WA1935 Mastering jQuery

Student Labs

Web Age Solutions Inc.

Table of Contents

Lab 1 - Getting Started with jQuery

In this lab, we will setup the basic development environment. We will also learn how to install jQuery and learn its basic usage.

Part 1 - A Note About Browsers

As a best practice, you should always test your application using multiple browsers and versions. For these labs, we recommend that you use the latest versions of these browsers:

1. IE
2. FireFox
3. Chrome

Some of the labs will not work with very old browsers, like IE 6.

Part 2 - Looking at Apache

You will use Apache 2.2.x as the web server for this class. This is already installed for you. Please verify the installation folder. It should be **C:\Program Files\Apache Software Foundation\Apache2.2**

From now on, we will refer to this folder as APACHE_DIR.

We will now validate that Apache is working properly.

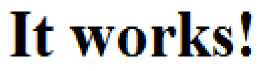
__1. Using Notepad, open **APACHE_DIR/htdocs/index.html**.

__2. Note that the file currently contains one line:

```
<html><body><h1>It works!</h1></body></html>
```

__3. From a web browser, enter the URL: **http://localhost**

__4. Make sure that you see:


It works!

Part 3 - Using a Text Editor

You can use any text editor for this class. You can use Notepad or Wordpad. You can also download and install your favorite text editor.

Part 4 - Install jQuery

For our labs, we will need jQuery and jQuery UI JavaScript libraries. Those have already been downloaded for you. All you need to do is copy them over to the right folder.

__1. In the **APACHE_DIR/htdocs** folder, create a new folder called **jquery**

In Windows XP and Vista, you may have to be an Administrator to create folders within APACHE_DIR.

__2. From folder **C:\LabFiles\** copy **jquery-1.10.2.min.js** and **jquery-ui.js** to the **APACHE_DIR/htdocs/jquery** folder.

Part 5 - Write a Basic jQuery Application

__1. In the **APACHE_DIR/htdocs** folder, create a new folder called **lab**

__2. In the lab folder, create a new file called **basic.html**

__3. Open the file with an editor and write the basic structure.

```
<html>

<head>
</head>

<body>
</body>

</html>
```

__4. Import jQuery JavaScript file as shown in bold face.

```
<head>
<script type="text/javascript"
    src="/jquery/jquery-1.10.2.min.js"></script>
</head>
```

__5. Add an empty paragraph within the body.

```
<body>
    <p id="p1"></p>
</body>
```

__6. Below the existing <script> tag, add another one.

```
<script type="text/javascript">
    $(function(){
        $("#p1").text("Hello World");
    });
</script>
```

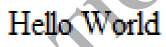
We will inspect the code later. For now, continue with testing.

__7. Save changes.

Part 6 - Test

__1. From the browser, enter the URL: **http://localhost/lab/basic.html**

__2. Make sure that you see:


Hello World

Part 7 - What Just Happened?

All the magic happens in the script:

```
<script type="text/javascript">
    $(function(){
        $("#p1").text("Hello World");
    });
</script>
```

First, let's talk about the core function:

```
$()
```

Almost all jQuery API is accessible through this one single function. You can also call the core function through these names:

```
jQuery()
window.jQuery()
```

But, $() is short and sweet and that's what most developers use.

The core function behaves completely differently depending on the input parameters you pass to it. In our case, we passed an anonymous function:

```
function(){
    $("#p1").text("Hello World");
}
```

When you pass a function as input argument, jQuery treats it as a DOM ready event handler. This event is fired when the DOM document of the page is fully ready. This can happen before the "onload" event of the body. For example, if there are many images in the document, the DOM ready event fires before all the images are actually downloaded.

You can start manipulating the DOM as soon as the DOM is ready.

When the DOM document is ready, FireFox fires the “DOMContentLoaded” event and IE the “onreadystatechange” event for the document object. jQuery makes it easy to attach an handler for that event in a browser independent manner.

