JQuery

# Getting Started

* JQuery is a cross browser JavaScript library.(Download from <http://jquery.com> )
* Advantage of using JQuery:
  + Abstracts cross browser issues so developer can concentrate on application logic
  + Concise & robust code to access DOM using selectors
* JQuery 1.x for IE 6-8 and JQuery 2.x for IE8 or higher or other browsers. (Download compressed for production and uncompressed for debugging)
* Use JQuery
  + Downloaded file //for intranet if users are all local.

**<head>**

**<script type=”text/javascript” src=”jquery.js”></script>**

**</head>**

* + Use JQuery from CDN (Content Delivery Network) like MS, Google – **src has path to jquery script hosted on CDN**. // Caching, Speed , Regional servers
  + Fallback if CDN fails: //best of both worlds

**<head>**

**<script type=”text/javascript” src=”http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js”></script>**

**</head>**

**<script>**

**windows.jquery || document.write(‘<script src=”jquery.js”></script>’)**

**</script>**

* Jquery alias is **$** // shorthand for jquery. (Using that in this document)
* **$(document).ready( function(){**

//perform actions

**});** // document is called when DOM is loaded whereas **winow.onload** is called once page is loaded

* JQuery documentation - <http://api.jquery.com/> (Recommend to bookmark this site!!)

# Selectors

* **$(‘tag’)** ex $(‘p’) //selects all paragraphs in document and returns a collection.
  + **$(‘tag1,tag2,tag3,…)** ex $(‘p,a’)//returns a collection of all paragraphs & links in document
  + **$ (‘ancestor descendant’)** ex (‘table tr’) // returns all tr that are part of table.
* **$(‘#ID’) ex (‘#myID’)** // returns object with given ID. Fastest (=document.getElementById(‘myID’);)
* **$(‘.class’)** ex (‘.myClass) // returns objects with given class.
  + Use tag to optimize **$(‘tag.class’) ex (‘a.myClass’)**
* **$(‘tag[attribute]’)** // returns all elements that have the given attribute
  + **$(‘tag[attribute = ”Value”]’)** ex $(‘input[type=”text”]’)// returns all elements that have the given attribute with specified value. Value is case sensitive
  + **^=** //searches all elements that begin with given value ex $(‘input[value^=”Start Value”]’)
  + **$=** //searches all elements that end with given value ex $(‘input[value^=”End Value”]’)
  + **\*=** //searches all elements that contain given value ex $(‘input[value^=”Conatin Value”]’)
* **$(‘:input’)** //selects all input elements including input, select, textarea, button, image etc
  + **$(‘:input).each(function(){**

**var elem = $(this);**

**alert(elem.val()); //gets value of input element**

**});** // iterates through all the input elements on the document.

* + **elem.val(‘new val’)** //sets the new value to ‘new val’ for given input element
  + To improve efficiency use **$(‘form :input’)** by tag or **$(‘#formId :input’)** by id.
* **$(‘:contains(“value”)’)** ex $(‘div:contains(“sample”)’)//searches all elements that have the given value. Case sensitive search
* **$(‘tr:odd | even’)** //gets all the odd or even rows ex used to style the rows
* **$(‘element:first-child’)** ex $(‘td:first-child’)// find first child of all the given elements

# Interaction with DOM

EACH LOOP FOR COLLECTION

* **$(selector).each(function(index,Element){** // both parameters are optional

Perform action// **$(this)** can be used to wrap current element with JQuery wrapper.

**});** // iterate through each of the item in the selected item collection

* + For better performance, get or set objects value outside the each loop

MODIFY PROPERTIES

* **this.property = “newVal”** // this refers to the current DOM Object & has no access to JQuery APIs. It repkaces the “newVal” for existing property and adds if property does not exist.
* **var val=$(selector).attr(‘property’);** // retrieves the value of given property
  + **$(selector).attr(‘property’,’value’);** //sets the specified value for given property
  + **$(selector).attr(JSON OBJECT);** // modifies multiple properties using JSON object ex:

**$(div).attr({**

**title:’New Title’,**

**style:’border :2px solid black;’**

**});**

* Chaining allows to modify multiple properties ex:

**$(div).att({**

**Title:’hello’,**

**Style:’border:2px’**

**})**

**.css(‘backgroung-color’,’yellow’)**

**.css(‘color’,’black’);**

MODIFYING NODES

* **$(‘HTML string’).appendTo(‘selector’) or $(‘selector).append(‘HTML string’)** // append HTML String to the selected elements
* **$(‘HTML string’).prependTo(‘selector’) or $(‘selector).prepend(‘HTML string’)** // prepend HTML String to the selected elements
* **$(‘selector).wrap(‘DIV HTML string’)** // wrap each object in collection in given div.
* **$(‘selector).remove()** // removes all the selected elements

MODIFYING CLASSES

* **$(‘selector).addClass(‘class1 class2 …’)** // adds given classes to the selected elements
* **$(‘selector).hasClass(‘styleSpecific’)** // returns true if element has matching class
* **$(‘selector).removeClass(‘class1 class2 …’)** // removes given classes from selected elements. **.removeClass()** No parameter means remove all classes.
* **$(‘selector).toggleClass(‘class1’)** //alternates adding / removing the class based on whether class is currently present or absent.
  + **$(‘selector).toggleClass(‘class1’,e.type===’mouseenter’)** // ensures that class is added when event type is mouseenters

# Handling Events

* JQuery provides cross browser event model that is simple and compact
* **$(selector).click(function(eventObject){**

**Code goes here**

**});** //Handling the events in JQuery. Use Ids to attach to specific element. eventObject is optional.

* **$(eventObject.target).attr(‘id’)** // gets the id of the target object
* Chaining can be used to handle multiple events
* **$(selector).on(eventType,handler(eventObject))** //attach event handler to event
  + **$(‘#myID’).on(‘mouseenter mouseleave’, function(e){**

**Handle multiple events**

**});** // attach mouseenter & mouseleave events to the element with id =myID

* + **e.type==’eventName’** //gets the event in case of multiple events
  + In older version less than 1.7 use **bind()**
* **$(‘selector).off(‘event1 event2 …’)** // removes given events from selected elements.
  + **.off()** No parameter means remove all events. In older version less than 1.7 use **unbind()**

BUBBLE UP EVENTS

* Binding events to each input type or row can eat up the memory & degrade performance. So JQuery allows to bubble up the events and handle only up the chain.
* $(selector).live(‘event’, someFunction) //bubbles the event up to the document.
  + $(selector).die(‘event’, someFunction) //stops event handling set by live().
* $(contextObj).delegate(‘selector’,‘event’, someFunction) //bubbles the event up to contextObj
  + .undelegate() //stops event handling set by delegate().
* $(contextObj).on(‘event’ , ‘selector’, someFunction) //used in place of live() & delegate()
  + **$(#MyTable tr).on(‘click’, ‘tr’ , function(e) {**

**Code goes here**

**});** // click event attached to tbody and handled there instead of each row

* + **$(#MyTable tbody).on({**

**mouseenter : function() {**

**Code goes here**

**},**

**mouseleave : function() {**

**Code goes here**

**}**

**});** / multiple events & handlers defined in on() using “map”

HOVER EVENT

* **$(selector).hover(handlerIn, handlerOut**) // shortcut for mouseenter & mouseleave events
  + **$(‘#target’).hover(**

**function() {**

**Code goes here for mouseenter**

**},**

**function() {**

**Code goes here for mouseleave**

**}**

**);**

* + **$(‘#target’).hover( function() {**

**$(this).toggleClass(‘Highlight’);**

**});** // uses only one handler events with the toggle functions

* **$(selector).toggle(handlerI1,handler2,handler3)** // alternates when item is clicked
* Used hover & toggle to swap classes on elements.

# AJAX and JQuery

* JQuery supports following Ajax requests
  + Allows part of page to be updated
  + Cross browser support
  + Simple API
  + GET and POSTS supported
  + Load JSON, XML, HTML or even scripts
* JQuery Ajax functions work with REST APIs or web services or native URLs
* **$(selector).load(url,data,callback)** // loads HTML from server and adds to DOM object
  + URL is only mandatory. Performance can be improved by caching the page.
  + **$(‘#ResDiv’).load(“Test.html #divInTest”)** // Loads the specified Div from given URL
  + **$(‘#ResDiv’).load(“Customers.aspx”, {PageSize:10})** // Custoemrs.aspx accepts PageSize parameter and returns html. Multiple parameters can be passed as JSON Object
  + **$(‘#ResDiv’).load(“Test.html”, function (response, status, xhr){**

