FULLSTACK JQUERY

JQUERY INTRODUCTION: - jQuery is a lightweight, "write less, do more" JavaScript library.
The purpose of jQuery is to make it much easier to use JavaScript on your website.
jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.
jQuery was originally created by John Resig in 2006.

Pre requisite: -

- HTML
- CSS
- JavaScript

Installation: -

- Goto https://jquery.com/ official site then, click on download and click on download jQuery
- It will open jQuery script save it in local storage using filename with .js extension
- Then create html page and give path of that file to src attribute in script tag.
- Also can copy URL of jQuery path and add it to src attribute of script tag.

Ex: -

```
<!DOCTYPE html>
<html>
   <head>
      <title>  </title>
      <style> </style>
   </head>
   <body>
      <script type="text/javascript" src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
      <script type="text/javascript" src="jquery371.js"></script>
      <script>
         //write code here
      </script>
   </body>
</html>
```

JQUERY SYNTAX: - The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).

Syntax: -

```
jQuery(selector).action()
$(selector).action() // also can use $ instead of jQuery
```

Ex: -

```
jQuery(document).ready(function(){
   document.write("ready")
   console.log("ready")
})
```

Document ready event: - This event prevent any jQuery code from running before the document is finished loading (is ready).
It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows to JavaScript code before the body of your document, in the head section.

Here are some examples of actions that can fail if methods are run before the document is fully loaded:

- Trying to hide an element that is not created yet
- Trying to get the size of an image that is not loaded yet

Ex: - shorter method

```
$(function(){
   document.write("ready")
   console.log("ready")
})
```

JQUERY SELECTORS: - jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors
All selectors in jQuery start with the dollar sign and parentheses: $().

Ex: -

```
<p id="p1">good day</p> <button>hide</button>
```

Js: - jQuery

```
$(document).ready(function(){
   $("button").on("click",function(){
      $("#p1").hide();
   });
});
```

Selectors: - jQuery uses CSS selectors for select elements some of examples

No.	Names	Selectors	Descriptions
1	Element selector	$("p")	The jQuery element selector selects elements based on the element name. it select all elements with provide name
2	Id selector	$("#p1")	The jQuery id selector uses the id attribute of an HTML tag to find the specific element.
3	Class selector	$(".p11")	The jQuery .class selector finds elements with a specific class.
4	All selector	$("*")	Selects all elements

5	This selector	$(this)	Selects the current HTML element
6	Attribute selector	$([type='text'])	Selects attribute with specific values

JQUERY EVENTS: -

All the different visitors' actions that a web page can respond to are called events.
An event represents the precise moment when something happens.

Examples: -

- moving a mouse over an element
- selecting a radio button
- clicking on an element

Common DOM events: -

Mouse Events	Keyboard Events	Form Events	Document/Window Events
Click	Keypress	Submit	Load
Dblclick	Keydown	Change	Resize
Mouseenter	Keyup	Focus	Scroll
Mouseleave		Blur	Unload
Contextmenu			

On method: - this method attaches one or more event handlers for the selected elements.

Syntax: -

```
$(target).on(event, function(){ ... })
```

Ex: -

```
<p id="p1" style="display: none">good day</p> <button >show</button>
```

Js: - jQuery

```
$("#show").on("dblclick",function(){
   $("#p1").show();
})
```

Multiple events for same target: -

Ex: -

```
<p id="p1">good day</p>
```

Js: - jQuery

```
$("p").on({
   mouseenter: function(){
      $(this).css("background-color", "lightgray");
   },
   mouseleave: function(){
      $(this).css("background-color", "transparent");
   },
   click: function(){
        $(this).css("background-color", "yellow");
    }
});
```

Press particular button: - respond on particular button pressed or released.

Keyboard event: - keyboard event interact on body, input type elements, use key or keycode properties for, get key code or value.
Properties: - key, keyCode
Ex: -

```
$("body").on("keydown",function(e){

    console.log(e)
    if(e.key=="a")
    {
        $("p").css("background-color","red")
    }
})
```

Special key combination: - set action when special keys shift, alt, ctrl or Meta (window / cmd) key pressed. Get values of these keys are true or false.
Properties: - altKey, ctrlKey, metaKey, shiftKey
Ex: -

```
$(document).on("keydown",function(e){
     if(e.altKey==true && e.keyCode==71) //key-code of 'g' key
     {
         $("p").css("background-color","pink")
     }
})
```

Mouse key event: - set action when mouse key is pressed or release, with button property as value 0, 1, 2 for respectively left click, middle mouse button and right click.
Property: - button= 0 //0,1,2
Ex: -

```
$("p").on("mousedown",function(e){
    if(e.button==2)
    {
        $("p").css("background-color","aqua")
    }
})
```

JQUERY EFFECTS: - set effect on elements with effect's methods.

