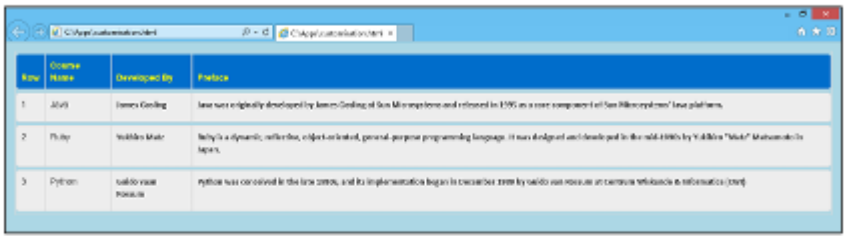


Figure 6.17: Output of Code for Customizing Bootstrap

6.4 Bootstrap Wells

In Bootstrap, a well is a container component to apply an inset style to a block-level element or sunken effect to its content. It is useful when a Web developer intends to insert content inside a box such that it stands apart from the other contents on that Webpage.

To define a well, a Web developer needs to wrap the content with a `<div>` containing the .well class. This class applies a rounded border with some padding and a gray background color.

It is also possible to change the well's size by controlling its padding. This is done with the help of two modifier classes namely, `.well-lg` and `.well-sm`. These optional classes work by modifying the default padding and size of the base class, `.well`. They render a well smaller or larger depending on the class a Web developer specifies with the base class. Code Snippet 3 shows the use of the modifier and base well classes.

Code Snippet 3:

```
<div class="container">
  <h2>Wells</h2>
  <div class="well well-sm">SMALL
WELL with reduced padding and
   radius</div>
  <div class="well">DEFAULT WELL with default padding and
   radius</div>
  <div class="well well-lg">A LARGE WELL with increased padding and
   radius</div>
</div>
```

Figure 6.18 shows a smaller, default, and larger well.

Figure 6.18: Wells of Different Sizes

Figure 6.19 shows how to place form controls in a well.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Bootstrap Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>
<div class="well well-sm">
    <h1>Login Form</h1>
    <hr />
    <form class="form-horizontal">
        <div class="form-group">
            <label for="inputUserName" class="control-label col-sm-2">Username</label>
            <div class="col-sm-10">
                <input class="form-control" placeholder="Login Username"
                    type="text" id="inputUserName" />
            </div>
        </div>
        <div class="form-group">
            <label for="inputPassword" class="control-label col-sm-2">Password</label>
            <div class="col-sm-10">
                <input class="form-control" placeholder="Login Password"
                    type="password" id="inputPassword" />
            </div>
        </div>
        <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
                <button type="submit" class="btn btn-default">Login</button>
            </div>
        </div>
    </form>
</div>
</body>
</html>
```

Figure 6.19: Code for Placing Form Controls in a Well

In figure 6.19, the code includes a horizontal form and its controls with a `div` that contains the `.well-sm` class. This means that the form shall appear in a well of smaller size than the default one.

Figure 6.20 shows the output.

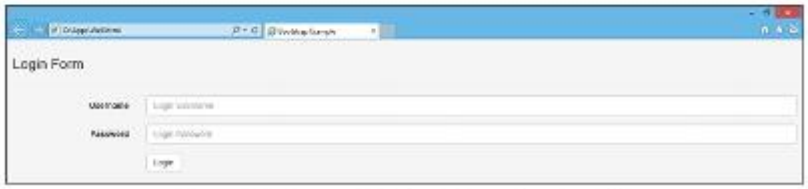


Figure 6.20: Output of Code for Placing a Form in a Well

In figure 6.20, it is clear that a smaller well features lesser round corners. The text also appears sunken inside a well.

6.4.1 Tabs in a Bootstrap Well

Let's now see one more example of using wells. A Web developer can put a tabbed interface within a well to make it stand out from the rest of the content on a Webpage. A tabbed interface contains a series of tabs, drop-down menus, or a combination of both.

To create a tabbed interface, Bootstrap offers the `.nav-tabs` class, which should be added to the `.nav` base class in `<ul>`. Figure 6.21 shows how to create a tabbed interface using data attributes and place it within a well.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>
    <br />
    <div class="well well-sm class">

        <div class="row">
            <div class="col-xs-6">

                <ul class="nav nav-tabs">
                    <li class="active">
                        <a href="#homeTab" data-toggle="tab">Home</a>
                    </li>
                    <li>
                        <a href="#contactTab" data-toggle="tab">Contact</a>
                    </li>
                    <li class="dropdown">
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle">
                            Locations <b class="caret"></b>
                        </a>
                        <ul class="dropdown-menu">
                            <li><a href="#indiaTab" data-toggle="tab">India</a></li>
                            <li><a href="#usaTab" data-toggle="tab">USA</a></li>
                        </ul>
                    </li>
                </ul>
                <div class="tab-content">
                    <div id="homeTab" class="tab-pane active">
                        <h3>Home</h3>
                        <p>Home page content</p>
                    </div>
                    <div id="contactTab" class="tab-pane">
                        <h3>Contact</h3>
                        <p>Contact page content</p>
                    </div>
                    <div id="indiaTab" class="tab-pane">
                        <h3>India</h3>
                        <p>India Location Details</p>
                    </div>
                    <div id="usaTab" class="tab-pane">
                        <h3>USA</h3>
                        <p>USA Location Details</p>
                    </div>
                </div>

            </div>
        </div>
    </div>
</body>
</html>
```

