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

Relation Selectors

- Elements can be selected based on their parent element
- Options include
  - Match elements that are a direct child to the parent element

```
$("#div1>b").css("color","#ff0000");
```

  - Match all the way down through the hierarchy

```
$("#div1 b").css("font-family","Forte");
```

jQuery06.htm

Relation Selectors…

```
<ul class="cssList">
  <li><a href="http://www.w3.org/css">CSS</a>
   <ul>
     <li><a href="CSS1">CSS 1</a></li>
     <li><a href="CSS2">CSS 2</a></li>
     <li><a href="CSS2.1">CSS 2.1</a></li>
   </ul>
  </li>
</ul>
```

- Match the inner <a> element

```
$("ul.cssList > li > a").css("color","#ff0000");
```

jQuery06-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

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

Manipulating CSS

- jQuery allows changing the style attribute as well as the classes of an element.

Methods include

- **`hasClass()`** – checks if the element already has a specific class defined
- **`addClass()`** – adds a class name to the element
- **`removeClass()`** – removes a class name from the element

jQuery12.htm
jQuery12-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

Wrapping Elements

- DOM manipulation may require wrapping an element or a set of elements in some markup
- **`wrap(wrapper)`**
  - Wraps the elements of the matched set with the passed HTML tags
- **`wrapInner(wrapper)`**
  - Wraps the contents with the passed HTML tags

jQuery10-A.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

jQuery Event Model

- jQuery abstracts the browser differences in writing event-handling code
- jQuery Event model provides the following features
  - Unified method for establishing event handlers
  - Standard event-type nomenclature
  - Event instance is available as a parameter to the handlers.
  - Multiple handlers for each event type on each element

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

Stopping animations

- Animations may need to be stopped due to a number of reasons
- **`stop()`**
  - Halts all animations currently in progress for the matched set of elements.

jQuery-stopAnim.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

AJAX - Introduction

- Stands for "Asynchronous JavaScript and XML "
- Development technique for creating interactive web applications
- Not a new *Technology* but more of a *Pattern*

Motivation for AJAX

- WebPages always RELOAD and never get UPDATED
- Users wait for the entire page to load even if a single piece of data is needed
- Single request/response restrictions

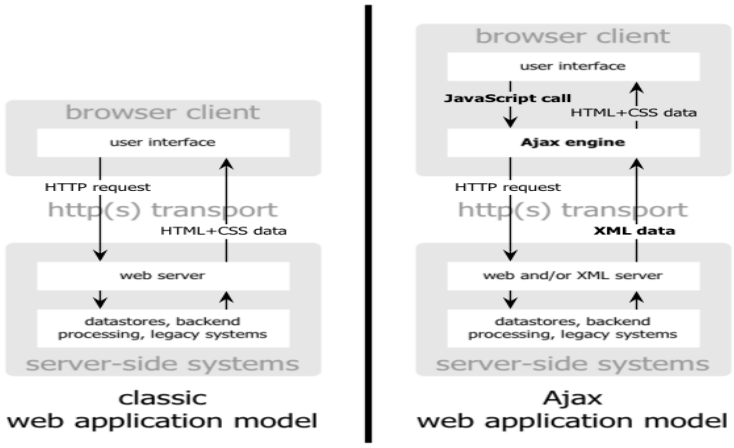
Components

- HTML (or XHTML) and CSS
  - Presenting information
- Document Object Model
  - Dynamic display and interaction with the information
- XMLHttpRequest object
  - Retrieving data **ASYNCHRONOUSLY** from the web server.
- Javascript
  - Binding everything together

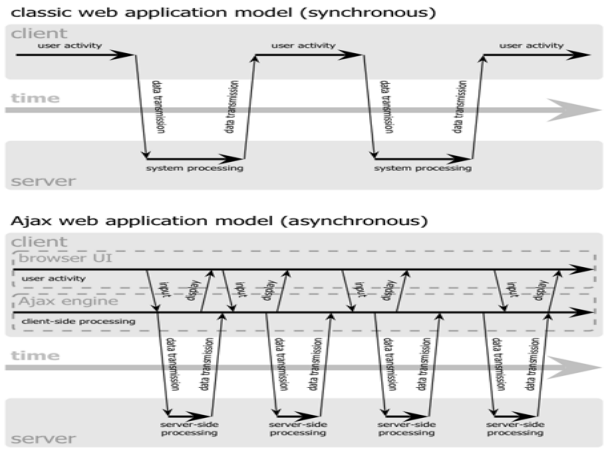
Uses of AJAX Paradigm

- Real-Time Form Data Validation
  - Form data that require server-side validation can be validated in a form "before" the user submits it.
- Auto completion
  - A specific portion of form data may be auto-completed as the user types.
- Master Details Operations
  - Based on a client event, an HTML page can fetch more detailed information on data without refreshing the page.
- Sophisticated UI Controls
  - Controls such as tree controls, menus, and progress bars may be provided without page refreshes.

