**jQuery**

* jQuery is a JavaScript library which makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler.
* Makes it easier to use javascript in your websites
* Adding jQuery to your site
  + Goto jQuery.com, download the latest stable production version of jQuery.
  + Using google CDN : <https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js>
* jQuery has the following
  + Lots of builtin functions
  + Selectors
  + Events
* Syntax
  + jQuery(selector).action();
  + **jQuery / $** : defines jQuery // always use **jQuery** instead of $ to avoid conflict with other libraries
  + **selector** : to find elements from HTML document
  + **action** : action to be performed on the matched element
* Example :
  + jQuery(document).ready(function() {

jQuery(this).hide() // hides the current element

jQuery(“h1”).hide(); // hides all h1 tag in the document

jQuery(“.h1\_class”).hide(); // hides all elements which has css class h1\_class

jQuery(“#test\_cls”).hide(); // hides element which has id set to test\_cls

});

**Document Ready Event**

* Most widely used jQuery event
* Checks whether the document has finished loading and executes all jQuery code written within document ready event
* jQuery(document).ready(function() {  
   jQuery(“.paragraph\_cls”).hide();  
  });  
  The above example waits for document to load completely and then finds all elements which has class **paragraph\_cls** and hides those elements

**Basic Selectors**

* Used to select elements from html document.
* Similar to javascript we have jQuery implementation of the following basic selectors
* **Element selector**
  + jQuery(“p”).hide(); // hides all the paragraph tags
* **ID selector**
  + jQuery(“#para\_id”).hide(); // hides all element which has id para\_id
* **Class selector**
  + jQuery(“.heading\_cls”).hide(); // hides all elements which has class heading\_cls

**jQuery selectors**

* **[name|=”value”] :** Selects elements that have the specified attribute with a value either equal to a given string or starting with that string followed by a hyphen (-)  
  + jQuery("p[details|='Arun']").hide();
  + Hides all paragraph elements which has attribute **details** and value either **Arun** or **Arun followed by -**
* **[name\*=”value”] :** Selects elements that have the specified attribute with a value containing a given substring.  
  + jQuery("p[details\*='Arun']").hide();
  + Hides all paragraph elements which has attribute details and value has which contains **Arun**
* **[name~=”value”] :** Selects elements that have the specified attribute with a value containing a given word  
  + jQuery("p[details~='Arun']").hide();
  + Hides all paragraph elements which has attribute details and value contains word Arun delimited by spaces
* **[name$=”value”] :** Selects elements that have the specified attribute with a value ending exactly with a given string.   
  + jQuery("p[details$='Arun']").hide();
  + Hides all paragraph elements which has details attribute with value ending with Arun and the comparison is case sensitive
* **[name=”value”] :** Selects elements that have the specified attribute with a value exactly equal to a certain value
  + jQuery("p[details='Arun']").hide();
  + Hides all paragraph element which has details attribute and has value exactly equal to Arun
* **[name!=”value”] :** Select elements that either don't have the specified attribute, or do have the specified attribute but not with a certain value.  
  + jQuery("p[details!='Arun']").hide();
  + Hides all paragraph elements which doesnt have details attribute or elements which has details attribute and the value not equal to Arun
* **[name^=”value”] :** Selects elements that have the specified attribute with a value beginning exactly with a given string.  
  + jQuery("p[details^='Arun']").hide();
  + Hides all paragraph elements which has details attribute and value starting with Arun
* **:contains() :** Select all elements that contain the specified text.  
  + jQuery("p:contains('Arun')").hide();
  + Hides all paragraph elements which contains the text Arun
* **(“parent > child”) :** Selects all direct child elements specified by "child" of elements specified by "parent"  
  + jQuery("p > span").hide()
  + Hides all span elements which is a direct child of paragraph element.
* **:disabled :** Selects all elements that are disabled  
  + jQuery("input:disabled").hide();
  + Hides all input element which are disabled.
* **:first-child :** Selects all elements that are the first child of their parent.  
  + jQuery("p > span:first-child").hide();
  + Hides all first child span inside all paragraph element
* **:first :** Selects the first matched DOM element  
  + jQuery("span:first").hide();
* **:has() :** Selects elements which contain at least one element that matches the specified selector.
  + jQuery("p:has(span)").hide();
  + Hides all span element which contains a span element
* **:last-child :** Selects all elements that are the last child of their parent.  
  + jQuery("p > span:last-child").hide();
  + Hides all last child span inside all paragraph element
* **:last :** Selects the last matched DOM element  
  + jQuery("span:last").hide();