JQuery hide/show: - use hide and show methods for hide or show elements.
Methods: - hide(),show()
Ex: -

```
<p> this is paragraph</p>
<button id="hide">hide</button>
<button id="show">show</button>
```
Js: - jQuery
```
$("#hide").click(function(){
   $("p").hide();
});

$("#show").click(function(){
   $("p").show();
});
```

Speed: - use optional parameters for speed
Values: - show, fast, time in milliseconds (1000);
Ex: -
```
$("#hide").click(function(){
   $("p").hide("slow");
});

$("#show").click(function(){
   $("p").show(1000);
});
```

JQuery toggle: - toggle between hiding and showing an element
Ex: -
```
<p style="display: none;">This is paragraph</p>
<button id="toggle">toggle</button>
```
Js: - jQuery
```
$("#toggle").click(function(){
   $("p").toggle();
});
```

Speed: - use optional parameters for speed
Values: - show, fast, time in milliseconds (1000);
Ex: -
```
$("# toggle").click(function(){
   $("p"). toggle ("fast");
});
```

JQuery callback: - A callback function is executed after the current effect is 100% finished. JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors. To prevent this, create a callback function.
Syntax: - $(selector).method (speed, callback);

Ex: - example with callback
```
$("#hide").click(function(){
    $("p").hide(1000,function(){
        alert("hidden")
    });
});
```

Ex: - example without callback

```
$("#hide").click(function(){
    $("p").hide(1000);
    alert("hidden");
});
```

JQuery fade: - fade an element in and out of visibility.

FadeIn: - this method is used to fade in a hidden element. Also can use optional value for speed
Method: - fadeIn("slow"); // slow, fast, milliseconds (500)
Ex: -

```
<div class="box" style="display: none;"></div>
<button id="fadein">fadein</button>
```

Js: - jQuery

```
$("#fadein").click(function(){
    $(".box").fadeIn("slow");
});
```

FadeOut: - this method is used to fade out a visible element. Also can use optional value for speed
Method: - fadeOut(2000); // slow, fast, milliseconds (1000)
Ex: -

```
<div class="box"></div>
<button id="fadeout">fadeout</button>
```

Js: - jQuery

```
$("#fadeout").click(function(){
    $(".box").fadeOut(2000);
});
```

FadeToggle: - this method is used to fade in when element hidden and fade out when it visible. Also can use optional value for speed
Method: - fadeToggle(); // slow, fast, milliseconds (200)
Ex: -

```
<div class="box" style="display: none;"></div>
<button id="fadetoggle">fadetoggle</button>
```

Js: -

```
$("#fadetoggle").click(function(){
    $(".box").fadeToggle();
});
```

FadeTo: - this method allows fading to a given opacity (value between 0 and 1), and also required value for speed.
Method: - fadeTo(speed,opacity); // fadeTo("slow",0.45)

Speed: - slow, fast, milliseconds (1000)
Opacity: - 0 to 1, 0.34, 0.22, 0.89, etc

Ex: -
```
<div class="box"></div>
<button id="fadeto">fadeto</button>
```
Js: - jQuery
```
$("#fadeto").click(function(){
   $(".box").fadeTo("slow",0.15);
});
```

JQuery slide: - create a sliding effect on elements.

SlideDown: - this method is used to slide down an element. Also can use optional value for speed
Method: - slideDown("slow") // slow, fast, milliseconds (600)
Ex: -
```
<button id="slidedown">slidedown</button>
<div id="panel" style="display: none;">...</div>
```
Js: - jQuery
```
$("#slidedown").click(function(){
   $("#panel").slideDown("slow");
});
```

SlideUp: - this method is used to slide up an element. Also can use optional value for speed
Method: - slideUp(900) // slow, fast, milliseconds (100)
Ex: -
```
<button id="slideup">slideup</button>
<div id="panel">...</div>
```
Js: - jQuery
```
$("#slideup").click(function(){
   $("#panel").slideUp(550);
});
```

SlideToggle: - this method is used to slide up and down an element. Also can use optional value for speed
Method: - slideToggle(900) // slow, fast, milliseconds (1200)
Ex: -
```
<button id="slidetoggle">slidetoggle</button>
<div id="panel" style="display: none;">...</div>
```
Js: - jQuery
```
$("#slidetoggle").click(function(){
   $("#panel").slideToggle();
});
```

JQuery animate: - create custom animations.

Animate: - this method is used to create custom animations. It also take speed values

Values: - slow, fast, time in milliseconds (1200)
Method: - $(selector).animate ({params}, speed, callback);
Ex: -
```
<button id="animate">animate</button>
<div id="d1">Good</div>
```
Js: - jQuery
```
$("#animate").click(function(){
   $("#d1").animate({left:'200px'},200)
})
```

Note: - jQuery animate function accepts animation properties value as single numeric only. Property values are treated as a number of pixels unless otherwise specified. The units' em and % can be specified where applicable.

