**Why use Jquery.?**

1. It’s a single java script file
2. **To overcome all cross browser issues. Jquery takes care of handling all these issues behind scenes. – Important reason**
3. It has a powerful feature Selectors -> These selectors have the ability to select and manipulate DOM object
4. It has a rich event architecture.
5. It has the support for Ajax(Ability to go to the server to fetch the data needed)
6. Support for plugins

Other reasons:

1. Easily locate elements with a specific class with a single line of code
2. Applying styles to elements easily
3. Handling events in cross browser manner. This greatly reduces the number of hours spent by the developers in cross browser issues.

**How to use jquery?**

Downloading it from your own server

1. Go to jquery.com
2. Download the jquery script
3. Use the following script in your page

<head>

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

</head>

Downloading it from Microsoft/google servers(Content Delivery Network – Large distributed system of servers deployed in multiple datacenters across the Internet).

*Advantages of downloading it from different data centers is*

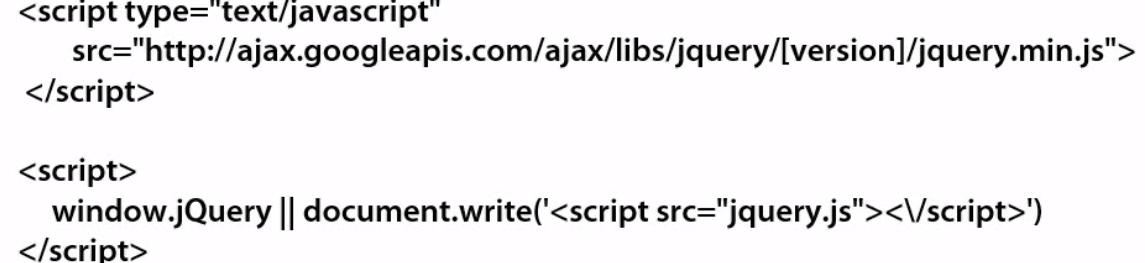
caching(you first hit a page which uses the google url to get the jquery script, script is cached for the domain,then you hit another page that uses the same to get the script, script is not reloaded instead the cached script is used),

speed(Google and Microsoft has multiple data centers across the globe so the script is downloaded from the location where you are rather than from the location where your server is),

parallelism(since this is a different domain from your actual application domain, they can be retrieved in parallel)

*Disadvantages*

Network problems, then the file cant be downloaded. In that case to be on the safe side, we can use the following snippet, If the script can be downloaded from CDN, then it will be available in jQuery property in window object, but if there are issues in getting the script, then it can be got from local.



**Jquery ready function**

$(document).ready(function(){

});

// when the jquery script is loaded from CDN or from local, it will be available in jquery property in window object. $ is an alias of jquery property in window object. ready() is called once DOM hierarchy is loaded but before all images are loaded.

**Jquery Documentation**

Goto jquery.com and choose APIDocumentation to see all the details

**Selectors**

When html gets loaded, it loads all the object into the memory in document object model, jquery allows to go to the memory into the DOM and select different nodes/elements from it and then it helps to manipulate those elements in the DOM.

Types of Selectors : Fastest Selector(Id & Element Selectors), Slowest Selector(Class selector)

*Syntax* $(selectorExpression) or jQuery(selectorExpression)

*Selectors by tag name*

$(‘p’)// selects all paragraph elements

$(‘p,a’)// selects all paragraph & anchor elements

$(‘table tr’)//this is of syntax $(‘ancestor descendant’) . This selects all tr elements within the table elements

*Selecting Nodes by Id*

Use # character to select elements by id. $(‘#myId’) selects <p id=”myId”> element

*Selecting Nodes by Class Name*

Use . character to select elements by class name. $(‘.myClass’) selects <p class=”myClass”> element

$(‘div.myClass’) selects <div class=”myClass”> element

*Selecting Nodes by AttributeValue*

Use brackets [attribute] to select based on attribute name or attribute value

$(‘a[title]’) // selects all <a> elements that have a title attribute

$(‘a[title = “Programming Info”]’)// selects all <a> elements that have a title attribute value as Programming Info

*Selecting Input Elements*

$(‘:input’) // selects all input elements including input, select, textarea, button, image, radio and more

To use the jquery methods, you have to wrap the object in jquery selector.

