JQUERY

Lakshman M N
Tech Evangelist
Lakshman.mn@gmail.com

What is jQuery?

- jQuery is a JavaScript framework that eases JavaScript usage
- jQuery abstracts and simplifies a lot of stuff like AJAX calls and DOM manipulation.
- jQuery does not replace JavaScript.
- The code authored with the help of jQuery is JavaScript code.

Getting jQuery

- jQuery needs to be included on the pages it is to be used.
- jQuery can be downloaded from http://www.jquery.com
- "Production" version has been minified and compressed to take up the least space.
- "Development" version hasn't been minified and compressed and helps debugging.

Including jQuery…

- Reference the jQuery.js in the pages using the **<Script>** tag.

```
<script type="text/javascript"
        src="jquery-1.5.1.js"></script>
```

- An alternate approach to hosting the jQuery.js locally is to include it from a CDN(Content Delivery Network)
  - Google and MS host several versions of jQuery
  - These files come from a common URL that other websites could have used too
    - The file could be served from the cache
    - The file could be downloaded from the closest server if needed

Hello World!

- An obligatory "Hello World"

```
<div id="div1"></div>
  <script language="javascript">
     $("#div1").text("Hello, world!");
  </script>
```

  - **$** shortcut used to access jQuery
  - Select an element with id "div1"
- Plain JavaScript would require

```
<div id="div2"></div>
  <script language="javascript">
     document.getElementById("div2")
                .innerHTML = "Hello, world!";
  </script>
```

jQuery01.htm

The Ready Event

- It is a good practice to wait for the document to be fully loaded before working with it
- **$(document).ready** event is fired to indicate that the document is ready for DOM manipulation.

```
<div id="div1"></div>

<script type="text/javascript">
function DocReady()
{
   $("#div1").text("Hello, world!");
}

$(document).ready(DocReady);
</script>
```

jQuery02.htm

The Ready Event…

- The ready event can be associated with an anonymous function.
- This simplifies the number of instructions.

```
<div id="div2"></div>
<script type="text/javascript">
$(document).ready(function()
{
    $("#div2").text("Hello, world!");
});
</script>
```

jQuery02-A.htm

The Ready Event…

- jQuery supports an overloaded version of the constructor that accepts a ready function as a parameter.

```
<div id="div3"></div>
<script type="text/javascript">
$(function()
{
    $("#div3").text("Hello, world!");
});
</script>
```

jQuery02-B.htm

Selectors

Selectors

The #id selector

- An ID attribute of an HTML tag should be unique and can be used to locate the element

```
$("#div1")
```

  - Locates an element with an ID "div1"

The .class selector

- Elements with a specific class can be matched with "." followed by the name of the class

```
$(".bold").css("font-weight","bold")
```

jQuery03.htm

Selectors…

The element selector

- Elements can be matched based on their names

```
$("a")
```

- The class selector can be used with elements of a particular type

```
$("span.bold").css("font-weight","bold");
```

jQuery04.htm

Using Attributes

- jQuery can help locate elements based on attributes

```
$(function() {
  $("[href]").css("font-size","18")
});
```

- All elements having an "**href**" attribute are matched.
- Elements having attributes with a specific value can also be located.

```
$("[href='#']").css("font-style","italic");
```

- All elements having an "**href**" attribute with a value "#" are located.

jQuery05.htm

Using Attributes…

- The **^=** operator can be used to find elements having attributes with values starting with a specific string

```
$("[name^='txt']").css("color","#0000ff");
```

- **$=** operator can be used to find elements having attributes with values ending with a specific string

```
$("textarea[name$='address']").css("font-
    family","Courier")
$("a[href$='.pdf']").css("color","#ff0000");
```

jQuery05-A.htm

Selecting by Position

- **a:first**
  - Matches the first **<a>** element on the page
- **p:odd** or **p:even**
  - Matches every odd or even paragraph
- **li:last-child**
  - Matches the last child of the parent element
- **li a:first**
  - Matches the first **<a>** element under **li**
- **li:nth-child(2)**
  - Matches the second **li** element

jQuery-positionselectors.htm

Generating New HTML

append() and prepend()

- **append()** and **prepend()** help add new content to existing elements

```
$("#list").append("<option>Item 3</option>");
$("#list").prepend("<option>Item 0</option>");
```