Animate manipulate multiple properties: - set multiple properties and their values
Ex: -
```
$("#animate").click(function(){
   $("#d1").animate({left:'200px',width:'200px',borderRadius:'30px'},1200)
})
```

Animate using relative values: - define relative values using += or -= in front of value.
Ex: -
```
$("#animate").click(function(){
   $("#d1").animate({left:'200px',width:'+=50px',fontSize:'-=5px'},1200)
})
```

Animate using pre-defined values: - specify a property's animation value as "show", "hide", or "toggle"
Ex: -
```
$("#animate").click(function(){
   $("#d1").animate({left:'200px',width:'toggle'},1200)
})
```

Animate using queue functionality: - write multiple animate() calls after each other, jQuery creates an "internal" queue with these method calls. Then it runs the animate calls ONE by ONE.
Ex: -
```
$("#animate").click(function(){
   var div = $("#d1");
   div.animate({height: '300px', opacity: '0.4'}, "slow");
   div.animate({width: '300px', opacity: '0.8'}, "slow");
   div.animate({height: '100px', opacity: '0.4'}, "slow");
   div.animate({width: '100px', opacity: '0.8'}, "slow");
});
```

JQuery stop: - this method works for all jQuery effect functions, including sliding, fading and custom animations.

The optional stopAll parameter specifies whether also the animation queue should be cleared or not. Default is false, which means that only the active animation will be stopped, allowing any queue animations to be performed afterwards.

The optional goToEnd parameter specifies whether or not to complete the current animation immediately. Default is false.

Syntax: - $(selector).stop(stopAll,goToEnd);
Default: - $(selector).stop(false, false);
Ex: -

```
<button id="animate">animate</button>
<button id="stop">stop</button>
<div id="d1">Good</div>
```

Js: - jQuery

```
$("#stop").click(function(){
   $("#d1").stop(true,true);
});
```

JQuery chaining: - this technique allows us to run multiple jQuery commands, one after the other, on the same element(s).
This way, browsers do not have to find the same element(s) more than once. simply append the action to the previous action.

Ex: -

```
$("#chain").click(function(){
   $("#d1").css("color", "white").slideUp(2000).slideDown(2000);
});
```

JQUERY HTML: -

JQuery get and set: - use methods for set and get values of selector elements.
Text method: - it set and returns only text from selected element, pass parameter in method for set text
Method: - $(selector).text(), $(selector).text("text") //set value
Ex: -

```
<div id="d1">good day to all <span>Hello world</span></div>
<button id="good">click button</button>
```

Js: - jQuery

```
$("#good").on("click",function(){
   console.log($("#d1").text())
   $("#good").text("clicked")
})
```

Html method: - it set and returns content of selected elements, including html markups, pass parameter in method for set content including html markups
Method: - $(selector).html(), $(selector). html ("text with html markups") //set value

Ex: -

```
<div id="d1">good day to all <span>Hello world</span></div>
<button id="good">click button</button>
```

Js: - jQuery

```
$("#good").on("click",function(){
   console.log($("#d1").html());
   $("#good").html("<h1>clicked</h1>")
})
```

Val method: - it set or returns values from form fields, pass parameter in method for set value to form fields.

Method: - $(selector).val(), $(selector). val ("values") //set value

Ex: -

```
<input type="text" id="name">
<button id="good">click button</button>
<input type="text" id="show">
```

Js: - jQuery

```
$("#good").on("click",function(){
   let nm= $("#name").val()
   $("#show").val(nm)
})
```

Attr method: - it set or returns value of selected element's attribute, pass attribute name as parameter to see value and pass second parameter for set value to that attribute.

Method: - $(selector).attr("attribute"), $(selector). val("attribute", "value of attribute") //set value

Ex: -

```
<input type="password" id="pass">
<button id="sp">show password</button>
```

Js: - jQuery

```
$("#sp").on("click",function(){
   if($("#pass").attr("type")=="password"){
      $("#pass").attr("type","text")
      $(this).text("hide password")
   }
   else{
      $("#pass").attr("type","password")
      $(this).text("show password")
   }
})
```

Set multiple attributes and values: - use javascript object for set multiple values to attr method using object as properties and value pairs.

Method: - $("selector").attr({"attribute1":"value1", "attribute2":"value2", ... })

Ex: -

```
$("#good").on("click", function(){
   $("#d1").attr({"title":"hello world", "class":"box"})
})
```

JQuery add: - add element and content using jQuery.

