Certificate in C# Programming

Creating Web Applications in C#

By Randal Root

# Module 05: Client Script Libraries

To create dynamic, data-driven, web applications require several web development tools. Of course, you need an editor that allows you to develop both client-side and server-side code. That server-side code will usually be C sharp or VB.NET if you're developing it an ASP.NET application. The client-side code will be a combination of HTML, CSS, and JavaScript. Over the years, many development tools have been created to help you manipulate these three languages. These tools become more or less popular as time goes on, but currently jQuery one of the most popular choices in use. jQuery is a set of pre-made JavaScript functions that you can either download or reference in your web pages.

In this Module, we see how developers use JavaScript via client script libraries. The examples we will use are jQuery and Bootstrap, both of which can improve your ASP.NET applications. Since this Module is outside of the course Textbook, we will use <http://www.w3schools.com> as a web resource instead.

## Referencing JavaScript

Almost all, client script libraries depend on a browsers support of JavaScript. You add JavaScript to webpages in three different ways; Inline, Internal, or External. Each has an advantage, but most developers place code in an "external" script file within a folder called **Scripts**.

#### Scripts\ExternalJavaScriptFile.js

function ExternalFunction(Message) {

if (5 == 5) {

alert(Message);

}

}

#### 01WorkingWithJavaScript.html

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>01WorkingWithJavaScript</title>

<script type="text/javascript">

function InternalFunction(Message) {

if (5 == 5) {

alert(Message);

}

}

</script>

<script src="Scripts/ExternalJavaScriptFile.js"></script>

</head>

<body>

<input id="Button1" type="button" value="Inline" onclick="if (5 == 5) { alert('Inline button') }" />

<input id="Button2" type="button" value="Internal" onclick="InternalFunction('Internal button')" />

<input id="Button3" type="button" value="External" onclick="ExternalFunction('External button')" />

</body>

</html>

# JQuery

JQuery is a collection of JavaScript types, properties, and methods added to web pages.

“jQuery is a fast and concise JavaScript library created by John Resig in 2006. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for Rapid Web Development.” (<http://www.tutorialspoint.com/jquery>)

You can find out more information about JQuery on the open-source community website: <http://jquery.com>

### Adding JQuery to your ASP.NET project

To add JQuery to your projects, you need to ***either*** **download** a *local* copy of the JQuery source code **or** **reference** a *remote* Content Delivery Network (CDN) like the ones hosted by Google or Microsoft. Downloading jQuery scripts is the **oldest** method of adding JQuery to a project. You **create a scripts folder** in the project and then **download** a set of **pre-made JavaScript files** from the JQuery website (Figure 1).

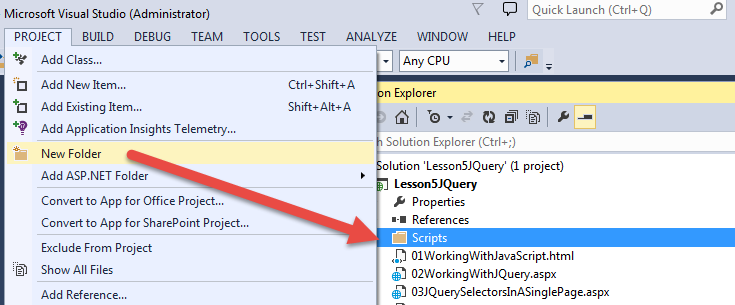


Figure 1: Creating a Scripts folder.

A **more modern** way to download in Visual Studio is to **use NuGet** through Tools > NuGet Package Manager > Manage NuGet Packages For Solution (Figure 2).

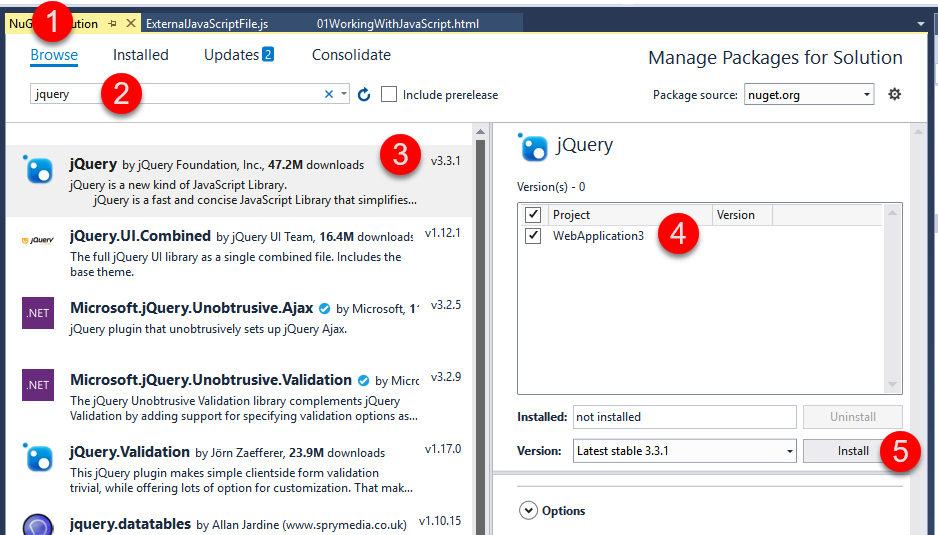


Figure 2: Installing jQuery using NuGet

Using either option, allows you to add a reference to the JQuery file in your web page, by referencing it (Figure 3).

