==

JQuery

What You Will Learn in This Lecture:

- What jQuery is?
- How to use jQuery to enhance your pages, including adding rich?
- Visual effects and animations.

JavaScript is the de facto language for client-side scripting and interacting with elements in your web pages at the client. JavaScript can do much more and is quite a powerful programming language. But powerful as it may be, it has a few shortcomings. One of the problems with JavaScript is that not all browsers interpret it the same way. A lot of the JavaScript code you'll write will work in all major browsers, but subtle differences in code and behavior exist that make it difficult to write code that behaves exactly the same in all major browsers. Also, JavaScript lacks some useful features that would come in handy in your day-today JavaScript coding. For example, it has built-in methods to find a specific element on a page (using getElementById) and to find all elements of a specific HTML tag (using getElementsByTagName), but it lacks features like getElementsByClassName to get a list of elements with a specific class applied to them. The client-side ASP.NET AJAX Library you were introduced to in the previous lecture helps to overcome some of these problems, but it isn't enough in all situations.

Fortunately, the Internet developer community has been very active developing frameworks that use JavaScript under the hood and that extend its power, while offering a very rich feature set that helps you create interactive client-side web pages. Over the years, many JavaScript libraries have been developed — most of which are free — including:

- Prototype (http://prototypejs.org)
- Scriptaculous, an add-on to Prototype (http://script.aculo.us)
- Ext JS (http://extjs.com)
- Dojo (http://dojotoolkit.org)

One of the frameworks that have gotten a lot of attention is *jQuery*. Initially developed and released by John Resig in January 2006, jQuery has grown to be a very popular client-side framework. It also caught the attention of Microsoft, which decided to start shipping jQuery with Microsoft products. Initially, jQuery shipped with the Microsoft ASP.NET MVC Framework, but it's now also included in Visual Studio and Visual Web Developer 2010.

An Introduction to jQuery

The main focus of the jQuery library has always been to simplify the way you access the elements in your web pages, provide help in working with client-side events, enable visual effects like animations, and make it easier to use Ajax in your applications. In January 2006, John Resig announced the first version of jQuery, which was followed by an official release of jQuery 1.0 in August 2006. Many more versions would follow, with version 1.4.1 as the latest, stable release. You can download the latest version of jQuery from the official web site at http://jquery.com. Not only will you find the downloadable files there, but you'll also find the documentation, FAQs, tutorials, and much more information you can use to make the most out of jQuery. Besides downloading the library from the jQuery web site, any new web site you create using the ASP.NET Web Site template already contains a Scripts folder with the necessary jQuery files. You see later how to add jQuery files to your site manually. Because the jQuery library adds to the size of your web pages, it should be a deliberate choice whether or not you include it in your site. When adding the jQuery library to your site, you have a few choices to make.

Choosing the Location for Your jQuery Reference

To include jQuery in your web site, you have a couple options:

1. Add a reference to the jQuery library in just the web pages or user controls that require it.
2. Add a reference to the jQuery library in the master page of your site so it's available in all pages.

Both methods have their own advantages and disadvantages. Adding a reference to the jQuery library in just the pages that need it helps keep the size of your pages down a bit. When your users browse only to pages without jQuery, they'll never have to download the library file. Once they've downloaded the file, the browser will cache a copy of it, removing the need to download it again on subsequent visits to pages.

Adding the reference to jQuery in the master page of your site is quite convenient, because all pages based on this master page automatically get access to the jQuery functionality. However, this results in a small performance hit on the first page of your site because the library needs to be downloaded from the server. Because the jQuery library is quite small, you typically want to include the library in the master page. Besides the location where you add your jQuery file, you also have a few options with regard to the way you include the file.

Different Ways to include the jQuery Library

Because the jQuery library consists of a single file with JavaScript code, you can embed a reference to the library in a page, user control, or master page using the standard <script> syntax:

```
<script src="FileName.ext" type="text/javascript"></script>
```

It's important to use a separate closing </script> tag because some browsers will choke if you use a self-closing tag. I prefer to store all my client-side script files in a Scripts folder in the root of my site, so a reference to the jQuery library (called jquery-1.4.1.min.js) will end up like this:

```
<script src="/Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

You can also embed the reference inside the ScriptManager control that you have added to the master. The ScriptManager control has a <Scripts> child element that lets you register JavaScript files that will be added to the final page in the browser. In its simplest form, a JavaScript file registered in the ScriptManager looks like this:

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
<Scripts>
<asp:ScriptReference Path="~/Scripts/jquery-1.4.1.min.js"/>
</Scripts>
</asp:ScriptManager>
```