Append method: - it inserts content at end of elements.
Method: - $(selectors).append(content)
Ex: -

```
$("p").append("Good day")
```

Prepend method: - it inserts content at beginning of elements.
Method: - $(selectors). prepend(content)
Ex: -

```
$("p").prepend("Good day")
```

Append or prepend several elements: -
Ex: -

```
var txt1 = "<p>Text.</p>";              // Create element with HTML
var txt2 = $("<p></p>").text("Text.");   // Create with jQuery
var txt3 = document.createElement("p");  // Create with DOM
txt3.innerHTML = "Text.";
$("body").append(txt1, txt2, txt3);      // Append the new elements
```

After method: - it inserts content after selected elements.
Method: - $(selector).after(content);
Ex: -

```
$("img").after("Some text after");
```

Before method: - it inserts content before selected elements.
Method: - $(selector).before(content);
Ex: -

```
$("img").before("Some text before");
```

Add several new elements with after and before methods: -
Ex: -

```
var txt1 = "<b>I </b>";                  // Create element with HTML
var txt2 = $("<i></i>").text("love ");     // Create with jQuery
var txt3 = document.createElement("b");    // Create with DOM
txt3.innerHTML = "jQuery!";
$("img").after(txt1, txt2, txt3);          // Insert new elements after <img>
```

JQuery remove: - remove content and elements.

Remove method: - it removes selected elements and its children
Method: - $(selector).remove();
Ex: -

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

Empty method: - it removes content and children of selected elements
Method: - $(selector).empty();
Ex: -

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

Filter elements to remove: - remove method also accept parameter, which allows filtering elements to remove
Method: - $(selector).remove(parameter for filters);
Ex: -
$("p").remove(".test"); //remove p elements with test class.
$("p").remove(".test, .demo"); //remove elements with demo and test class.

JQuery CSS classes: - use function for manipulate classes with jQuery
AddClass method: - adds one or multiple classes to selected elements.
Method: - $("selector").addClass("classes");
Ex: -
$("#d1").addClass("box");
$("#d1").addClass("box danger"); //add multiple classes
$("#d1, h1").addClass("danger"); //add class to multiple selected elements
Note: - define multiple selectors using comma between to selectors.

RemoveClass method: - remove specific class from selected elements.
Method: - $("selectors").removeClass("classes");
Ex: -
$("#d1, h1").removeClass("danger");

ToggleClass method: - this method toggle between add and remove class.
Method: - $("selectors").toggleClass("classes");
Ex: -
$("#d1").toggleClass("danger");

JQuery CSS method: - this method sets or returns one or more style properties for the selected elements.
Return CSS property: - return value of specified CSS property
Method: - $("selector").css("property name")
Ex: -
let prop1=$("#d1").css("background-color")
console.log(prop1)

Set CSS property: - set specified value to CSS property
Method: - $("selector").css("property name","value")
Ex: -
$("#d1").css("background-color", "red")

Set multiple properties: - set multiple properties and values to selected element using object.
Method: - $("#d1").css({"property-name1":"value1","property-name2":"value2",...})
Ex: -
$("#d1").css({"background-color":"red","color":"white","font-size":"30px"})

JQuery dimensions: - these methods work with dimensions.

Width and height method: - this methods sets or returns height and width of element.
Method: - $("selector").width("value"), $("selector").height("value")
Ex: -

```
let w=$("#d1").width("200px");
let h=$("#d1").height("200px");
console.log("width:- ",w);
console.log("height:- ",h);
```

InnerWidth and innerHeight method: - this methods returns height and width of element, including padding.
Method: - var1=$("selector").innerWidth(), var2=$("selector").innerHeight()
Ex: -

```
let wi=$("#d2").innerWidth();
let hi=$("#d2").innerHeight();
console.log("inner width:- ",wi);
console.log("inner height:- ",hi);
```

OuterWidth and outerHeight method: - this methods returns height and width of element, including padding, border and margin.
Method: - var1=$("selector").outerWidth(), var2=$("selector").outerHeight()
Ex: -

```
let wo=$("#d2").outerWidth();
let ho=$("#d2").outerHeight();
console.log("outer width:- ",wo);
console.log("outer height:- ",ho);
```

JQUERY TRAVERSING: -

What is traversing: - jQuery traversing, which means "move through", are used to "find" (or select) HTML elements based on their relation to other elements. Start with one selection and move through that selection until reach to the desired element.

Traversing the DOM: - jQuery provides a variety of methods that allows to traverse the DOM, The largest category of traversal methods is tree-traversal.

JQuery ancestors: - An ancestor of element is a parent, grandparent, great-grandparent, and so on
Parent method: - this method returns the direct parent element of the selected element.
Method: - $("selector").parent()
Ex: -

```
let a=$(".box1").parent()
console.log(a)
```

Parents method: - this method returns all ancestor elements of the selected element, all the way up to the document's root element (<html>).
Method: - $("selector").parents()

Ex: -
```
let b=$(".box1").parents()
console.log(b)
```

Filter search for ancestor: - use optional parameter for target ancestor of elements
Ex: -
```
$("#b1").on("click",function(){
   $(this).parents(".container").css("background-color","red");
})
```

ParentUntil method: - this method returns all ancestor elements between two given arguments.
Method: - $("selector").parentsUntil("selector")
Ex: -
```
let c=$("#b1").parentsUntil("body")
console.log(c)
```

JQuery descendants: - A descendant is a child, grandchild, great-grandchild, and so on.

