JQUERY

jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto: Write less, do more. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery:

 DOM manipulation: The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called Sizzle.

 Event handling: The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.

 AJAX Support: The jQuery helps you a lot to develop a responsive and feature-rich site using AJAX technology.

 Animations: The jQuery comes with plenty of built-in animation effects which you can use in your websites.

 Lightweight: The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).

 Cross Browser Support: The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+

 Latest Technology: The jQuery supports CSS3 selectors and basic XPath syntax.

HowtousejQuery?

There are two ways to use jQuery.

 Local Installation − You can download jQuery library on your local machine and include it in your HTML code.

 CDN Based Version − You can include jQuery library into your HTML code directly from Content Delivery Network (CDN).

1. OVERVIEW

LocalInstallation

- Go to the https://jquery.com/download/ to download the latest version available.
- Now, insert downloaded jquery-2.1.3.min.js file in a directory of your website, e.g. /jquery.

Example

Now, you can include jquery library in your HTML file as follows:

```
<html>
 <head>
 <title>The jQuery Example</title>
 <script type="text/javascript" src="/jquery/jquery-2.1.3.min.js"></script>
 <script type="text/javascript">
 $(document).ready(function(){
 document.write("Hello, World!");
 });
 </script>
 </head>
 <body>
 <h1>Hello</h1>
 </body>
</html>
```

This will produce the following result –

```
Hello, World!
```

CDNBasedVersion

You can include jQuery library into your HTML code directly from Content Delivery Network (CDN). Google and Microsoft provides content deliver for the latest version.

We are using Google CDN version of the library throughout this tutorial.

Example

Now let us rewrite above example using jQuery library from Google CDN.

```
<html>
 <head>
 <title>The jQuery Example</title>
 <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
 </script>
 <script type="text/javascript">
 $(document).ready(function(){
 document.write("Hello, World!");
 });
 </script>
 </head>
 <body>
 <h1>Hello</h1>
 </body>
</html>
```

This will produce the following result:

```
Hello, World!
```

HowtoCallajQueryLibraryFunctions?

As almost everything, we do when using jQuery reads or manipulates the document object model (DOM), we need to make sure that we start adding events etc. as soon as the DOM is ready.

If you want an event to work on your page, you should call it inside the $(document).ready() function. Everything inside it will load as soon as the DOM is loaded and before the page contents are loaded.

To do this, we register a ready event for the document as follows:

```
$(document).ready(function() {
 // do stuff when DOM is ready
});
```

To call upon any jQuery library function, use HTML script tags as shown below:

```
<html>
<head>
<title>The jQuery Example</title>
 <script type="text/javascript"
 src="/jquery/jquery-1.3.2.min.js"></script>

 <script type="text/javascript" language="javascript">
 // <![CDATA[
 $(document).ready(function() {
 $("div").click(function() {
 alert("Hello world!");
 });
 });
 // ]]>
 </script>
</head>
<body>
<div id="newdiv">
Click on this to see a dialogue box.
jQuery
13
</div>
</body>
```

```
</html>
```

This will produce the following result:

Click on this to see a dialogue box.

HowtoUseCustomScripts?

It is better to write our custom code in the custom JavaScript file : custom.js, as follows:

```
/* Filename: custom.js */
$(document).ready(function() {
 $("div").click(function() {
alert("Hello world!");
 });
});
```

Now we can include custom.js file in our HTML file as follows:

```
<html>
<head>
<title>The jQuery Example</title>
 <script type="text/javascript"
 src="/jquery/jquery-1.3.2.min.js"></script>
 <script type="text/javascript"
 src="/jquery/custom.js"></script>
</head>
<body>
<div id="newdiv">
Click on this to see a dialogue box.
</div>
</body>
</html>
```

This will produce the following result:

Click on this to see a dialogue box.

UsingMultipleLibraries

You can use multiple libraries all together without conflicting each others. For example, you can use jQuery and MooTool javascript libraries together. You can check jQuery noConflict Method for more detail.

jQuery noConflict() Method

Many JavaScript libraries use $ as a function or variable name, just as jQuery does. In jQuery's case, $ is just an alias for jQuery, so all the functionality is available without using $.

Run $.noConflict() method to give control of the $ variable back to whichever library first implemented it. This helps us to make sure that jQuery doesn't conflict with the $ object of other libraries.

Here is a simple way of avoiding any conflict:

```
// Import other Library
// Import jQuery Library
$.noConflict();
// Code that uses other library's $ can follow here.
```

This technique is especially effective in conjunction with the .ready() method's ability to alias the jQuery object, as within the .ready() we can use $ if we wish without fear of conflicts later:

```
// Import other library
// Import jQuery
$.noConflict();
jQuery(document).ready(function($) {
 // Code that uses jQuery's $ can follow here.
});
// Code that uses other library's $ can follow here.
```

WhatisNext?

Do not worry too much if you did not understand the above examples. You are going to grasp them very soon in subsequent chapters. In the next chapter, we would try to cover few basic concepts which are coming from conventional JavaScript.