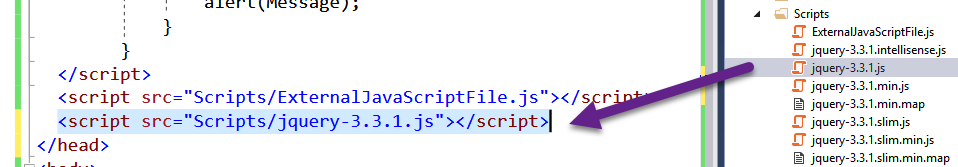


Figure 3: Referencing the jQuery Script file

#### Demo 01 Module5Site.Master

<%@ Master Language="C#"%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

<%-- This script block adds the JQuery reference to all pages the use this Master --%>

**<script src="Scripts/jquery-3.1.1.js" type="text/javascript"></script>**

<asp:ContentPlaceHolder ID="**head**" runat="server">

</asp:ContentPlaceHolder>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:ContentPlaceHolder ID="**ContentPlaceHolder1**" runat="server">

</asp:ContentPlaceHolder>

</div>

</form>

</body>

</html>

### JQuery Basic Syntax

Now that we have a reference to the script file, we create content that uses the types, properties, and methods within it.

In JQuery, you use a ***Selector*** to reference an object in the Brower’s DOM and then interact with it using actions. The standard syntax is: **$(selector).action()**

* + The **$** symbol is used to define the **start** of a JQuery **selector**
  + The **(selector)** is the name of the **HTML element** you want to reference
  + The **action()** is the **method or property** you want to apply to the selected object

Here is an example:

$(document).ready(//Add code here!);

### The $(document).ready() function

One of the first function you must know about is the “$(**document**).**ready**()” function. This function selects the document object and responds when the Browser's DOM fires the ready event handler. That event fires once a page completely loads in the Web Browser.

Tip: You place whatever code you wish to run at that event and can even prepare other event processing by creating event delegates and handlers, similar to C#.

#### Demo 02WorkingWithJQuery.aspx

<%@ Page Title="02WorkingWithJQuery" Language="C#" MasterPageFile="~/Module5Site.Master" %>

<%--My jQuery reference is in the Master page!--%>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">

<script type="text/javascript">

$(document).ready(function () { alert("This is a test"); });

</script>

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<ol>

<li>Add JQuery to your project site. (NuGet or http://jquery.com/download/)</li>

<li>Add a Reference to the file in your page using the Script tag with a src attribute</li>

<li>Add your JavaScript code to one or more script blocks.</li>

</ol>

\* In older versions of Visual Studio you had to also download a document file for IntelliSense

</asp:Content>

<asp:Content ID="Content1" ContentPlaceHolderID="**head**" runat="server">

<%-- This script block is specific for this single page, but uses the JQuery script reference --%>

**<script type="text/javascript">**

**$(document).ready(function () { alert("This is a test"); });**

**</script>**

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="**ContentPlaceHolder1**" runat="server">

<ol>

<li>Download JQuery</li>

<li>Add a Reference to the file in your page using the Script tag with a src attribute</li>

<li>Add your JavaScript code to one or more script blocks.</li>

</ol>

</asp:Content>

LAB 01: Using JQuery

In this lab, you will:

1) Create a Visual Studio project called **MY**Module05Labs using the Visual Studio -> ASP.NET Empty Web Application template (Figure 4).

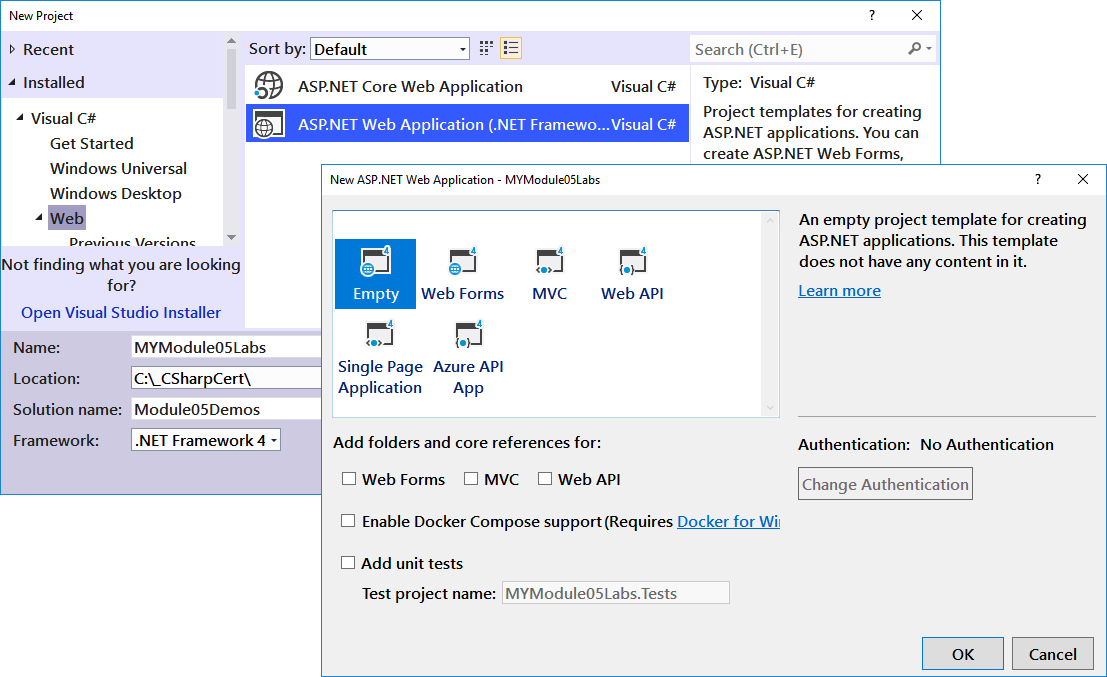


Figure 4: Creating a project for the labs.

2) Add the JQuery source code to the project using NuGet (Figure 5).