Figure 6.21: Code for Placing a Tabbed Interface in a Well

In figure 6.21, `.well-sm` class is used to define a small well in a grid layout defined for extra small devices. The `.nav-tabs` class is used to define the tabs in a navbar. There are two tabs namely, `Home` and `Contact` and one drop-down menu named `Locations` with `India` and `USA` in its drop-down list. For tabs, the `data-toggle` attribute is set to `tab`. Similarly, for a drop-down menu, the attribute is set to `dropdown`. The `.caret` class produces a caret arrow icon, which denotes that the tab is a drop-down menu. The `.tab-content` and `.tab-pane` classes are used to define the content that will be displayed when a user clicks the respective tabs.

Figure 6.22 shows the output.

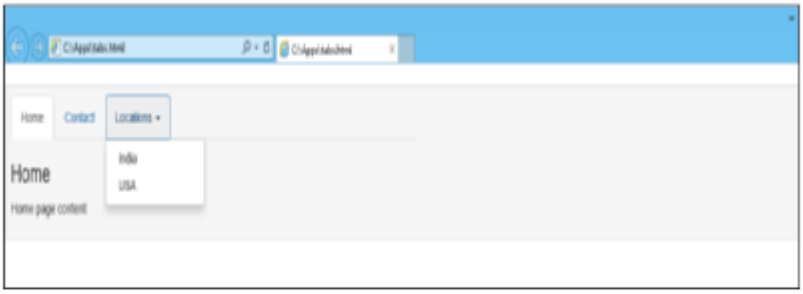


Figure 6.22: Output of a Tabbed Interface in a Well

6.5 Building a Website Using Bootstrap and jQuery

jQuery is used in Bootstrap for designing more robust and dynamic Webpages by extending their HTML and JavaScript functionalities. It makes it simpler to use and quicker to design Webpages that are compatible across a multitude of browsers.