Now, let's look at what our DOM ready event handler function does. Firstly, we see that the core function is used again.

```
$("#p1")
```

This time, the core function is taking a string as an argument. This signifies to the core function that the argument is an element selector. jQuery borrows heavily from CSS selector model. According to CSS, "#p1" select a DOM element with the ID of "p1". That will be the empty paragraph we had added earlier.

$("#p1") will return a jQuery object for the paragraph. We then call the text() function for the object. This sets the inner text of the element to "Hello World".

What does the core function - $("#p1") – actually return? It returns a jQuery object that is a collection of all DOM element objects that match the supplied selector rule. This jQuery object builds on top of the JavaScript array API. For example, you can obtain the length:

$("p").length; //Number of paragraphs in the page.

To obtain a DOM element object in the collection, simply call get(index).

alert($("p").get(1).innerHTML); //Show inner HTML of the second paragraph.

When you invoke a method of the collection object, jQuery internally applies it to all DOM elements in the collection:

$("p").text("Hello!"); //Change text of all paragraphs to Hello!.

A getter function normally works on the first DOM element in the collection only.

$("p").text(); //Returns the text of the first paragraph only.

To convert a DOM element object into a jQuery collection do:

$(elementObject)

For example:

$(document.getElementById("p1")); //Same as $("#p1").

Part 8 - Something More Advanced

So far, $("#p1") returned a single object. This expected since an element always has a unique ID. Now, we will use selector to return a collection of jQuery objects and work on them.

__1. Below the current paragraph add another one as shown in bold face below.

```
<p id="p1"></p>
<p id="p2"></p>
```

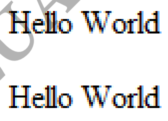
__2. Change the DOM ready event handler as shown below.

```
$("p").text("Hello World");
```

Now, $("p") will select all DOM elements with a <p> tag. Once again, jQuery borrows this from the CSS selector model.

__3. Save changes.

__4. Refresh the browser. Now, both paragraphs will say "Hello World".



Hello World

Hello World

What this means is that the text() function was called for the entire collection of jQuery objects returned by $("p"). Trying to do this using raw JavaScript will take quite a few lines of tedious coding.

Part 9 - Review

In this lab, we installed jQuery and wrote a very basic application. We learned about the core function $(). We wrote a DOM ready handler. We used a simple ID based selector first. And then a tag based selector.

Lab 2 - More on Selectors

In this lab, we will learn a few more advanced uses of selectors. We will also learn about how to manipulate the CSS styles of elements.

Part 1 - Iterating Over a Selected Collection

We know that the core function returns a collection of DOM element objects. We can iterate over the collection using the each() function.

We currently have two paragraphs in basic.html.

```
<p id="p1"></p>
<p id="p2"></p>
```

Let us say that we want to iterate over them and change their text.

__1. Change the line:

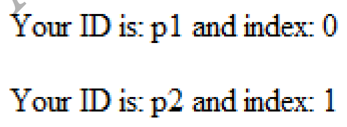
```
$("p").text("Hello World");
```

As follows.

```
$("p").each(function (index) {
    $(this).text("Your ID is: " + this.id + " and index: " + index);
});
```

__2. Save changes.

__3. Refresh the browser or enter the URL: **http://localhost/lab/basic.html**



```
Your ID is: p1 and index: 0

Your ID is: p2 and index: 1
```

There is a lot going on with the each() function. It takes as argument a function object. This function is called repeatedly, once for each jQuery object in the collection. The function receives the index of the jQuery object in the collection starting with 0.

Important: The "this" keyword within the iteration function points to the DOM element. We can not directly call the text() method for it since that is not a DOM API. We must obtain the jQuery object using $(this) and then call the text() function to set the text.

Since, "this" is a DOM element, we can call this.id to get it's ID.