Another alternative is to refer to an online version of the library with Microsoft's *Content Delivery Network* (CDN) or Google Code. For more information on this, visit Microsoft's CDN site at www.asp.net/ajax/cdn or Google's API site at http://code.google.com/apis/ajaxlibs/. The advantages of using online versions of external libraries are improved performance and lowered bandwidth for your servers. Since it is likely that visitors to your site already have downloaded the shared scripts when visiting another site, they don't have to download them again when visiting yours. In the following exercise, you add the jQuery library version 1.4.1 to the master page.

Your First jQuery Page

In this exercise, you add the jQuery library to the master page so it's available to all pages in your site.

1. Start by adding a new Scripts folder to the root of the site in Visual Web Developer.

2. Next, open the folder where you extracted the downloaded code. If you followed the instructions in the Introduction of this lecture, the file jquery-1.4.1.min.js is the actual jQuery library and jquery-1.4.1-vsdoc.js is a documentation file for IntelliSense and is used only in the context of VWD. Finally, jquery.updnWatermark.js contains a plugin for jQuery and is discussed toward the end of this lecture. Your Solution Explorer should now look like Figure 1.

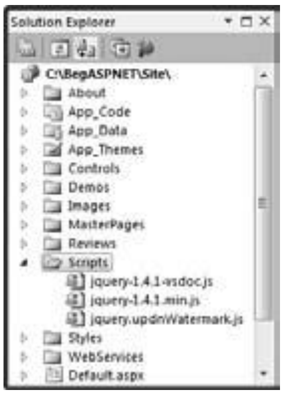


Figure 1

3. Next, it's time to add the library to the site's master page so the pages in your web site have access to the jQuery library. To do this, open the file Frontend.master from the MasterPages folder and switch it to Markup View if necessary. Locate the ScriptManager control and add the following bolded markup to it:

```
<asp:ScriptManager ID="ScriptManager1" runat="server"
EnablePageMethods="true">
<Scripts>
<asp:ScriptReference Path="~/Scripts/jquery-1.4.1.min.js" />
</Scripts>
</asp:ScriptManager>
```

If your ScriptManager didn't have a separate closing tag yet you should add one now (and remove the slash (/) from the opening tag) or the code won't be added correctly.

4. Save and close the master page because you're done with it for now.

5. To try out the jQuery library, create a brand new Web Form in the Demos folder based on your custom template. Call the page jQuery.aspx, and set its Title to **jQuery Demo**.

6. With the new page open in Markup View, add the following code to the Content block for cpMainContent:

```
<asp:Content ID=”Content2” ContentPlaceHolderID=”cpMainContent”
runat=”Server”>
<input id="Button1" type=" button" value="button" />
<script type="text/javascript">
$(document).ready(function() {
$('#MainContent').css('background-color', 'green')
$('#Button1').click(function() {
$('#MainContent').css('background-color', 'red')
.animate({ width: '100px', height: '800px' })
});
});
</script>
</asp:Content>
```

Just like many other programming languages, JavaScript (and thus jQuery) is quite sensitive to missing quotes, brackets, and parentheses, so make sure you type this code exactly as shown here. Note that while typing; IntelliSense pops up, helping you complete the code, and giving you information about various methods and parameters in a tooltip. If it doesn’t pop up, make sure you added the right <Scripts> element to the master page. Also, try saving and closing all open documents and then reopen jQuery.aspx.

7. Save the changes to the page and then press Ctrl+F5 to open it up in the browser. Notice how the background color of the MainContent div has turned to green. Click the button and notice how the background color changes to red and how the MainContent element changes size, ending up with a width of 100 pixels and a height of 800 pixels.