**Attributes**

* Methods used to set and get DOM attributes of elements
* **.attr() :** get / set the value of an attribute
  + jQuery(“#test”).attr(“name”, “arun”); // sets the value of attribute name to arun
  + var name = jQuery(“#test”).attr(“name”); // gets the value of attribute name
* **.addClass() / removeClass() :** add / remove css class to an element
  + jQuery(“#test”).addClass(“test\_cls”);
  + jQuery(“#test”).removeClass(“test\_cls”);
* **.removeAttr() :** used to remove an attribute from an element
  + jQuery(“#test”).removeAttr(“name”);
* **.html() :** get / set html contents of an html element
  + jQuery(“#test”).html(“this is a test information”); // set html content
  + var html\_content = jQuery(“#test”).html(); // gets html content;
* **.val() :** get / set the value of html element
  + jQuery(“#test”).val(“test value”); // sets value of html element
  + var value = jQuery(“#test”).val(); // gets current value of html element
* **.toggleClass() :** toggles css class specified for an element
  + jQuery(“#test”).toggleClass( "active" );
* **.css() :** allows to add style properties to an element dynamically
  + jQuery(“#test”).css({“color”:”#FF0000”, “background-color”:”#000”} );

**Positioning functions**

* .append()
* .prepend()
* before()
* after()

**Events**

* These methods are used to register behaviors to take effect when the user interacts with the html document
* In jQuery events are classified into following

**Browser Events**

* **.resize() :** used to attach event handler to resize event.  
  + jQuery( window ).resize(function() {  
     alert(jQuery(this).width());  
    });
  + Above example alerts the width of window when window is resized.
* **.scroll() :** attaches event handler to scroll event of an element  
  + jQuery("#test").scroll(function() { alert(“Scroll captured...”) });

**Document Loading Events**

* **.load() :** attach to event handler load event
  + jQuery( window ).load(function() { alert(“Document fully loaded including all the assets”); });
* **.ready() :** fires when DOM is fully loaded.
  + jQuery(document).ready(function() { alert(“DOM full loaded, Still waiting for all the other assets to load”); });
* **.unload() :** fire when a user navigates away from the page.
  + jQuery( window ).unload(function() { alert(“User left the page”); });

**Event handler attachment**

* **.on() :** Attach an event handler function for one or more events to the selected elements.
  + jQuery("p").on("click, mouseover", function(){ jQuery(this).hide(); });
  + The above example can also be written as
  + jQuery(document).on(“click, mouseover”, “p”, function() { jQuery(this).hide(); });
  + You can also specify multiple events.
  + jQuery(document).on({  
     mouseenter: function(){  
     jQuery(this).css("background-color", "red");  
     },   
     mouseleave: function(){  
     jQuery(this).css("background-color", "green");  
     },   
     click: function(){  
     jQuery(this).css("background-color", "yellow");  
     }   
     }, "#arun");
* **.off() :** Removes event handler
  + jQuery(document).off(fn\_obj, "#arun");

**Form Events**

* **.blur() :** attaches event handler to blur event
  + jQuery(document).on(“blur”, ‘#test’, function() {  
     alert(“Blur event”);  
    });
* **.change() :** attaches event handler to change event
* **.focus() :** attaches event handler to focus event
* **select() :** attaches event handler to select event
  + jQuery(document).on('click', '#sel\_text', function(){  
    jQuery("#text\_sel\_p").select();  
    });
* **submit() :** attaches event handler to submit event
  + jQuery(document).on(“submit”, “#test\_id”, function() { alert(“Submit button clicked”); });

**Keyboard Events**

* **.keydown() :** The keydown event is sent to an element when the user presses a key on the keyboard
  + jQuery( "#test" ).keydown(function() {  
     alert( "keydown event." );  
    });
* **.keyup() :** The keyup event is sent to an element when the user releases a key on the keyboard

**Mouse Events**

* **click():** attaches mouse click event
  + jQuery( "#test\_p" ).click(function() {  
     alert( "Click event fired" );  
    });
* **.dblclick() :** attaches mouse double click event
  + jQuery( "#test\_p" ).dblclick(function() {  
     alert( "Double click event fired" );  
    });
* **.hover() :** Bind one or two handlers to the matched elements, to be executed when the mouse pointer enters and leaves the elements.
  + jQuery( "td" ).hover(  
     function() {  
     jQuery( this ).addClass( "hover\_cls" );  
     }, function() {  
     jQuery( this ).removeClass( "hover\_cls" );  
     }  
    );
* **.mouseover() :** The mouseover event is sent to an element when the mouse pointer enters the element
  + jQuery( "#outer" ).mouseover(function() {  
     alert(“mouseover event fired”);  
    });
* **.mouseout() :** The mouseout event is sent to an element when the mouse pointer leaves the element
* **.toggle() :** Bind two or more handlers to the matched elements, to be executed on alternate clicks
  + jQuery( "#test\_elem" ).click(function() {  
     jQuery(‘p#p\_elem’).toggle()  
    });