Children method: - this method returns all direct children of the selected element. This method only traverses a single level down the DOM tree.
Method: - $("selector").children()
Ex: -
```
let d=$("body").children()
console.log(d)
```

Filter search for children: - return children with provided selector only.
Ex: -
```
let d=$("body").children("script")
console.log(d)
```

Find method: - this method returns descendant elements of the selected element, all the way down to the last descendant.
Method: - $("selector").find("selector")
Ex: -
```
let e=$("body").find("div")
console.log(e)
```

Return all descendants: - use "*" selector for return all descendant elements
Ex: -
```
let f=$("body").find("*")
console.log(f)
```

JQuery siblings: - With jQuery traverse sideways in the DOM tree to find siblings of an element.

Siblings method: - this method returns all sibling elements of the selected element.
Method: - $("selector"). siblings()
Ex: -
```
let g=$(".box1").siblings()
console.log(g)
```

Next method: - this method returns the next sibling element of the selected element.
Method: - $("selector").next()
Ex: -
```
let h=$(".box1").next()
console.log(h)
```

NextAll method: - this method returns all next sibling elements of the selected element.
Method: - $("selector").nextAll()
Ex: -
```
let i=$(".box1").nextAll()
console.log(i)
```

NextUntil method: - this method returns all next sibling elements between two given arguments.
Method: - $("selector").nextUntil("selector")
Ex: -
```
let j=$(".box1").nextUntil("#b2")
console.log(j)
```

Prev method: - this method returns the previous sibling element of the selected element.
Method: - $("selector").prev()
Ex: -
```
let k=$(".box1").prev()
console.log(k)
```

PrevAll method: - this method returns all previous sibling elements of the selected element.
Method: - $("selector").prevAll()
Ex: -
```
let l=$(".box1").prevAll()
console.log(l)
```

PrevUntil method: - this method returns all previous sibling elements between two given arguments.
Method: - $("selector"). prevUntil("selector")
Ex: -
```
let m=$(".box1").prevUntil("h2")
console.log(m)
```

JQuery filtering: - selects elements by filtering them.

First method: - this method returns the first element of the specified elements.
Method: - $("selector").first()
Ex: -
```
let n=$("div").first()
console.log(n)
```

Last method: - this method returns the last element of the specified elements.
Method: - $("selector").last()

Ex: -
let n=$("div").last()
console.log(n)

Eq method: - this method returns an element with a specific index number of the selected elements.
Note: - The index numbers start at 0, so the first element will have the index number 0 and not 1.
Method: - $("selector ").eq(index)
Ex: -
let p=$("div").eq(1)
console.log(o)

Filter method: - this method specifies criteria, elements that do not match the criteria are removed from the selection and those that match will be returned.
Method: - $("selector").filter("selector")
Ex: -
let q=$("div").filter(".container")
console.log(q)

Not method: - this method returns all elements that do not match the criteria.
Method: - $("selector").not("selector")
Ex: -
let r=$("div").not(".container")
console.log(r)

JQERY AJAX: -

JQuery AJAX intro: -

What is AJAX: - it is Asynchronous JavaScript and XML. AJAX is about loading data in the background and displays it on the webpage, without reloading the whole page.
Examples of applications using AJAX: Gmail, Google Maps, Youtube, and Facebook tabs.

JQuery and AJAX: - With the jQuery AJAX methods, it can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post - And also can load the external data directly into the selected HTML elements of web page!

JQuery Load method: - The jQuery load() method is a simple, but powerful AJAX method. The load() method loads data from a server and puts the returned data into the selected element.

method: - $(selector).load(URL data,callback);
Ex: - $("#d1").load("good1.html")

Load particular element: - load particular element from file using selector
Ex: - $("#d1").load("good1.html #d1")

Load callback function: - also can pass callback function, it can take three arguments

- responseTxt - contains the resulting content if the call succeeds

- statusTxt - contains the status of the call, values are ‘success’ or ‘error’
- xhr - contains the XMLHttpRequest object

Ex: -

```
$("button").on("click",function(){
   $("#d1").load("good1.html",function(response,status,xhr){
      console.log(response)
      console.log(status)
      console.log(xhr)

      if(status=="error"){
         $("#d1").html(response)
         setTimeout(function(){
            $("#d1").empty()
         },2000)
      }
   })
})
```

JQuery Get method: - this method requests data from the server with an HTTP GET request.
Method: - $.get(url, callback function);
Ex: -

```
$("#b1").on("click",function(){
   $.get("good.txt",function(data,status){
      console.log(data)
      console.log(status)
   })
})
```