- **appendTo()** and **prependTo()** are called on the new elements that need to be added to existing elements.

```
$("<p></p>").text("Hello Appended
          World!!").appendTo("#docBody");
$("<p></p>").text("Hello Prepended

     World!!").prependTo("#docBody");
```

jQuery13.htm
jQuery13-A.htm

before() and after()

- Content may need to be inserted before or after elements.
  - This is unlike **append()** and **prepend()** that add stuff inside an element.

```
$("#span1").before("<b>Before the span</b>");
$("#span1").after("<i>After the span</i>");
```

- **insertBefore()** and **insertAfter()** are called on the content that need to be added to existing elements.

```
$("<b></b>").text("Text inserted before
                   ").insertBefore($
  ("#btnAppend"));
$("<b></b>").text("Text inserted after
                   ").insertAfter($
  ("#btnPrepend"));
```

jQuery14.htm

Adding & Negating elements

- **add(selector)** allows the chaining of a group of selectors together into an or relationship
  - Creates a union of elements that satisfy both of the selectors

```
$("li").add("a").css("color","ff0000")
```

- **not(expression)** removes elements from the matched set according to the value of the expression
  - Selectors that can be passed to the **not()** method are limited to filter expressions that omit element references

```
$("a").not("#css1").css("background-
  color","aqua")
```

jQuery-elementSetAddNot.htm

Getting sets using relationships

- New wrapped sets based on the hierarchical relationships of the wrapped element can be fetched.
  - **`children()`** – returns a set containing children of the wrapped elements
  - **`parent()`** – returns a set containing direct parents of the wrapped element
  - **`next()`** – returns a set containing next siblings of the wrapped element
  - **`prev()`** – returns a set containing previous siblings of the wrapped element

jQuery-elementSetRelationships.htm

DOM Manipulation

Setting and Retrieving Attributes

- **`attr()`** method can be used to change one or more attributes of an element.

```
alert("href : " +
                                        $
  ("#link1").attr("href"));
$("#link1").attr("href",
                "http://www.bing.com");
```

jQuery11.htm

Removing attributes

- **`removeAttr(name)`**
  - Removes the specified attribute from every matched element.

```
$("#txtName").removeAttr("value")
```

jQuery11-A.htm

Setting and Retrieving Data

- DOM manipulation involves setting and retrieving HTML, text and values.
  - **Text**- textual (no HTML) representation of the inner content
  - **Value** – for form elements
  - **HTML** – similar to text but can include markup
- Methods include **`text()`**, **`html()`** and **`val()`**

jQuery10.htm

remove() and empty()

- jQuery provides mechanisms to do away with elements and content
- **`remove()`** – deletes the selected elements (including content)

```
$("#div1").remove();
```

- **`empty()`** – deletes all child elements of the selected elements

```
$("#div1").empty();
```

jQuery15.htm

Events

Binding Event Handlers

- Event handlers on DOM elements can be established with **`bind()`**
- **`bind(eventType, data, listener)`**
  - Establishes a function as a event handler for the specified event type on matched elements

```
$("div").bind("click",function()
 {
     alert($(this).text());
 });
```

jQuery16.htm

one() as specialized bind()

- **one()** establishes an event handler for a one-time activity
- **one(eventType,data,listener)**
  - Once the event handler executes for the first time it is automatically removed.

```
$("div").one("click",function()
 {
     alert($(this).text());
 });
```

jQuery-EventOne.htm

Removing Event Handlers

- Some interactions may require event handlers to be removed based on specific criteria
- **unbind(event, listener)**
  - Removes event handlers from all matched elements
  - Specific handlers are removed by providing a reference to the function originally established as a listener
  - In case no parameters are specified all events are removed from all matched elements

jQuery16-A.htm

Effects

Toggling the display state

- jQuery defines toggle() to toggle the display states of elements between revealed and hidden
- **`hide(speed, callback)`**
  - Causes the elements in the wrapped set to be hidden
    - Speed – optionally specifies the duration of the effect in milliseconds
    - Callback – optional function invoked when the animation completes
- **`show(speed, callback)`**
  - Causes the elements in the wrapped set to be shown
- **`toggle(speed, callback)`**
  - Alternates between **`show()`** and **`hide()`**

jQuery-toggle.htm

Fading Elements

- Simple animations can be accomplished in jQuery.
- Fading an element in and out of visibility is supported.
- `fadeIn()` can accept either "`fast`", "`slow`" or duration in milliseconds.

```
$("#div1").fadeIn("fast");
$("#div2").fadeIn(6000);
```