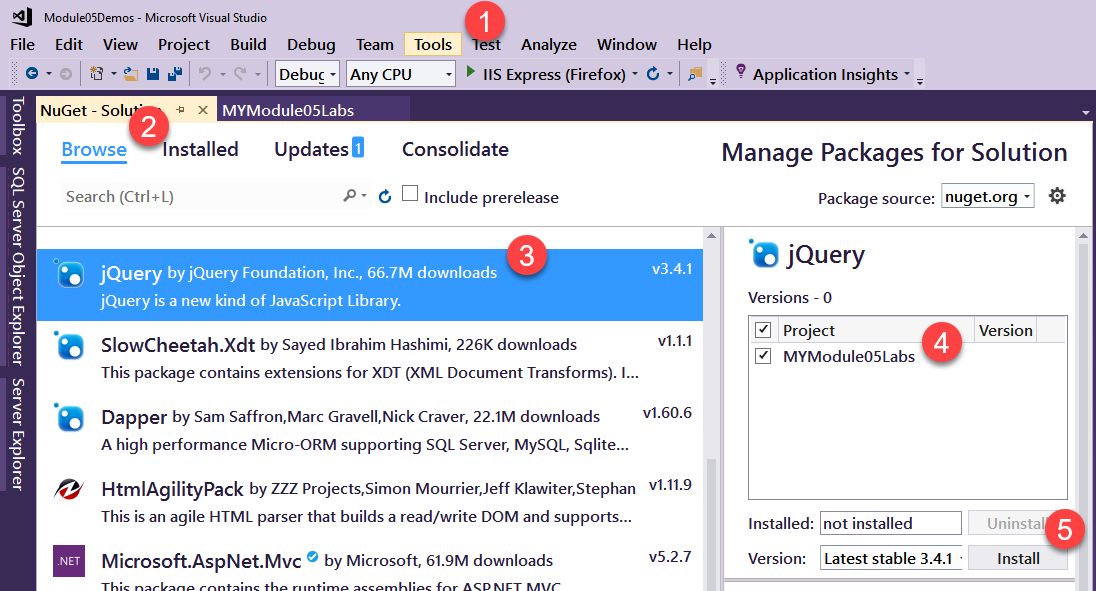


Figure 5. Adding jQuery using NuGet.

3) Create a new Master Page called LabSite.Master. Add a code to reference the JQuery script in the **Master** page so that all content pages can use it (Figure 6.)

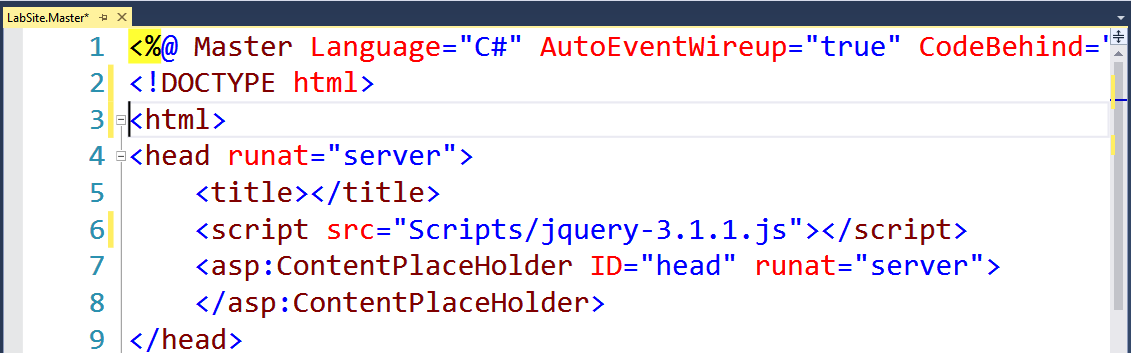


Figure 6: Adding a script reference

**Notes**:

* *The External Script Link must be above the Content Place Holder!*
* *You need to reference the correct version that you downloaded!*
* Older versions of Visual Studio required you to click on a JQuery file to activate the following dialog option. This option installed additional JQuery snippets to your project before IntelliSense would work.

5) Create a content page that uses this code:

<script type="text/javascript">

$(document).ready(**DisplayMsg**());

function **DisplayMsg**() {

alert("JQuery is working in Lab01");

}

</script>

6) Test that the JavaScript code displays a message in an alert box (Figure 7).

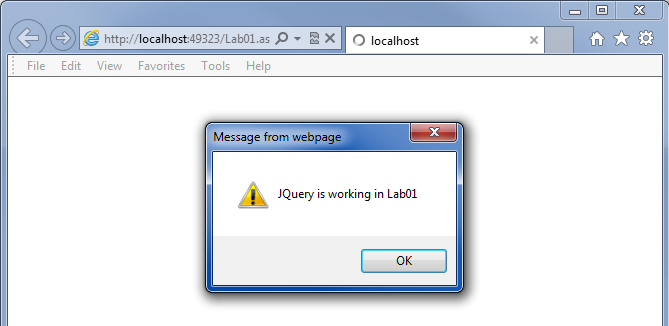


Figure 7: The result of the jQuery code

**This lab should take about 5 minutes**

## Basic JQuery Syntax

When using JavaScript and JQuery in a web page, you may find the syntax to be a challenge at first. Here are some tips to help you get started:

1. JavaScript function definitions require **parentheses**

<script type="text/javascript">

function MyClickEventMethod**()** { alert("You clicked a label 1"); }

2. JavaScript functions are often **anonymous**

$(**document**).**ready**(function () {

// Place your code in this anonymous function

});

3. You use **selectors** to manipulate properties, methods, and event handlers

// Using a **[#]** to select by an **ID** (ID Label1)

$("#Label1").html("This is using an ID");

// Using a **[.]** to select by a **Class** (In this case Class Lable2)

$(".ClassDemo").html("This is using a Class");

// Using a **["HTML Element"]** to select by all Elements

// of that type (In this case All Spans in the DOM)

$("span").addClass("SilverStyle");

4. There are **different** ways to call functions **with or without** **parameters**

//MyClickEventMethod**()** has no parameters it is defined with parenthese.

function **MyClickEventMethod**() { alert("You clicked a label 1"); }

//but, is called without parentheses.