Var inputs = $(‘:input’);

inputs[1].val();// Error

$(inputs[1]).val();// This will return the actual value

NOTE: <http://codylindley.com/jqueryselectors/>

This site helps to test the selectors

**Iterating through Nodes**

$(‘<tagName>’).each(function(index, element){

});

*Difference between DOM object and jQuery Object.?*

 A **DOM object represents a visual or functional object on the page which was created from the original HTML document**. The **advantage with working with DOM objects** is that you have direct access to everything you need to manipulate the HTML element. The **disadvantage of DOM objects** is that most of the attached functions and attributes are things that the browser needs and are not necessarily useful when you’re working with JavaScript. It makes working with them a little slower, at least for less-experienced developers.

jQuery objects are wrapper objects around single or multiple DOM elements. The jQuery objects (though technically still JavaScript objects) provide access to the wrapped DOM elements — however, in a much different, much easier, and often much more effective way.

**Modifying properties and attributes**

this.title // will access the DOM object’s title property

$(‘img’).attr(‘title’) // will access the jquery object’s attribute title

$(‘img’).attr({

Title : ‘’,

Style : ‘’

});// will modify the jquery object’s multiple attributes

$(‘img’).attr(‘title’,’’).css(‘color’,’black’);// Object chaining.

Object chaining is more efficient one as it does all modifications in one shot. Whereas when updating multiplt objects using json, it has to go through multiple iterations.

**Adding and Removing Nodes**

.append()

.appendTo()

.prepend()

.prependTo()

.remove()

**Jquery Event Model Benefits:** It has a rich event architecture. All cross browser issues related to events are taken care in jquery within the events.

**Handling Events**

Refer <http://api.jquery.com/category/events>

Type jquery source viewer in google to view what jquery does behind the scene

James.padolsey.com/jquery

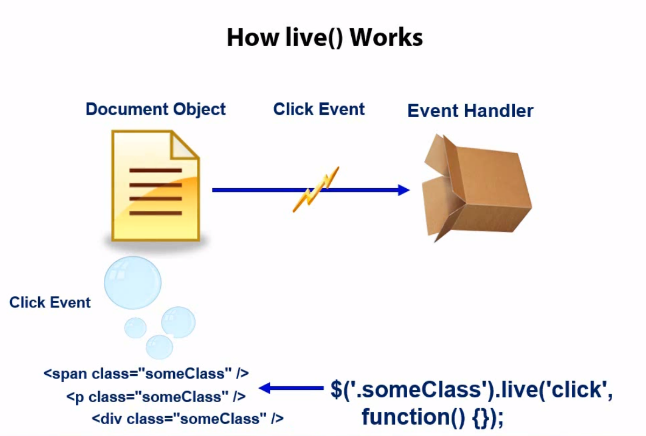
**Binding to events**

On & off have replaced bind & unbind() from jquery version 1.7

**Live, delegate,on**

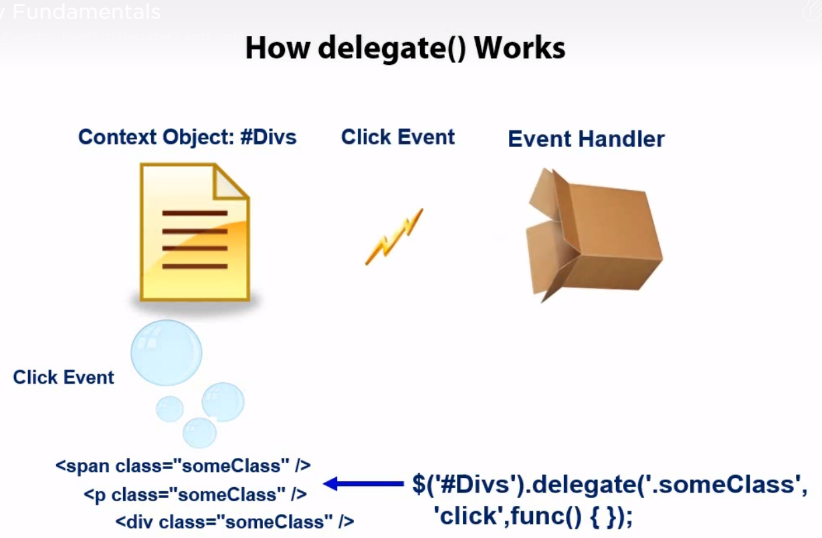
Consider the situation, you have 300 rows in a table and how do you know if a user clicks/mouses over a tr.? Identify the trs using selector and you attach the call back event. This will end up adding 300 event handlers. This overloads the memory. To avoids this,live, delegate and on are introduced in the latest version of jquery. They will attach event handlers to the higher level. They allow new DOM elements to be automatically be attached to an event handler. Allow children to be added to a container without explicitly attaching an event handler to each child.

Live()/die() are removed in jquery 1.9. Event handlers can be set using live(). The DOM object handles events by default. Works even when new objects are added in DOM



Normally, when we write click event on someclass, the 3 different event handlers will be attached to these 3 elements and this will load the memory. But now with live, function will be attached to the document object instead of the element with .someclass. Only 1 event handler will be attached to the DOM object. Now the call back function will be called whenever the element with.someclass is clicked. This will have a bubble up effect on document object.