Part 2 - Class Selector

One of the common uses of selector is to select all elements that belong to a CSS class. This is done using the format $(.class_name). This is same as the CSS class selector.

Now, we will select one of the paragraphs by its class name and change its background color.

__1. Set the class of the second paragraph to "highlighted".

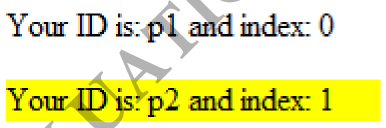
```
<p id="p2" class="highlighted"></p>
```

__2. At the end of the DOM ready handler function add the line shown in bold face.

```
$(function(){
    $("p").each(function (index) {
        $(this).text("Your ID is: " + this.id + " and index: " + index);
    });
    $(".highlighted").css("background", "yellow");
});
```

__3. Save changes.

__4. Refresh the browser. The second paragraph will now have a yellow background.



```
Your ID is: p1 and index: 0

Your ID is: p2 and index: 1
```

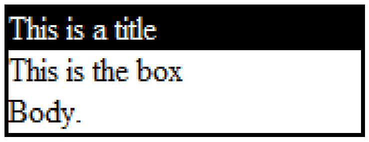
This was also a good example of how to tweak the style of an element using the css() function.

If you know that it is a "p" tag that has the "highlighted" class, you can help jQuery a little by using $(p.highlighted). This will have a performance improvement.

Part 3 - Child Pseudo Selector

So far, we have been using selectors that follow the same syntax as the CSS selector. Now, we will use pseudo selectors that are introduced by jQuery. A pseudo selector is always prefixed by ":".

We will now use jQuery to create a box widget that looks like this.



```
This is a title
This is the box
Body.
```

__1. Copy **C:\LabFiles\template.html** to **APACHE_DIR/htdocs/lab/**.

__2. Rename **template.html** to **box.html**

__3. Open **box.html** in an editor.

__4. Below the line:

```
<!-- Body HTML here -->
```

Add:

```
<div class="box">
<div>This is a title</div>
<div>
This is the box<br/>
Body.
</div>
</div>
```

Basically, any <div> element with style "box" will be converted into a box. The first child <div> element of the box will be converted into the title.

First, we will draw the border around the box.

__5. Below the line:

```
//jQuery code here
```

Add:

```
$("div.box").css("border-style", "solid");
$("div.box").css("border-width", "2px");
```

Now, we will change the background color of the first child <div> element to black. We will change its foreground to white.

If we use $("div div") that will select all div child elements of another div. We don't want that. We only wish to select the first child.

__6. Continue to develop the DOM ready function by adding these lines:

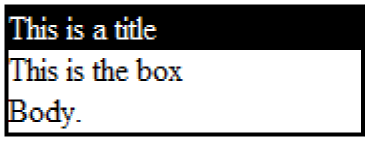
```
$("div.box div:first-child").css("background", "black");
$("div.box div:first-child").css("color", "white");
```

Here, we use the "first-child" pseudo selector. This will select the first <div> child element of any div element that has the box class.

__7. Save changes.

__8. In the browser, enter the URL: **http://localhost/lab/box.html**

__9. You should see the page as follows.


This is a title
This is the box
Body.

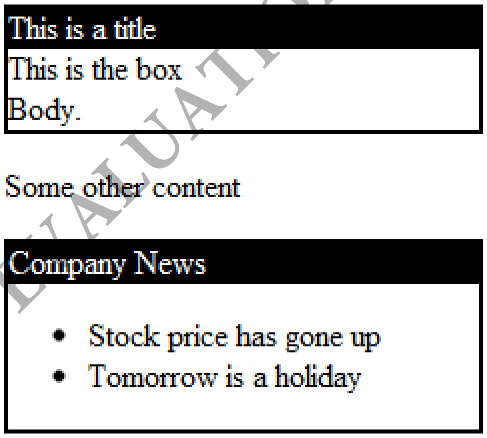
It's now easy to add as many boxes as we want.