$("#Label1").click(**MyClickEventMethod**);

//**MyClickEventMethodWithSimpleParams(p1)** has a parameter

// and its argument is treated as objects with a data property!

function **MyClickEventMethodWithSimpleParams**(MyMessage) { alert(MyMessage.**data**); }

//Arguments are passed to the function like this:

$("#Label2").click("A Simple Message",**MyClickEventMethodWithSimpleParams**);

5. You can also use **anonymous functions to create event handlers** without parameters

//Here we create an ***anonymous function*** to handle a click event

$("#Label4").click(function () {

$('#Span1').text("New Text for span1");

// .text() is a JQuery function used to access the innerHTML text of a element

// Since .text() is a JQuery function, as compared to your own Custom function, you

//change the text of a DOM element like this [Function]([Argument])!

$('#Label4').text("New Text for Label3");

})

6. You use anonymous types to send arguments as Key and Values pairs

// This is another JavaScript function with a parameter, note the way the data is used.

function **MyClickEventMethodWithParams**(MyEventData) { alert(**MyEventData.data.SomeMessage**); }

// Here we are *passing a named argument* the MyClickEventMethodWithParams

// function with parentheses.

$("#Label3").click({**SomeMessage:** "Randal's Data from the Argument" }, **MyClickEventMethodWithParams**);

/\* Note: The Key:Value pair are contained in a set of braces creating **an anonymous type** \*/

7. Use the text() and val() function to access data depending on the HTML element's type

// .text() accesses the innerHTML text of a element

alert($('#Label4').text());

// while .val() accesses **input objects** (TextBoxes, DropDowns,etc.)

alert($('#TextBox1').val());

Let's look at an example page that includes these features.

#### Demo03JQuerySelectorsInASinglePage.aspx

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Demo 03</title>

<style type="text/css">

.SilverStyle {

background-color: Silver;

}

</style>

<%-- This script block adds the JQuery reference, enabling access to all the JQuery functions and fields --%>

<script src="Scripts/jquery-3.1.1.js" type="text/javascript"></script>

<%-- Note: You MUST use another, separate, script block for your personal code!! --%>

<script type="text/javascript">

// This is a simple JavaScript function without parameters, but note that the PARENTHESES are STILL REQUIRED.

function MyClickEventMethod() { alert("You clicked a label 1"); }

// This is another JavaScript function with a parameter note that the value is an Object.

function MyClickEventMethodWithSimpleParams(MyMessage) { alert(MyMessage.data); }

// This is another JavaScript function with a parameter.

function MyClickEventMethodWithParams(MyEventData) {

alert(MyEventData.data.SomeMessage);

alert(MyEventData.data.AnotherMessage);

}

</script>

<%-- Note: You can use separate script blocks to organize your code as well --%>

<script type="text/javascript">

$(document).ready(function () { // This Anonymous function will fire once the document has completed

// loading in the Browser and is "READY."

// You can map event handling using the [Selector].[Event] syntax as follows...

// http://api.jquery.com/click/

$("#Label1").click(MyClickEventMethod); // NOTE: We DON'T use Parentheses after the function name

//since MyClickEventMethod() has are no arguments.

//Here we are passing an argument the MyClickEventMethodWithSimpleParams function and do use Parentheses.

$("#Label2").click("A Simple Message", MyClickEventMethodWithSimpleParams);

//Here we are passing an argument the MyClickEventMethodWithParams function and do NOT use Parentheses.

$("#Label3").click({ SomeMessage: "Randal's Data from Argument", AnotherMessage: "More Data" }, MyClickEventMethodWithParams);

// ---------- Note: The Name : Value pair are contained in a set of braces creating an anonymous type (with name and value pairs)

// These are typed before the method's name, but will be used by the function MyClickEventMethodWithParams automatically----

//Here we create an anonymous function to handle a click event

//\*\*\*(And yes, it is an anonymous JS Function inside of another anonymous JS Function!)\*\*\*

$("#Label4").click(function () {

$('#Span1').text("New Text for span1");// .text() is a JQuery function used to access the innerHTML text of a element

// Since .text() is a JQuery function, as compared to your own Personal function, you change the text of a

//DOM element like this[Function]([Argument])!

$('#Label4').text("New Text for Label3");

})

$("#HTMLButton").click(function () {

// You use the following [Selector].[Property] syntax to access a browser's Document Object Model (DOM) objects with JQuery.

// Here are three common ways to $elect an element in a Browser's DOM. (http://www.w3schools.com/Jquery/jquery\_selectors.asp)

$("#Label1").html("This is using an ID Selector"); // Using a [#] to select by an ID (ID Label1)

$(".ClassDemo").html("This is using a Class Selector"); // Using a [.] to select by a Class (In this case Class Lable2)

$("span").addClass("SilverStyle"); // Using a ["HTML Element"] to select by all Elements of that type

(In this case All Spans in the DOM)

})

$("#Button1").click(function () {

alert($('#Label4').text());// .text() accesses the innerHTML text of a element

alert($('#TextBox1').val());// while .val() accesses input objects (TextBoxes, DropDowns,etc.)

//NOTE: The PostBack will reset it to the original text!

// The HTML span changes and loses it's state data on a PostBack to the server.