**Callback code goes here**

**});**// if Text.heml is not found, status is “error” & xhr.text has error message

* **$.get(url,data,callback,datatype)** //global function to retrieve raw data from server (HTTP GET)
  + Get requests are cached, can be bookmarked, seen in browser history
  + **$.get(“Test.html”, function (data){**

**Callback code goes here ex $(‘ResDiv).html(data)**

**});**// if Text.heml is not found, status is “error” & xhr.text has error message

* + **$.get(“Customer.aspx”, {id:1}, function (data){**

**Callback code goes here ex $(‘ResDiv).html(data.firstName)**

**},’json’);**// returns customer data for customer id=1 in JSON format

* **$.getJSON(url,data,callback) //global function to retrieve JSON data from server**
  + **$.getJSON(“Customer.aspx”, {id:1}, function (data){**

**Callback code goes here ex $(‘ResDiv).html(data.firstName)**

**});**// returns customer data for customer id=1 in JSON format

* **$.post(url,data,callback,datatype)** //global function to update data to server & retrieve results
  + HTTP POST requests are used to send large or sensitive data
  + **$.post(“Customer.aspx”, {id:1}, function (data){**

**Callback code goes here ex $(‘ResDiv).html(data.firstName)**

* + **},’json’);**// returns customer data for customer id=1 in JSON format **$.post(“CustomerService.svc/GetCustomers”, null, function (data){**

**Callback code goes here ex $(‘ResDiv).html(data.d[0].firstName)**

**},’json’);**// returns customer data in JSON format from Ajax enabled WCF Service

* **$.ajax(settings)** // provides core functionality for AJAX calls
  + **$.ajax({**

**url : “Customers.svc/InsertCustomers”,**

**data : customer,**

**dataType : ‘json’,**

**success : function (data,staus,xhr){**

**success code goes here**

**},**

**error : function (data,staus,xhr){**

**error code goes here**

**}**

**});**

* + Additional Properties that can be specified are : contentType, type (GET |POST)

# Common Properties

* **$(selector).length** //length of the collection
* **$(selector).css(‘Property’,’Value’) ;** //applies given style property to all selected items.
  + **Use JSON Object to modify multiple styles**
* **$(selector).html()** //returns the html of given object
  + **$(selector).html(‘new value’)** // sets the new value as html of given object.

# JQuery Common Bugs

Code Review Checklist

* Ensure that Markup formed is valid. Use template for formatting

<script id=”custom-list” type=”text\template”> <ul>

<li> test1 </li>

<li> test2 </li>

</ul>

</script>//type doesn’t matter till browser doesn’t understand it

$($(‘#custom-list’).html)).appendTo(“ResDiv”);

* Avoid using JavaScript keywords as identifiers i.e. function/variable /member /array /property name.
  + Use the member/array/property name in double quotes ex addClass(obj[“class”]);
* Watch out for trailing commas and semicolon in your objects and arrays esp. JSON Objects.
* Avoid using special characters in selectors. If used, escape them with “\\” ex (‘#foo\\.bar’)
* False value in JavaScript is 0 , -0 , false, null, undefined , “” , NaN. Everything else is true
  + Even empty Objects & arrays. Use obj.length or array.length for condition check
* Use .closest() instead of .parent() to traverse up the DOM because JQuery ends up wrapping the code with divs at runtime.
* Use .prop() instead of .attr() to get /set value of a checkbox ex to check if checkbox is toggled
  + elem.checked or $(elem).prop(“checked”) or $(elem).is(“:checked”) //prop() changes with checkbox state whereas attr() is undefined
* Use .detach() to temporarily remove DOM instead of remove(). .detach() keeps all the bound events & jQuery data associated to be reinserted.
* Use Event delegation so that events bubble up for all children even the newly added ones.

Common Bugs

* HTTPS site tries to load HTTP resource (OLD IE warns). Move HTTP site to HTTPS

**<script src=**[**http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js**](http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js)**></script>**

* + Omit HTTP: in resource.