__10. Below the last </div> tag, enter:

```
<p>Some other content</p>

<div class="box">
<div>Company News</div>
<ul>
          <li>Stock price has gone up</li>
          <li>Tomorrow is a holiday</li>
</ul>
</div>
```

__11. Save changes and refresh the browser.


This is a title
This is the box
Body.

Some other content

Company News

- Stock price has gone up
- Tomorrow is a holiday

Part 4 - Optimize the Code

Selectors require a lot of DOM tree navigation. This can be slow for a large document. Once you have retrieved a collection, you should reuse it as much as possible. In our case, we are running the same selector multiple times. We should be able to optimize that.

__1. Change the DOM ready function as follows.

```
$(function(){
        //jQuery code here
        var box = $("div.box");
        box.css("border-style", "solid");
        box.css("border-width", "2px");
        var title = $("div.box div:first-child");
        title.css("background", "black");
        title.css("color", "white");

});
```

Now, a selector is run only once.

__2. Save changes.

__3. Refresh the browser. It will look same as before.

Part 5 - Pseudo Selector for Table Rows

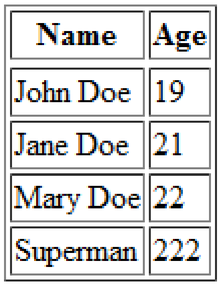
jQuery has :odd and :even pseudo selectors that apply to table rows. They are great for styling alternate rows differently.

__1. Copy **C:\LabFiles\table.html** to **APACHE_DIR/htdocs/lab**.

__2. Open **APACHE_DIR/htdocs/lab/table.html** in your editor.

__3. Note that the header is created using <thead> and the body using <tbody>.

__4. View the page in a browser: **http://localhost/lab/table.html**



Name	Age
John Doe	19
Jane Doe	21
Mary Doe	22
Superman	222

Nothing spectacular. Now, we will apply different colors to every alternate rows.

__5. Below the line:

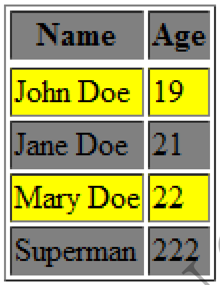
```
//jQuery code here
```

Add:

```
$("tr:even").css("background-color", "grey");
$("tr:odd").css("background-color", "yellow");
```

__6. Save.

__7. Refresh the browser.



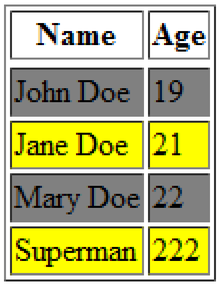
Name	Age
John Doe	19
Jane Doe	21
Mary Doe	22
Superman	222

It worked. But, the <tr> elements within <thead> are also being colored. We can fix that by combining the even and odd selector with child selector. We want only the <tr> elements that are children of <tbody> to be colored.

__8. Change the code as follows.

```
$("tbody tr:even").css("background-color", "grey");
$("tbody tr:odd").css("background-color", "yellow");
```

__9. Save and test again.



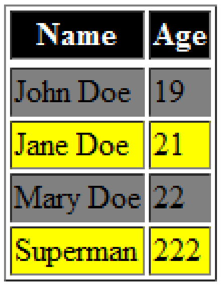
Name	Age
John Doe	19
Jane Doe	21
Mary Doe	22
Superman	222

Now the header is not colored any more. In fact, while we are at it, let's style the header row.

__10. Add this code.

```
$("thead tr").css("background-color", "black");
$("thead tr").css("color", "white");
```

__11. Save and test.



Name	Age
John Doe	19
Jane Doe	21
Mary Doe	22
Superman	222

__12. Close all your editors and web browsers.

Part 6 - Review

In this lab we played with more selectors. We also learned how to apply basic styling using the css() function. As you saw, with only a few lines of code, you can apply consistent look and feel throughout the site.