$('#Span1').text("New Text for span1. At least for 3 seconds!");

// An ASP:Label and looses its state data with or without the ViewStateMode property being enabled!

$('#Label4').text("New Text for span1. At least for 3 seconds!");

//IMPORTANT:

// If you want the data to persist you need to store it somewhere the client can re-access it

// or not use code in that overwrites PostBack data, (such as the $(document).ready(function ()...)

})

});

</script>

<script runat="server">

protected void Page\_Load(object sender, EventArgs e)

{

//Slowing the response so we can see the change happen easier.

System.Threading.Thread.Sleep(2000);

}

</script>

</head>

<body>

<form id="form1" runat="server">

<div>

<h3>Click on the labels and buttons to demonstrate how the JavaScript events and functions work!</h3>

<asp:Label ID="Label1" runat="server" Text="Label 1 (click me)"></asp:Label>

<br />

<asp:Label ID="Label2" Class="ClassDemo" runat="server" Text="Label 2 (click me)"></asp:Label>

<br />

<asp:Label ID="Label3" Class="ClassDemo" runat="server" Text="Label 3 (click me)"></asp:Label>

<br />

<span id="Span1">This is Original text in an HTML span</span>

<br />

<asp:Label ID="Label4" runat="server" Text="This is Original text in an ASP:Label" ViewStateMode="Enabled">

</asp:Label>&nbsp;&nbsp;

<br />

<asp:TextBox ID="TextBox1" runat="server" Text="This is Text in Textbox1"></asp:TextBox>

&nbsp;

<asp:Button ID="Button1" runat="server" Text="ASP Button" />

<br />

<input type="button" id="HTMLButton" value="This is a HTML Button" onclick=""/>

</div>

</form>

</body>

</html>

## Referencing ASP.NET Dynamically Named Objects

Because of the way ASP.NET controls dynamically assign a client-side name, your code may need to be adjusted to use the ClientID property.

#### Demo 04JQuerySelectorsInAContentPage.aspx

<%@ Page Title="03" Language="C#" MasterPageFile="~/Lesson5Site.Master" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">

<script src="Scripts/jquery-1.11.1.js" type="text/javascript"></script>

<script type="text/javascript">

$(document).ready(function () {

// This Anonymous function will fire once the document has

//completed loading in the Browser and is "READY."

$("#Button1").click(function () {

//This is using the standard HTML name does not work because the name is dynamic.

$("#Label1").html("Standard HTML name does NOT work!");

//In a Content page access objects using

//the **Content Place Holder's Name + the Control's ID syntax**

$("#ContentPlaceHolder1\_Label2").html("Standard JavaScript syntax DOES work!");

//Usung the **the ClientID property works too!**

//it's using Response.write shorthand code %= to get the labels name

$("#<%=Label3.ClientID%>").html("Using the ASP.NET ClientID property DOES works!");

});

});

</script>

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<div>

<h1>Click on the labels and the button to demonstrate how the events functions run...</h1>

<asp:Label ID="Label1" runat="server" Text="Original Text for Label1"></asp:Label>

<br />

<asp:Label ID="Label2" runat="server" Text="Original Text for Label2"></asp:Label>

<br />

<asp:Label ID="Label3" runat="server" Text="Original Text for Label3"></asp:Label>

<br />

<input id="Button1" type="button" value="button" />

</div>

</asp:Content>

LAB 02: Using JQuery in a Content Page

In this lab, you will:

1) Create a new Content Page called Lab02 by copying and pasting the Lab01 page into your existing project and Verify that the page still works as it did before.

2) Add a new TextBox and Button to the content page.

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<%--3) Add code to create Label, Textbox, and Button --%>

<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>

<br />

<asp:TextBox ID="TextBox1" runat="server" Text="Text in Textbox1"></asp:TextBox>

<br />

<asp:Button ID="Button1" runat="server" Text="Button" ViewStateMode="Disabled" />

<br />

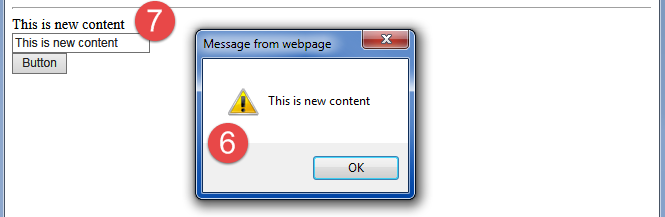
</asp:Content>

3) Add code to handle the Button.Click event and test it using an alert box.

4) Add code to display the contents of TextBox1 in an Alert, then test the page.

5) Add code to display the contents of TextBox1 in the Label and the alert box. Then, test the page.

6) Test that the page works as expected.



#### Figure 9: Results of steps 6 and 7.

**This lab should take about 10 minutes**

## Using CDN (Content Delivery Network)

If you don't want to host the jQuery script file in your project, you can reference it from a CDN (Content Delivery Network) like Google or Microsoft. Many examples you find on the Internet will use this option.

#### Demo: 05UsingRemoteJQueryScripts.aspx

<%@ Page Title="05UsingRemoteJQueryScripts" Language="C#" MasterPageFile="~/Module5Site.Master" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">

<%-- Here is an **example from the W3Schools** http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery\_lib\_google --%>

<%-- , but this time we will do the something similar with Microsoft's CDN --%>

**<script type="text/javascript" src="//ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.1.min.js"></script>**

<script type="text/javascript">

$(document).ready(function () {

//Inset a new click event function into the page after it is loaded.

$("p").click(function () { $(this).hide(); });

// [this] refers to the DOM object that fired the event

});

</script>

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<div>

<p>

If you click on me, I will disappear.

</p>

<p>

Click me away!

</p>

<p>

Click me too!