Web Application and AJAX



Request Processing



Asynchronous processing - XMLHttpRequest

- Allows to kick off an HTTP request in background
- Callbacks kick back into Javascript Code
- Supported in all standard browsers
- Similar to "image" object
  - Dynamically change the URL of the image source without using a page refresh

Example using XMLHttpRequest – Step 1

- Create Object
  - Worry about Browser Specific Creation !
- Example
  - `var requester = new XMLHttpRequest();`
  - `var requester = new ActiveXObject("Microsoft.XMLHTTP");`

Example using XMLHttpRequest – Step 1

- Some browsers expect the response from the server to contain an XML mime-type header.
- To satisfy this add:

```
requester.
        overrideMimeType('text/xml')
```

Using XMLHttpRequest – Step 2

- Transferring data to Server
  - `Open()` to initialize connection to Server
  - `Send()` to send the actual Data
- Example
  - `requester.open("GET", url, true)`
    - The first parameter is the HTTP request method.
    - The second parameter is the url of the page being requested.
    - The third parameter indicates whether the request is asynchronous.

Using XMLHttpRequest – Step 2

- The parameter to the `send()` method is the data to be sent to the server if POST is employed.
- The data should be in the form of a querystring.

```
name=value&name1=value1&name2=value2
```

- In order to POST data the MIME type of the request needs to be changed

```
requester.setRequestHeader('Content-
    Type','application/x-www-form-urlencoded')
```

What happens after sending data ?

- **`XMLHttpRequest`** contacts the server and retrieves the data
  - Can take indeterminate amount of time
- Event Listener to determine when the object has finished retrieving data
  - Specifically listen for changes in "**`readyState`**" variable

Using XMLHttpRequest – Step 3

- Set up a function to handle the event when the **`readyState`** is changed to 4
  - 0 – Uninitialised
  - 1 – Loading
  - 2 – Loaded
  - 3 – Interactive
  - 4 – Completed
- Example

```
requester.onreadystatechange = stateHandler;
```

Using XMLHttpRequest – Step 3 Contd

- Check whether the **`XMLHttpRequest`** object successfully retrieved the data, or was given an error code
- Example

```
if (requester.readyState == 4)
   {
    if (requester.status == 200)
    {
              success();
          }
   }
```

Using XMLHttpRequest – Step 4

- Parse and display data
  - **`responseXML`**
    - DOM-structured object
  - **`responseText`**
    - One complete string
- Example

```
var nameNode = requester.responseXML.
                getElementsByTagName("name")[0];
var nameTextNode = nameNode.childNodes[0];
var name = nameTextNode.nodeValue;
```

AJAX

- AJAX allows data to be loaded in the background and rendered on the page
- Various browsers have different implementations to support AJAX.
- jQuery allows us to add AJAX functionality in a simple browser independent manner.

load() method

- **`load()`** represents a simple method to load data asynchronously
- The element that needs the content loaded into calls the **`load()`** method.
  - The URL is accepted as a parameter
  - A selector can be passed with the URL to get only a part of the page.

```
$(function(){
 $("#div1").load("servedContent.htm");
 $("#div2").load("servedContent.htm #divData");
});
```

jQuery17.htm

load() method…

- The **`load()`** method can accept a couple of parameters.
  - A set of querystring key/value pairs
  - A callback function to be executed when the load method finishes
    - The callback function specifies 3 parameters
      - Resulting content
      - Status
      - The XMLHTTPRequest object

jQuery17-A.htm

get() and post() methods

- **`get()`** and **`post()`** allow for sending an HTTP request to a page and get results.
- Both are provided as static methods
- They accept the URL as a parameter in addition to a callback function.
  - The first parameter of the callback function is the content of the requested page
  - The second parameter is the textual status of the request

jQuery18.htm
http://localhost/jQuery/jQuery18.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