JQuery Post method: - this method send data from the server with an HTTP POST request.
Method: - $.post(url, data, callback function);
Data: - this is optional parameter for send data along with request, this data is send as object form.
Ex: -

```
$("#b1").on("click",function(){
   $.post("https://jsonplaceholder.typicode.com/posts",{age:12,name:"madhav"},
function(data,status){
      console.log(data)
      console.log(status)
   })
})
```

Note: - post method will only work if that allowed on server

Check data: - see data that send along with request, for inspect page, goto network tab, there will be file or url name click on that name, in playload section data will be show that send along with request

JQuery AJAX method: - it performs an asynchronous HTTP (Ajax) request.
Syntax: - $.ajax(url(string),setting(object));
Setting: - a set of key/value pairs that configure Ajax request, all settings are optional.
Ex: -
```
$("#b1").on("click",function(){
   $.ajax("good.txt", {success: function(result){alert(result)}});
})
```

Note: - Due to browser security restrictions, most "Ajax" requests are subject to the 'same origin policy'; the request cannot successfully retrieve data from a different domain, sub-domain, port, or protocol.
* Script and JSONP requests are not subject to the same origin policy restrictions.
The same-origin policy: - it is a browser security feature that restricts how documents and scripts on one origin can interact with resources on another origin.

Ajax settings: -
Success: - a function to be called if request succeeds, this function get three arguments as data, status and xhr (XMLHttpRequest) object.
Ex: -
```
$.ajax("good.txt", {
    success: function(data,status,xhr){
        console.log(data)
        console.log(status)
        console.log(xhr)
    }
});
```

Cache: - it take data from browser cache memory, it's default value is true, if any changes get in requested data in source file at server, it will not show it , If set to false, it will force requested pages not to be cached by the browser.
Values: - true(default), false
Ex: -
```
$.ajax("good.txt", {
    success: function(data){console.log(data)},
    cache: false
});
```

Async: - by default all request sent asynchronously, set it as false for send synchronous (sequential) request.
Values: - true, false
Ex: -
```
$("#b1").on("click",function(){
    $.ajax("good.txt", {
        success: function(data){console.log(data)},
        async: false
    });
    console.log("Function completed")
})
```

Error: - it called function if request fails, this function get three arguments as xhr (XMLHttpRequest) object, status as 'error' or 'timeout' and errorThrown that receive textual portion of the HTTP status as 'Not found', 'timeout' or 'Internal server error' etc.

Ex: -

```
$.ajax("good1.txt", {
    success: function(data){console.log(data)},
    cache: false,
    error: function(xhr,status,errorThrown){
        console.log(xhr)
        console.log(status)
        console.log(errorThrown)
    }
});
```

Complete: - it called function when request finished, get passed two arguments as xhr and status

Ex: -

```
$.ajax("good1.txt", {
    cache: false,
    complete: function(xhr,status){
        console.log(xhr)
        console.log(status)
    }
});
```

Timeout: - set timeout in milliseconds for Ajax request, 0 value means there will be no timeout. The timeout period starts at the point the $.ajax() call is made; if several other requests are in progress and the browser has no connections available, it is possible for a request to time out before it can be sent.

Values: - 0, 100, 300, 40 etc

Ex: -

```
$.ajax("good1.txt", {
    success: function(data){console.log(data)},
    cache: false,
    async: true,
    timeout: 30,
    error: function(xhr,status,errorThrown){
        console.log(xhr)
        console.log(status)
        console.log(errorThrown)
    }
});
```

url: - A string containing the URL to which the request is sent. Default value is current page.

Ex: -

```
$.ajax({
    success: function(result){console.log(result)},
    cache: false,
    url: "good.txt",
});
```

Datatype: - set type of data that are expecting back from the server. If none is specified, jQuery will try to infer it based on the MIME type of the response
Values: - xml, html, script, json, text
Ex: - load and execute javascript using ajax

```
$.ajax({
    url:"good2.js",
    dataType:"script",
    cache:false
});
```

Data: - Data to be sent to the server. If the HTTP method is one that cannot have an entity body, such as GET, the data is appended to the URL.
Values: - string,objects
Ex: -

```
$.ajax({
        url: " https://jsonplaceholder.typicode.com/posts",
        data:{
            name: $("#nm").val(),
            age: $("#ag").val()
        },
        success: function(r,s){(s);}
})
```

Method: - The HTTP method to use for the request
Values: - GET, POST
Ex: -

```
$.ajax({
    url: " https://jsonplaceholder.typicode.com/posts",
    method: "POST",
    data:{
        name: $("#nm").val(),
        age: $("#ag").val()
    },
    success: function(response,s){
        console.log(response);
        alert(s);
    }
});
```

ASP.NET USER: - in case of error http 404.3 not found
In web.config file set mimeMap with file extension and mimeType
Ex: -

```
<configuration>
   <system.webServer>
     <staticContent>
       <mimeMap fileExtension=".json" mimeType="text/json"/>
     </staticContent>
   </system.webServer>
</configuration>
```