</p>

</div>

</asp:Content>

## Using jQuery to access Server-Side code

One common use of jQuery is to access Web Service methods. We will look at web services at the end of the course, but for now, here is a simple example:

#### Demo: 06SamePageWebService.aspx

<%@ Page Language="C#" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

<script runat="server">

//This code creates a Web Service with a single Web Method.

//This is odd, but you can use one ASP.NET page to both view content and

//background processing, but normally you use a Web Service .asmx page.

[System.Web.Services.WebMethod]

public static string GetServerOutput()

{

return DateTime.Now.ToLongTimeString();

}

</script>

<script src="Scripts/jquery-3.1.1.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$("#Button1").click(function () {

$.ajax({

type: "POST",

dataType: "json",

contentType: "application/json",

url: "05SamePageWebService.aspx/GetServerOutput",

success: function (data) {

$("#spanResults").html(data.d);

},

error: function () {

alert("The call to the web service failed.");

}

})

});

});

</script>

</head>

<body>

<form id="form1" runat="server">

<div>

<input id="Button1" type="button" value="Button1" />

<br />

<br />

<span id="spanResults"></span>

</div>

</form>

</body>

</html>

## Other Tools

JQuery is just one of several web development tools that allow you to use pre-made JavaScript functions. Common tools include Bootstrap, KnockOut , AngularJS, and many others. Let's look at Bootstrap and an example of what these other libraries look like.

# Using Bootstrap for Web Client Development

Bootstrap is a collection of pre-made CSS styles and JavaScript functions pre-tested to work on almost any device and nearly every browser.

“Advantages of Bootstrap:

* Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
* Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
* Mobile-first approach: In Bootstrap 3, mobile-first styles are part of the core framework
* Browser compatibility: Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Safari, and Opera).” ([http://www.w3schools.com/Bootstrap/Bootstrap\_get\_started.asp](http://www.w3schools.com/bootstrap/bootstrap_get_started.asp))

## Getting Started with Bootstrap

One place to start learning about Bootstrap is from its official website, [http://getBootstrap.com](http://getbootstrap.com). To access Bootstrap's CSS styles and JavaScript functions, you can download the Zip files or use its CDN (Figure 11). Currently, Bootstrap uses jQuery functions to do a lot of its work, so you must download or reference that as well.

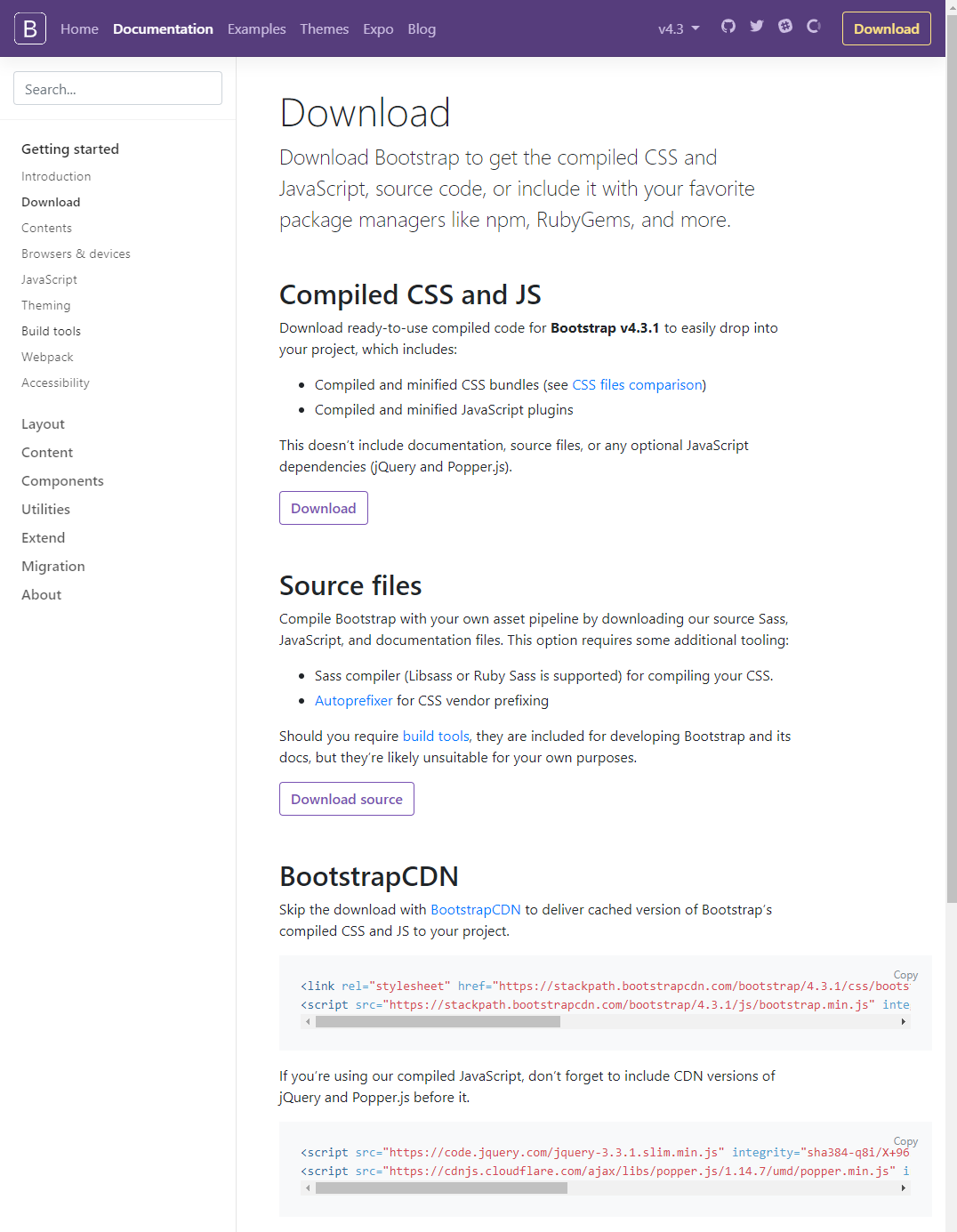


Figure 11: Bootstrap's Download and Reference Page.