**Effects**

* The jQuery library provides several techniques for adding animation to a web page.
* **hide() / show() :** 
  + jQuery(“#test\_elem”).hide()
  + jQuery(“#test\_elem”).show()
* .**animate() :**
  + jQuery("#animate\_div").animate({  
      
    width:"90%",  
    border:"10px solid red",  
    "font-size":"20px",  
    padding:"20px"  
      
    }, 1500);
* **fadeIn :**
  + jQuery("#animate\_div").fadeIn();
* **fadeOut :**
  + jQuery("#animate\_div").fadeOut();
* **delay() :**
  + jQuery("#animate\_div").fadeIn().delay(2000).fadeOut();
* **fadeToggle():** 
  + jQuery("#animate\_div").fadeToggle();
* **slideDown():** 
  + jQuery("#animate\_div").slideDown();
* **slideUp() :**
  + jQuery("#animate\_div").slideUp();
* **slideToggle() :**
  + jQuery("#animate\_div").slideToggle();
* **Callback :**
  + For all the the effects above can have callback functions
  + jQuery("animate\_div").slideUp("slow", function(){  
     alert("Callback function called");  
     });

**Traversing DOM**

* **Ancestors :** includes functions to access current element’s parent elements  
  + **parent():** returns the direct parent element of the selected element
  + jQuery("animate\_div").parent();
  + **parents():** return all the ancestor elements of the selected element
  + jQuery("animate\_div").parents();
  + **parentsUntil() :** return all the ancestor element till a selected position
  + jQuery("animate\_div").parentsUntil(“#test\_div”);
* **Descendants :** includes functions to access current element’s child element  
  + **children() :** returns direct child element of the selected element
  + jQuery("animate\_div").children();
  + jQuery("animate\_div").children(“div.animate\_cls”)
  + **find():** returns descendant elements of the selected element, all the way down to the last descendant
  + jQuery("animate\_div").find(“p”);
* **Siblings() :**
  + **siblings() :** returns all sibling elements of the selected element.
  + jQuery("animate\_div").siblings();
  + jQuery("animate\_div").siblings(“#test\_div”);
  + **next() :** return next sibling element of the selected element
  + **nextAll() :** return all next sibling element of the selected element
  + **nextUntil() :** return all next siblings till the selected positions
  + Similar to next() we have all the functions available for fetching previous siblings
  + prev(), prevAll() & prevUntil()

**jQuery.each()**

var obj = {

"name": "Arun",

"loc": "India"

};

jQuery.each( obj, function( key, value ) {

alert( key + ": " + value );

});

**.each()**

jQuery( "p" ).each(function() {

jQuery( this ).addClass( "cust\_cls" );

});

**preventDefault()**

Prevents the default action of the element

jQuery(document).on('click', 'a', function(e) {

e.preventDefault();

});

**event.stopPropagation()**

Prevents the event from bubbling up the DOM tree, preventing any parent handlers from being notified of the event.

jQuery('#btn\_div').click(function(e) {

e.stopPropagation();

alert('Hello Marlabs...!!!');

});

**Ajax**

* Ajax stands for asynchronous JavaScript and XML
* It runs in the background
* Helps you to add update content in your web page without reloading it
* Allows to request / receive data from server even after the page has been fully loaded
* jQuery(document).on('click', '#btn\_ajax', function () {  
   jQuery.ajax({  
   'type': 'POST', // type of request  
   'url': 'ajax\_content.php', // url to which request has to be fired  
   'data': {'name': 'Arun', 'age': '30', 'address': 'India'}, // data sent to server  
   'dataType': 'json', // type of response data (html, text, script, )  
   'success': function (data) { // success callback function  
   jQuery('#ajax\_cntr').html(data);  
   },  
   beforeSend: function (data) { // event gets fired get before the ajax request is fired to server  
   jQuery('#ajax\_cntr').html('Loading response...');  
   },  
   error: function (data) {  
   jQuery('#ajax\_cntr').html('Error loading data');  
   },  
   statusCode: {  
   404: function () {  
   alert("Requested page not found");  
   }  
   }  
   });  
   });

**Javascript Ajax**

function loadDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (xhttp.readyState == 4 && xhttp.status == 200) {

document.getElementById("demo").innerHTML = xhttp.responseText;

}

};

// get method

xhttp.open("GET", "ajax\_info.txt", true);

xhttp.send();

//post method

xhttp.open("POST", "ajax\_info.txt", true);

xhttp.send("firstname=Arungopan&lastname=Gopakumar");

**// or using json JSON.stringify({name:”arun”, location:”India”})**

**// xhttp.setRequestHeader("Content-Type", "application/json");**

}

To send a request to a server, we use the open() and send() methods

Onreadystatechange : Stores a function to be called automatically each time the readyState property changes

readyState : value ranges from 0 - 4

0: request not initialized

1: server connection established

2: request received

3: processing request

4: request finished and response is ready

Status : 200 : means Okay

404 : page not found