Figure 6.23 shows how to design a Webpage using Bootstrap components and jQuery library.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
   <script type="text/javascript">
      $(document).ready(function () {
         $('#cbShowPassword').click(function () {
            $('#txtPassword').attr('type', $(this.is(':checked') ? 'text' : 'password');
         });
          $('input[name="language"]').click(function () {
            getSelectedCheckBoxes('language');
          });
          var getSelectedCheckBoxes = function (groupName) {
             var result = $('input[name="' + groupName + '"]:checked');
             if (result.length > 0) {
                var resultString = result.length + " languages checked\n";
                result.each(function () {
                   resultString += $(this).val() + "<br>";
                });
                $('#div' + groupName).html(resultString);
             }
             else {
                $('#div' + groupName).html("No checkbox checked");
             }
          });
       });
    </script>
</head>
<body>
<div class="container">
  <h2>Login Page</h2>
  <form>
    <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" class="form-control" id="email" placeholder="Enter email">
    </div>
    <div class="form-group">
      <label for="pwd">Password:</label>
      <input type="password" class="form-control" id="txtPassword" placeholder="Enter password">
    </div>
    <div class="checkbox">
      <label><input type="checkbox" id="cbShowPassword"> Show Password</label>
    </div>
    <div class="form-group">
    <button type="submit" class="btn btn-default">Submit</button>
   </div>
   <div class="form-group">
   Known Languages:
    <div class="checkbox">
    <label><input type="checkbox" name="language" value="English" />English</label>
    <br /><br />
   <label> <input type="checkbox" name="language" value="German" />German</label>
   </div>
   Selected Languages:<br />
    <label> <div id="divlanguage"></div> </label>
    <br />
   </div>
  </form>
</div>
</body>
</html>
```

Figure 6.23: Code for Using Bootstrap and jQuery

In figure 6.23, the code includes Bootstrap components such as container, form, and form controls such as buttons and input controls. It integrates jQuery functionality into Bootstrap by including a script. In this example, we are using jQuery to display the characters of a typed password whenever the user selects the `Show Password` check box. The jQuery library is also used to display the number of languages selected along with their names. The click event is raised and is handled through jQuery. The script uses the `attr()` method associated with id selector to fetch the typed password characters only when the `cbShowPassword` check box is selected. The ternary operator checks whether the check box is selected or not. For fetching the selected languages, the code defines an anonymous function `getSelectedcheckBoxes`. The function iterates through each selected check box to count the total number of selected check boxes and fetch the corresponding language names.

Figure 6.24 shows the output before clicking the `Submit` button.

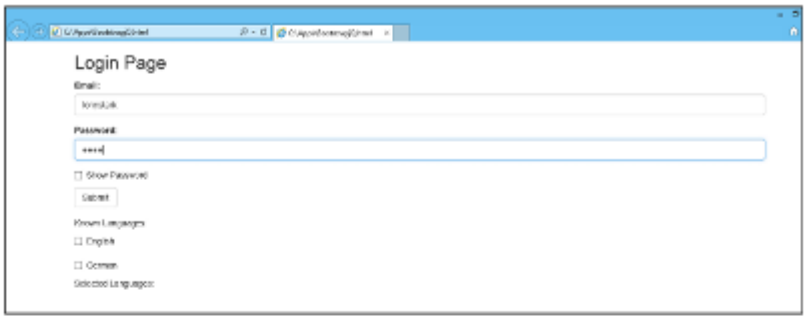


Figure 6.24: Output before Submitting the Form

Figure 6.25 shows the output after clicking the `Submit` button.

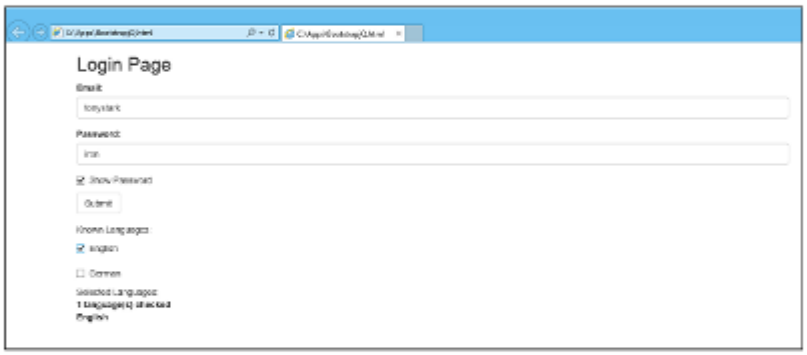


Figure 6.25: Output after Submitting the Form

Quick Test 6.1

1. Bootstrap 3 allows including device-specific styles in a single Webpage.

   a. True
   b. False

2. There is a custom builder for building one's own Bootstrap version.

   a. True
   b. False

Summary

- A media query defines styles for displaying a Webpage on a specific type of device without modifying the markups.
- Media queries are useful in determining the device's physical properties namely, viewport's and device's width and height, screen resolution, and orientation.
- Media queries define the breakpoint for the media's width from where the layout changes on a specific screen.
- Web developers can define device-specific style sheets in Bootstrap and include all in a single Webpage.
- A basic Bootstrap table features distinct appearance in terms of only horizontal splitters and slight padding of 8 pixels.
- A Bootstrap table can have striped rows, borders on all sides, a hovering effect on rows, and emphasized effect on rows or cells.
- The emphasized effect on rows and cells comes from one or more contextual classes namely `.active`, `.success`, `.info`, `.danger`, and `.warning`.
- Bootstrap 3 activates horizontal scrolling on a small device whose screen width is not more than 768 pixels.
- Instead of messing up with the core files, it is more efficient to override the default CSS file in Bootstrap.
- A well is a container component to apply an inset style to a block-level element or sunken effect to its content.
- It is also possible to change the well's size by controlling its padding through the modifier classes.

Check Your Progress

1. **Which of the following properties does a media query helps in determining in Bootstrap 3?**
   a. Viewport's width
   b. Screen's resolution
   c. Device's height
   d. All of these
2. **Which of the following classes is not added to the base `.table` class?**
   a. `.table-bordered`
   b. `.table-hover`
   c. `.table-responsive`
   d. None of these
3. **The _____ class is used to make a compact Bootstrap table.**
   a. `.table-compact`
   b. `.table-condensed`
   c. `.table-sm`
   d. `.table-ls`
4. **Which of the following statements is true about customizing Bootstrap?**
   a. The SASS version allows compiling custom CSS
   b. It allows changing the core CSS
   c. It allows overriding the core CSS
   d. All of these
5. **Which of the following effect is well suited to the content of a Webpage?**
   a. Round borders
   b. Gray background
   c. Borders on all sides
   d. Collapsed text

Answers to Check Your Progress

1. All of these.
2. `.table-responsive`
3. `.table-condensed`
4. All of these
5. Round borders

Answers to Quick Test

Quick Test 6.1

1. True
2. True