**Notes:**

* Bootstrap 5, will move away from using JQuery.
* You can also install Bootstrap through NuGet.

## Creating a Bootstrap page

Once you download or reference Bootstrap, you create a simple page to test it. The w3Schools website offers several starting templates (<https://www.w3schools.com/bootstrap/bootstrap_templates.asp>), but here is a very simple example.

#### 07SimpleBootStrapTemplate.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">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>

</head>

<body>

<div class="container">

<h1>My First Bootstrap Page</h1>

<p>This is some text.</p>

</div>

</body>

</html>

## Containers

“Bootstrap also requires a containing element to wrap site contents.

There are two container classes to choose from:

The .container class provides a responsive **fixed width container**

The .container-fluid class provides a **full-width container**, spanning the entire width of the viewport

**Note:** Containers are not nestable (you cannot put a container inside another container).” ([http://www.w3schools.com/Bootstrap/Bootstrap\_get\_started.asp](http://www.w3schools.com/bootstrap/bootstrap_get_started.asp))

Here is the w3School starting template.

### *Viewport* Meta Tag

In CSS 2.1, the Viewport Meta tag attribute was added to help with formatting on mobile devices. You can find out more on [the official w3c website](http://dev.w3.org/csswg/css-device-adapt), but the simple description on the w3Schools website is a good start. Mozilla published more detailed examples and information about this topic on their development site (figure 12). (<https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag>).

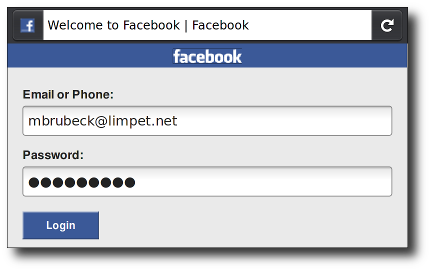
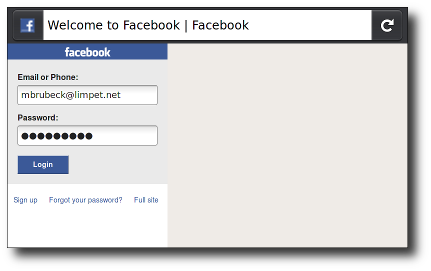


Figure 12: touch.facebook.com before and touch.facebook.com after the Viewport tag

## The Bootstrap Grid

Bootstrap divides every page into a grid of **12 columns**. Columns are automatically merged if you do not indicate otherwise, which means, that if you do not define enough div space, adding up to 12, Bootstrap will fill the rest in for you.

<div class="row" style="border-style:dashed">

<div class="col-sm-2" style="border-style:solid">.col-sm-**2**</div>

<div class="col-sm-4" style="border-style:solid">.col-sm-4</div>

<!--( Since 2 and 4 are only 6, the next division is implied! ) -->

<!--<div class="col-sm-6" style="border-style:solid">.col-sm-6</div>-->

</div>

Here is some good info about the grid system: [http://www.w3schools.com/Bootstrap/Bootstrap\_grid\_system.asp](http://www.w3schools.com/bootstrap/bootstrap_grid_system.asp)

## Bootstrap Styles

Bootstrap has styles it automatically applies to standard HTML 5 tags. For example, it will make h2, p, and many others, change location based to the size of a browser.

<h2>Bootstrap applies default styles</h2>

<p>Bootstrap applies default styles to common HTML 5 </p>

#### Demo8ColumnsAndStyles.html

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Columns and Styles</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>

<!--CSS -->

<!--Remember that..-->.

<style>

#AddIndividualStyle {

background-color: gold;

}

/\* Styles can be individualized\*/

.text-info {

color: darkorange;

}

/\* Styles can overridden\*/

.text-danger {

background-color: yellow;

}

/\* Styles can added to \*/

</style>

</head>

<body>

<p>

Bootstrap displays columns vertically on small screens and horizontal on larger ones.

Columns are set in a <mark>grid of 12 spaces.</mark>

</p>

<div class="row" style="border-style:dashed">

<div class="col-sm-2" style="border-style:solid">.col-sm-2</div>

<div class="col-sm-4" style="border-style:solid" id="AddIndividualStyle">.col-sm-4</div>

</div>

<br />

<p>You create a Bootstrap container by adding the class="container" attribute</p>

<div class="container" style="border-style:dashed">

<div class="row" style="border-style:solid">

<div class="col-sm-4" style="border-style:solid">.col-sm-4</div>

<div class="col-sm-6" style="border-style:solid">.col-sm-6</div>

</div>

</div>

<br />

<h2>Bootstrap applies default styles</h2>

<p>

Bootstrap applies default styles to common HTML 5 tags

that are a bit different then the Browser defaults

</p>

<h2>Bootstrap has Contextual Colors</h2>

<p>There are a set of pre-made color schemes</p>

<p class="text-muted">Test</p>

<p class="text-primary">Test</p>

<p class="text-success">Test</p>

<p class="text-info">Test</p>

<p class="text-warning">Test</p>

<p class="text-danger">Test</p>

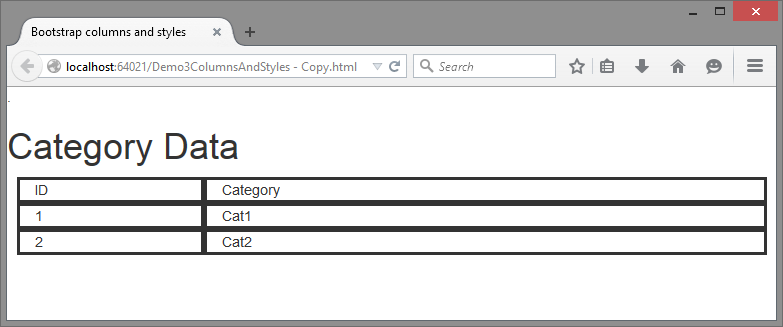
</body>

</html>

LAB 03: Creating a Bootstrap web client

In this lab, you will:

1. Create a web page that uses a Bootstrap Grid to display the data for Category IDs, Category Names in two columns similar to this:



1. Test it in a Browser.

**Hint**: check out this webpage before you start ([http://www.w3schools.com/Bootstrap/Bootstrap\_grid\_system.asp](http://www.w3schools.com/bootstrap/bootstrap_grid_system.asp))

**This lab should take about 5 to 10 minutes**

Bootstrap Table Styling

Rumors of the demise of HTML tables have "been greatly exaggerated," and Bootstrap includes several table styles.

#### Demo9TableStyles.html

<!DOCTYPE html>

<html lang="en">

<head>

<title>Table Styles</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>

<!--CSS -->

<style>

.table.table-RjRsDEMO {background-color:gold;} /\* Styles can sub-classed\*/

</style>

</head>

<body>

<h1>A Bootstrap table with 'sub-classes'</h1>

<!--These are premade Bootstrap Classes, except the last one, of course -->

<table class="table table-bordered table-hover table-condensed table-RjRsDEMO">

<thead>

<tr>

<th>ProductID</th>

<th>ProductName</th>

<th>ProductPrice</th>

</tr>

</thead>

<tbody>

<!--Bootstrap row can use the same style classes for formatting row backgrounds-->

<tr class="success">

<td>1</td>

<td>ProdA</td>

<td>1.99</td>

</tr>

<tr class="info">

<td>2</td>

<td>ProdB</td>

<td>2.99</td>

</tr>

<tr class="danger">

<td>3</td>

<td>ProdC</td>

<td>3.99</td>

</tr>

</tbody>

</table>

</body>

</html>

LAB 04: Creating Bootstrap Tables

In this lab, you will:

1. Convert the Bootstrap Grid you made in Lab 01 into a table
2. Test it in a Browser.

**This lab should take about 5 to 10 minutes**

## Bootstrap, JQuery, and ASP.NET

Now that we have the client-code is working, we add server-code to our pages. In this next example, we pull data from a SQL database as content for an HTML table.

#### 10ServerAndClientCode.aspx

<%@ Page Title="10" Language="C#" MasterPageFile="~/Lesson5Site.Master" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">

<meta name="viewport" content="width=device-width, initial-scale=1"/>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"/>

<style>

.table.table-RjRsDEMO {

background-color: gold;

}

/\* Styles can sub-classed\*/

</style>

<script runat="server">

protected void getDataButton\_Click(object sender, EventArgs e)

{

System.Data.OleDb.OleDbConnection objOleCon = new System.Data.OleDb.OleDbConnection();

System.Data.OleDb.OleDbCommand objCmd = new System.Data.OleDb.OleDbCommand();

try

{ //1. Make a Connection

string strOledbConnection = @"Provider=SQLOLEDB;

Data Source=.\SQLExpress;

Integrated Security=SSPI;

Initial Catalog=ASPNetHomework";

objOleCon.ConnectionString = strOledbConnection;

objOleCon.Open();

//2. Issue a Command

objCmd.Connection = objOleCon;

objCmd.CommandText =

@"Select LoginID, Name, EmailAddress, LoginName, ReasonForAccess From Logins";

System.Data.OleDb.OleDbDataReader objDR = objCmd.ExecuteReader();

//3. Process the Results

string strTable =

@"<table class='table table-bordered table-hover table-condensed table-RjRsDEMO'>

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Email</th>

<th>Login</th>

<th>Reason</th>

</tr>

</thead>

<tbody>";

while (objDR.Read() == true)

{

strTable += "<tr>";

strTable += "<td>" + objDR["LoginID"] + "</td>";

strTable += "<td>" + objDR["Name"] + "</td>";

strTable += "<td>" + objDR["EmailAddress"] + "</td>";

strTable += "<td>" + objDR["LoginName"] + "</td>";

strTable += "<td>" + objDR["ReasonForAccess"] + "</td>";

strTable += "</tr>";

}

strTable += @"</tbody>

</table>";

lblTable.Text = strTable;

}

catch (Exception ex) { lblMessage.Text += "<b>" + ex.ToString() + "</b>"; }

finally { objOleCon.Close(); } //4. Run clean up code

}

</script>

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<div>

<asp:Button ID="getDataButton" runat="server" Text="Get Data" OnClick="getDataButton\_Click" />

&nbsp;<asp:Literal ID="lblTable" runat="server" Text=""></asp:Literal>

&nbsp;<asp:Label ID="lblMessage" runat="server" Text="" Style="font-size: large"></asp:Label>

</div>

</asp:Content>

# Moving On

Bootstrap 5 is a re-write that replaces jQuery with their own JavaScript functions. W3.CSS is a CSS Framework that also does not use jQuery or other JavaScript libraries. However, like so much of Web technology, once you learn about one library, you can figure out another.

For now, you can see that using any of the ASP.NET technologies along with jQuery and Bootstrap, is easy. Perhaps this is why it has become a popular option in web development!