How It Works

Although the effects shown in this exercise aren't that fancy, a lot is going on under the hood to make this example work. To understand how it works, first look back at the master page where you added a reference to the jQuery library:

```
<asp:ScriptManager ID="ScriptManager1" runat="server"
EnablePageMethods="true">
<Scripts>
<asp:ScriptReference Path="~/Scripts/jquery-1.4.1.min.js" />
</Scripts>
</asp:ScriptManager>
```

This tells the script manager to include a script element pointing to the jQuery library. If you look in the HTML source for the page in the browser you should see the following script element:

```
<script src="../Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

This in turn tells the browser to download the jquery-1.4.1.min.js file from the Scripts folder, giving your page access to all functionality included in the jQuery library. The next thing to look at is the code in the jQuery demo page. First, you added a standard <script> block that can contain JavaScript. Inside this block, you added some jQuery code that is fired as soon as the browser is done with loading the page. Everything between the opening ({) and closing (}) curly braces are executed when the page is ready:

```
<script type="text/JavaScript">
$(document).ready (function () {
// Remainder of the code skipped
});
</script>
```

Because the jQuery code interacts with the elements on the page, you often have to wait until the entire page has loaded so the elements you're programming against are available. Adding jQuery code like this is a standard practice to delay execution of the code until the entire page is ready. The code that is executed when the page is ready consists of two parts. The first line of code sets the background color of the MainContent div to green:

```
$('#MainContent').css ('background-color', 'green')
```

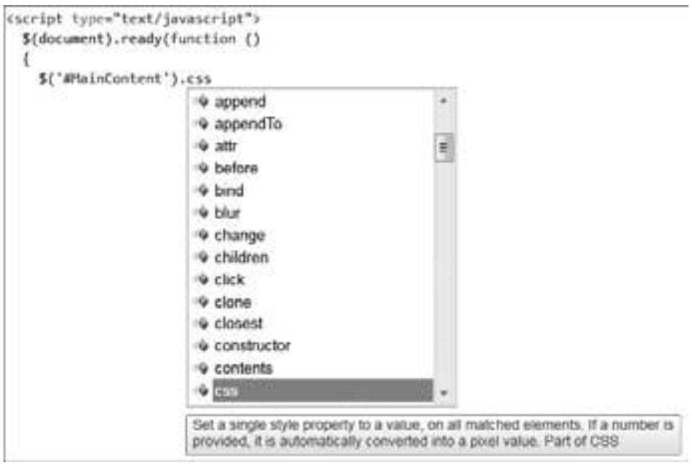
This code gets a reference to the MainContent div element and then calls a css method to change the background color to green. Remember $get from the previous lecture that gets a reference to an element in the page by its id in the client-side ASP.NET AJAX Library? In this example, $('#MainContent') is jQuery's equivalent, but as you see later, it's much more powerful. The second part sets up a click handler for the HTML button you added to the page. Inside the click handler you see some code that changes the background color of the MainContent div to red, and changes the height and the width of it using a fluid

Animation:

```
$('#Button1').click (function () {
$('#MainContent').css ('background-color', 'red').animate ({width: '100px', height:'800px' })
});
```

Again, you learn more about how jQuery is able to find the button and the div element and how the css and animate methods work later in this lecture, so don't worry too much if none of this is making a lot of sense right now. When you click the button in the browser, the MainContent's background color is changed to red, and then its width and height are changed to 100 and 800 pixels, respectively. When you typed the jQuery code you may have noticed you got help from IntelliSense. As soon as you typed $(you got a tooltip explaining the information you can pass to this function. Likewise, IntelliSense helps you find and complete the css method and the various arguments you need to pass to it as shown in Figure 2 (which shows the tooltip under the IntelliSense list to better accommodate the width of this book). IntelliSense for jQuery works through the extra file — jquery-1.4.1-vsdoc.js — you added to the site. VWD scans the solution for files ending in vsdoc.js, parses them, and then uses the documentation it finds in them to build up the IntelliSense list.

==



```
<script type="text/javascript">
  $(document).ready(function ()
  {
    $('#MainContent').css
```

Set a single style property to a value, on all matched elements. If a number is provided, it is automatically converted into a pixel value. Part of CSS

Figure 2

The reason for the separate documentation file is to keep the size of the original jQuery library down. Without the documentation, the library is only 70KB; with the documentation the file size increases to 230KB. You should use the vsdoc file only for development in VWD and never include a link to it in your pages because it doesn't add any value in the browser compared to the real library file. Because of the large size of the documentation file, you're wasting valuable bandwidth and time if you use that file instead of the real library file.

JQuery Core

Most of the jQuery code you write will be executed when the browser is done loading the page. It's important to wait with executing your code until the page is done loading the *DOM*. The DOM — the document object model — is a hierarchical representation of your web page and contains a tree-like structure of all your HTML elements, script files, CSS, images, and so on. The DOM is always in sync with the page you see in the browser, so if you make a programmatic change to the DOM (for example, with jQuery code), the change is reflected in the page in the browser. If you execute your jQuery code too early (for example, at the very top of the page), the DOM may not have loaded the elements you're referring to in your script, and you may get errors. Fortunately, it's easy to postpone the execution of your code until the DOM is ready using the ready function in jQuery. You've already seen the ready function at work in the previous Try It Out, but it's shown here again now that you better understand what it's used for:

```
$(document).ready (function () {
// Code added here is executed when the DOM is ready.
});
```

Any code you add between the opening and closing curly brace is executed when the page is ready for DOM manipulation. JQuery also comes with a shortcut for the ready function to make it easier to write code that fires when the DOM is ready. The following snippet is equivalent to the preceding example:

```
$(function () {
// Code added here is executed when the DOM is ready.
});
```

Because jQuery code is often specific to a page, it makes sense to add the code to the end of just the pages that require it. To make this a little easier, you can add a ContentPlaceHolder in your master page especially for this purpose. The pages that use this master page then have an easy location to write jQuery code. You see how to do this in the next exercise. In the previous jQuery example you saw some code that selected the MainContent div and the button in your page. However, jQuery comes with a lot more options to select specific elements in your pages. These options are discussed next.

Selecting Items Using jQuery

In jQuery you use the dollar sign ($) as a shortcut to find elements in your page. The elements that are found and returned are referred to as a *matched set*. The basic syntax for the $ method is this: `$('Selector Here')` Between the quotes (you can use single or double quotes, as long as you use the same type on each end) you enter one or more selectors, which are discussed later. The $ method returns zero or more elements that you can then influence using one of the many jQuery methods. For example, to apply some CSS to all h2 elements, you use the css method:

```
$('h2').css ('padding-bottom', '10px');
```

This applies a padding of ten pixels at the bottom of all headings at level two in the page. The cool thing about many of the jQuery methods is that, besides applying some design or behavior, they return the matched set again. This enables you to call another method on the same matched set. This concept is called *chaining*, where you use the

===

result of one method as the input of another, enabling you to create a chain of effects. For example, the following code first changes the font size of all level-two headings in the page, and then fades them out until they are invisible in five seconds:

```
$('h2').css('font-size', '40px').fadeOut(5000); // timeout is in milliseconds.
```

Basic Selectors

JQuery *selectors* enable you to find one or more elements in your page's document object model so you can apply all sorts of jQuery methods to these elements. The great thing about jQuery selectors is that you already know how they work. Rather than inventing a new technique to find page elements, the designers of jQuery decided to use an existing selector-based syntax that you are already familiar with: CSS. Remember the CSS selectors from lecture 5? You can use the exact same ones
in jQuery.

The Universal Selector

Just as its CSS counterpart, the universal selector matches all elements in your page. To set the fontfamily of each element in your page to Arial, you use this code:

```
$('*').css('font-family', 'Arial');
```

The ID Selector

This selector finds and retrieves an element by its id, the same as you would do in CSS. For example, to set the CSS class for a button called Button1, you use this code:

```
$('#Button1').addClass('NewClassName');
```

When this code sets the CSS class (using the addClass method), the standard CSS rules apply. That means that for this code to work and change the appearance of the button, the NewClassName class needs to be available to the page, either through an external CSS file or by an embedded style sheet.

The Element Selector

This selector gets a reference to zero or more elements that match a specific tag name. For example, this code turns the text color of all headings at level two to blue:

```
$('h2').css('color', 'blue');
```

The Class Selector

The class selector gets a reference to zero or more elements that match a specific class name. Consider this HTML fragment:

==

```
<h1 class="Highlight">Heading 1</h1>
<h2>Heading 2</h2>
<p class="Highlight">First paragraph</p>
<p>Second paragraph</p>
```

Notice how two of the four elements have a CSS class called Highlight. The following jQuery code changes the background color of the first heading and the first paragraph to red, leaving the other elements unmodified:

```
$('.Highlight').css('background-color', 'red');
```

Grouped and Combined Selectors

Just as with CSS, you can group and combine selectors. The following grouped selector changes the text color of all h1 and h2 elements in your page:

```
$('h1, h2').css('color', 'orange');
```

With a combined selector, you can find specific elements that fall within some others. For example, the following jQuery touches just the paragraphs that fall within the MainContent element, leaving all other paragraphs alone:

```
$('#MainContent p').css('border', '1px solid red');
```

To get a feel of the selectors in jQuery and the affects you can apply to the matched set, the next exercise shows you how to use some of the selectors and apply some animations to the matched sets.