**<script src=**[**//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js**](http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js)**></script>**

* + Replacing all HTTP with HTTPS is cumbersome solution
* Using conflicting libraries with jquery.js like prototype.js
  + Use jQuery.noConflict() – this reverts $ back to what it was in prototype.js. Use jQuery() and not $() to call jQuery methods.
  + Pass $ as function parameter so inside the function $ refers jQuery

jQuery(document).ready(function($){...Safe to use $ as jQuery …}); for DOM

jQuery(function($){…Safe to use $ as jQuery …}); //shortcut for line above

(function($) {… Safe to use $ as jQuery …}(jquery)); //anonymous function that has jQuery as argument

* .add() creates a new set and leaves the original set unchanged
  + var pdiv =$(‘p’); pdiv.add(‘div’); pdiv.fadeIn(); // fades only the p elements
  + var pdiv =$(‘p’); pdiv = pdiv.add(‘div’); pdiv.fadeIn(); // assign new set explicitly to variable
  + $(‘p’).add(‘div’).fadeIn(); // use chaining
* Wait for DOM to be ready if accessing an element from DOM
  + $(document).ready(function($){...Code …}); or $(function($){…Code…}); // Use .ready()
  + $(document).on(‘ready’ , function($){...Code …}); // won’t call handler if it already occurred
  + Put the script @ bottom of page
  + If using Require.js, use domReady function.
* JQuery sets ‘this’ to the DOM element in question ex if button is clicked this will refer to button
  + $.proxy(event handler,object); // enables to set this as object in given event handler
* Use String.fromCharCode(e.which) instead of String.fromCharCode(e.keyCode)
  + Avoid cross browser inconsistencies
* Remove all Event handler of given type ex : $(“#shapes”).off(“click”)
  + Use second parameter in on & off to refer to same instance

$(“#shapes”).on(“click”, eventName) ; $(“#shapes”).off(“click”, eventName)

* + Use namespace to remove handlers

$(“#shapes”).on(“click.shape”, eventName) ; $(“#shapes”).off(“click.shape”)

* + $(“#shapes”).off(“.shape”) // Removes all event handlers in that namespace
  + $(“#shapes”).off(“.shape”,”\*\*”) // Removes only delegatedevent handlers in that namespace and not traditional
* Return false in .each function means to break out of loop. Return true means continue in .next loop

Tricky Concepts

* To pass an array to $.when use $.when.apply($,promises)
* Nth-child selector
  + $(‘li:nth-child(1)’) //gives the first child from all the lists on page
  + $(‘#listname li:nth-child(1)’) //gives the first child from list with id listname
  + $(‘#listname li:eq(0)’) or $(‘#listname li:first’) //same as above
  + $(‘#listname’).find(“li:first”) //better to read and in performance
* Access List items
  + .nextAll() //gets all the list items after the current item
  + .prevAll() //gets all the list items before the current item
  + .siblings() //gets all the list items other than the current item ex .siblings().removeclass()
* triggerHandler(“click”) – Only matches 1st element. Event does not bubble to DOM. Not chainable
  + Use trigger(“click”) instead. //chainable, matches all elements, bubble up in DOM
* e.target – returns the element that registered the event or its descendent. Useful in event delegation
  + e.currentTraget – returns the current DOM element in bubbling phase.
* Asynchronous Code
  + Flow of call ->Ajax function -> Any code after Ajax function -> Success or Error Event handler based on response -> Complete Event handler showing that ajax call is complete.
    - Success has data, status, xhr as parameters
    - Error has xhr, status, error as parameters
    - Success has xhr, status as parameters
  + $getJSON(“URL”).done(function) or $.ajax({url: “url”,dataType:”json”}).done(function) // use the function with data parameter to retrieve the result from url and display in DOM.
* $getJSON(“URL”) – URL refers to http://
  + Browser same-origin policy requires that request matches same domain , protocol & port #
  + Append “&callback=?” to the URL or to ajax method set dataType to “jsonp”
* Animation Methods :
  + Each element has it’s own internal queue. Animation method add to queue & jQuery moves to next when effect completes.
  + .stop(true,true) // stops the animation by clearing the queue.
  + fadeIn(“slow”).remove() // removes the animation method even before executing. To correct it put remove in complete event handler of animation method
  + Common animation methods : fadeIN() , fadeout(), slideDown() , slideUp()
* $.grep() iterates over every item in the array. There is no break mechanism.
* Try feature detection instead of browser detection. Use jquery-migrate.js for backward compatibility
* Ensure jquery plugin methods return this so as to keep the methods chainable.
* $.extend({}, default value, options) //use empty object in first to ensure that default is not overwritten
* Use pushStack() for plugins that changes the stack