- Fading an element in and out of visibility depending on its current state

```
$("#div3").fadeToggle();
```

jQuery07.htm

Sliding Elements

- Sliding effects can at times make for a better choice as against fading

```
$("#div1").slideDown("fast");
$("#div2").slideDown(6000);
```

- Sliding an element up or down depending on its current state

```
$("#div3").slideToggle();
```

jQuery08.htm

Custom animations

- **`animate()`** method can be used to create custom animations.
- Any CSS property of an element can be manipulated
- The **`animate()`** method accepts the CSS property to be altered as the first parameter.
- The second parameter specifies the duration of animation in milliseconds

Custom animations…

```
<script type="text/javascript">
$(function()
{
$("#div1").animate( { "left":"500px" } );
$("#div2").animate( { "left":"300px" },
                                    3000 );
$("#div1").animate( { "left":"100px" },
                                    3000 );
});
</script>
```

jQuery09.htm

Avoiding conflicts with other libraries

- If another JavaScript library being used employs the $ variable it could conflict with jQuery.
- jQuery should be put in no-conflict mode immediately after it is loaded to avoid conflicts
- A variable name can be assigned to replace $

```
<script src="prototype.js"></script>
<script src="jquery.js"></script>
<script>var $j = jQuery.noConflict();</
  script>
```

jQuery-noConflict.htm

Utility Methods

- jQuery offers several methods to accomplish routine programming tasks
- `$.trim` – removes leading and trailing whitespace

  `$.trim(value)` – returns the value devoid of whitespace
- `$.each` – iterates over arrays and objects

```
$.each(['Apple',Bloomberg','Carlyle'], function(idx,
  val) {console.log('element ' + idx + 'is ' + val);
});
```

- `$.inArray` – returns a value's index in an array

```
var myArr = ['a','b','c','d','e'];
if($.inArray('d',myArr)!== -1) {
      alert("found!"); }
```

jQuery-Utilities.htm

Utility Methods…

Filtering arrays

- **`$.grep`** finds the elements of an array that satisfy a filter function.
  - The original array is not affected.
- **`$.grep(array, callback, invert)`**
  - Traverses the passed array invoking the callback function for each element
  - A return value true of the callback function causes the current value to be collected
  - If **`invert`** is false or not provided an array is returned consisting of all elements for which callback returns true

jQuery-grep.htm

Utility Methods…

Translating arrays

- **`$.map`** translates all items in an array to a new array of items
- **`$.map(array, callback)`**
  - The callback function returns values that are collected in a new array

Convert to arrays

- **`$.makeArray`** converts an array-like object into a JavaScript array
- **`$.makeArray(object)`**

jQuery-map.htm

Debugging Tools

- There are a few good jQuery development and debugging tools available
- Tools for Firefox
  - Firebug
  - Web Developer Toolbar
- Tools for Internet Explorer
  - Microsoft Internet Explorer Developer Toolbar
  - Microsoft Visual Web Developer

Debugging Tools…

- Tools for Google Chrome
  - Web Developer
  - Pendule
  - Firebug Lite
- Code can be tested online at http://jsbin.com

Resources

- 15 jQuery Plugins
  - http://devsnippets.com/article/reviews/15-jquery-plugins-to-fix-and-beautify-browser-issues.html
- jQuery resources
  - http://www.lateralcode.com/15-exceptional-jquery-resources-and-tutorials/
- jQuery from scratch
  - http://net.tutsplus.com/tutorials/javascript-ajax/15-resources-to-get-you-started-with-jquery-from-scratch/
- Plugins
  - http://jqueryplugins.com

Resources…

- Learning jQuery
  - http://www.learningjquery.com/
- jQuery for designers
  - http://www.jqueryfordesigners.com
- Books
  - jQuery: Visual kickstart Guide
  - jQuery Cookbook