Downside of this is, it will all the way bubble up to DOM object. To fix this, newer version of live called delegate was introduced in jquery 1.4



Any child of the object with the id Divs that has someClass when clicked will bubble up to the parent with the id Divs instead of the DOM object directly.

On() is a new replacement for bind(),live() and delegate().

$(div).on(‘click’, function(){

// when the div element is clicked, function will be called

});

$(#Mytable tbody).on(“click”, “tr”, function(){

// On a click of tr, event will be bubbled up to the tbody in the table with the id Mytable

});

Multiple events and handlers can be defined in on() using a “map”

$(#Mytable tr). On ({

“mouseover” : function(){},

“mousedown” : function(){}

});

**Handling Hover Events**

$(selector).hover(handlerIn, handlerOut)// hnadlerIn is equivalent to mouseenter and handlerOut is equivalent to mouseleave

$(selector).hover(handlerInOut)// same call back function will be called when its hovered in/out

**Jquery Ajax functions:**

**Load**

$(selector).load(url, data, callback function)

$(document). Ready(function()

{

$(#mybutton).click(function(){

$(#mydiv).load(sample.html);// Displays the entire sample.html in mydiv

$(#mydiv).load(‘sample.html #myTOC’);// loads the entire sample.html and filters only myTOC and displays in mydiv

$(#mydiv).load(‘sample.html’ , {PageSize: 25});//displays only 25 pages

});

});

$(#mydiv).load(notfound.html, function(response, status,xhr){

If(status == ‘error’){

Alert(xhr.statusText);  
}

});

**Get**

$.get(url,data,callbackfunction,datatype);

**getJSON**

$.getJSON(url,data,callbackfunction);

**Post**

$.post(url,data,callbackfunction,datatype);

**Ajax()**

**Note:**

To convert json to string, use json stringyfy script. First refer json2.js script file and then say, JSON.stringyfy(‘’);

Use Fiddler to monitor the http traffic, normally localhost will not work in fiddler. To make it work, add a dot. **Localhost.:9080**

Reference: Jquery Fundamentals in pluralsight

Latest version of Jquery is 3.3.1

* Jquery is not a replacement for Javascript
* Jquery is not w3c std
* It’s a library for client scripting
* Event.PreventDefault – This will prevent the default functionality from happening. When you call on the anchor tag, it will not take you to the appropriate link. Default functionality will be prevented
* Javascript is a scripting language. Jquery is a library built in the JS language that helps to use the JS language.
* Starting point of execution in jquery is $(document).ready() function which is executed when DOM is loaded
* Can we use our own character instead of $.? Yes, you can use it using jquery.noConflict()[Other client side libraries can be used with Jquery. They also use $ as their global fn and use variables. To avoid conflict, we use this function like var$j=jquery.noConflict(), then use $j everywhere in place of jquery]
* .min.js is the deployment version -minimal size. .js is the development version
* If there are multiple selectors, execution starts from the last selector
* Native Javascript is always faster than jquery.

Document.getElementById(‘txt’) is faster compared to $(‘#txt’), as it internally uses the JS call

* Jquery.size() & jquery.length returns the number of element in the object. You should always use length as it doesn’t have the overhead of a function call
* Parent() travel only one level in DOM tree. Parents() search the whole DOM tree
* Eq() – returns a element as a jquery object. get() return a DOM element
* Jquery.fx.off – disable Jquery animation
* Stop currently running animation , use stop()
* Plugins available in interaction section : Draggable, Droppable, Resizable, selectable, sortable
* Diff between body.load() and document.ready – document.ready is called when the DOM is loaded. It doesn’t wait till other resources are loaded and they can be used multiple times in a page. Load() is used only once and will be called when all the resources are loaded
* Bind and live are used to attach the events for the controls. Bind is used with static elements. Live is with dynamic elements
* Event.stoppropagation – stop the event from bubbling from the child elements. Prevents parent from being notified to the child events
* Children() is used to find child controls only one level. Find() is used find elements in all levels in the DOM tree
* We can delay the execution of document.ready by holdReady method
* Detach() and remove() - .detach() retains all child elements associated with the removed element, but keeps the data and event handlers of the removed element. Remove() removes all the data, event handlers and child elements. empty() will remove all content and data from selected element but not the selected element
* Cloning : .clone() performs the deep copy meaning it will copy the matched element as well as their descendent element and text nodes. It will not copy the objects and arrays associated with the selected element. It will be available common to the copied element and the original element. Event handlers are not copied unless you specify withDataAndEvents parameters to true.