AJAX JQUERY promise methods: - these methods call after request completed. These method work with jqXHR object that return from ajax method.
Done method: - this method is alternative to success callback option
Syntax: - $.ajax(…).done(function)
Ex: -

```
$.ajax("https://jsonplaceholder.typicode.com/posts").done(
    function(data,status,jqXHR){
        console.log(data)
        console.log(status)
        console.log(jqXHR)
    }
)
```

Fail method: - this method is alternative to error callback option
Syntax: - $.ajax(…).fail(function)
Ex: -

```
$.ajax("https://jsonplaceholder.typicode.com/posts").fail(
    function(jqXHR,status,errorThrown){
        console.log(jqXHR)
        console.log(status)
        console.log(errorThrown)
    }
)
```

Always method: - this method is alternative to complete callback option, if request get successful, then function arguments are same as done() method, if request failed then arguments are same as fail() method.
Syntax: - $.ajax(…).always(function)
Ex: -

```
$.ajax("https://jsonplaceholder.typicode.com/posts").always(
    function(data,status,xhr){
        if(status=="success"){
            console.log("Data:- ",data)
            console.log("Status:- ",status)
            console.log("xhr:- ",xhr)
        }
        else{
            console.log("xhr:- ",data)
            console.log("Status:- ",status)
            console.log("error:- ",xhr)
        }
    }
)
```

Then method: - Incorporates the functionality of the .done() and .fail() methods, allowing the underlying Promise to be manipulated
Syntax: - $.ajax(…).then(function //done, function //fail)
Assign handler after making request: -
Ex: -

```
var request=$.ajax("https://jsonplaceholder1.typicode.com/posts/")
request.then(function(data,status,jqXHR){
        console.log("1")
        console.log(data)
        console.log(status)
        console.log(jqXHR)
    },
    function(jqXHR,status,errorThrown){
        console.log("2")
        console.log(jqXHR)
        console.log(status)
        console.log(errorThrown)
})
```

JSON INTRODUCTION: - JSON stands for JavaScript Object Notation, JSON is a text format for storing and transporting data, JSON is "self-describing" and easy to understand

JSON syntax: -

- Data is in key/value pairs, keys must be strings, written with double quotes
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

JSON datatypes: -

In JSON, values must be one of the following data types:

- a string
- a number
- an object
- an array
- a boolean
- null

Ex: -

```
{
   "string":"hello",
   "number":3.4,
   "object":{"key1":"value1", "key2":2},
   "array":[1,2,3,4],
   "boolean":"true",
   "null":null
}
```

JSON Objects: - JSON data is stored into objects with key and value pairs and key must be string in double quotes.

Syntax: -

```
{
   "key1":"value1",
   "key2":"value2",
   "key3":"value3",
   ...
}
```

Ex: -

```
{
   "name":"ram",
   "age":24,
   "salary":345.67
}
```

JSON array of objects: - set array of multiple JSON objects

Syntax: -

```
[
   {Object1},
   {Object2},
   ...
   {ObjectN}
]
```

Ex: -

```
[
   {
      "name":"ram",
      "age":24,
      "salary":345.67
   },
   {
      "name":"krishna",
      "age":20,
      "salary":735.17
   }
]
```

JQUERY JSON method: - this method is used to get encoded JSON data using an AJAX HTTP GET request.

Syntax: -

```
$.getJSON(url, data, callback function);
```

Load callback function: - also can pass callback function, it can take three arguments

- responseTxt - contains the resulting content if the call succeeds
- statusTxt - contains the status of the call, values are ‘success’ or ‘error’
- xhr - contains the XMLHttpRequest object

Ex: -

```
$("#b1").on("click",function(){
   $.getJSON("test.json",{name:"ram",age:34},function(data,status,xhr){
      console.log(data)
      console.log(status)
      console.log(xhr)
   })
})
```

JQUERY MISC: -

JQUERY noConflict method: - jQuery uses the $ sign as a shortcut for jQuery. There are many other popular JavaScript frameworks like: Angular, Backbone, Ember, Knockout, and more. There is posibility of that other javascript framework can also use $ shortcut, If two different frameworks are using the same shortcut, one of them might stop working. The jQuery team have already thought about this, and implemented the noConflict() method. The noConflict() method releases the hold on the $ shortcut identifier, so that other scripts can use it.

Ex: - use jQuery after noConflict method instead of $ sign.

```
$.noConflict();
jQuery(document).ready(function(){
   jQuery("button").on("click",function(){
      jQuery("p").text("jQuery is still working!");
   });
});
```

Ex: - create own shortcut using noConflict method, this method returns reference to jQuery, that can save in variable

```
var $$=$.noConflict();
$$(document).ready(function(){
   $$("button").on("click",function(){
      $$("p").text("jQuery is still working!");
   });
});
```

Ex: - use $ sign inside document ready method, pass $ sign to function as parameter inside ready method

```
$.noConflict();
jQuery(document).ready(function($){
   $("button").click(function(){
      $("p").text("jQuery is still working!");
   });
});
```

Filter method: - Reduce the set of matched elements to those that match the selector or pass the function's test.

Syntax: - $("selector").filter(selector), $("selector").filter(function)

Ex: - design perticular element using filter

```
$("p").filter(".good").css("border","1px solid black")
```

Ex: - using function that take index argument for return filtered index

```
$( "p" ).filter(function(i){
   return i%2==0;
}).css("background-color", "red")
.css("color", "white")
```

Filter table result: -

Html: -

```
<input id="search" type="search" placeholder="search">
<table border="1px">
   <thead>
      <tr>
         <th>id</th>
         <th>name</th>
         <th>marks</th>
      </tr>
   </thead>
   <tbody id="data">
      <tr>
         <td>1</td>
         <td>ram</td>
         <td>34</td>
      </tr>
      <tr>
         <td>2</td>
         <td>shyam</td>
         <td>67</td>
      </tr>
      <tr>
         <td>3</td>
         <td>madhav</td>
```

```
            <td>78</td>
        </tr>
    </tbody>
</table>
```

Script: -

```
$(document).ready(function(){
$("#search").on("keyup", function() {
    var value = $(this).val().toLowerCase();
        $("#data tr").filter(function() {
            $(this).toggle($(this).text().toLowerCase().indexOf(value) > -1)
        });
    });
});
```

Each method: - Execute a function for each matched element, it function take index and element as argument

Syntax: - $("selector").each(function)

Ex: -

```
$(document).ready(function(){
    $("p").each(function(i,element){
        let t=$(element).text()
        $(element).html("<b>"+t)
    })
});
```

Load data in table and add new data in table: -

Html: -

```
<input type="number" placeholder="userid" id="user"><br>
<input type="text" placeholder="title" id="title"><br>
<textarea placeholder="body" id="body"></textarea><br>
<button id="b1" type="submit">Show</button>
<table id="t1" border="1px">
    <thead>
        <tr>
            <th>ID</th>
            <th>userId</th>
            <th>title</th>
            <th>body</th>
        </tr>
    </thead>
</table>
```

Script: -

```
$(document).ready(function () {
    $("button").on("click",function(){
        $.ajax({
            method: "POST",
            url: "https://jsonplaceholder.typicode.com/posts",
            success: function (result) {
                  $(result).each(function (i, t) {
                  $("#t1").append($("<tr>")
                     .append($("<td>").append(t.id))
                     .append($("<td>").append(t.userId))
                     .append($("<td>").append(t.title))
                     .append($("<td>").append(t.body))
                  )
               })
            },
            cache: false,
         })
      })

      $.ajax({
         method: "GET",
         url: "https://jsonplaceholder.typicode.com/posts",
         success: function (result) {
            $(result).each(function (i, t) {
               $("#t1").append($("<tr>")
                  .append($("<td>").append(t.id))
                  .append($("<td>").append(t.userId))
                  .append($("<td>").append(t.title))
                  .append($("<td>").append(t.body))
               )
            })
         },
         cache: false,
      })
   });
```

Data method: - Store arbitrary (unrelative) data associated with the matched elements, or return that data associated with the first element in the jQuery collection, as set by data() or by an HTML5 data-* attribute.
Syntax: - $(selector).data(key,value), $(selector).data(obj) // store data
$(selector).data(key), $(selector).data() //return data

Ex: - return data stored in html data attribute
Html code: -

```
<p data-status="good" data-grade="bad">good day 1</p>
```

Script: -

```
$(document).ready(function(){
   let data=$("p").data()
   console.log(data)
});
```

Ex: - store data in element
Script: -
```
$(document).ready(function(){
   $("p").data("greet1","good day") //key value
   $("p").data({"greet2":"good night","greet3":"hello world"}) //object
   let data=$("p").data()
   console.log(data)
});
```

Remove-Data method: - remove data values from element, if its stored by data() method
Syntax: - $(selector).removeData(key), $(selector).removeData([array of keys])
Ex: -
Script: -
```
$(document).ready(function(){
   $("p").data("greet1","good day")
   $("p").data({"greet2":"good night","greet3":"hello world"})
   $("p").removeData(["greet1","greet2"])
   $("p").removeData("greet3")
   let data=$("p").data()
   console.log(data)
});
```

Index method: - Search for a given element from among the matched elements.
Syntax: - $(selector).index(),$(selector).index(selector)
Ex: -
```
let ele=$("p").index($(".good"))
console.log(ele)
```

Ex: - get index on click
```
$(document).ready(function(){
   $("p").on("click",function(){
      let arr=$("p").index(this)
      console.log(arr)
   })
})
```

Ex: - stored and remove data on button click
Html: -
```
<div>A div</div>
<button class="in">Get "state" from the div</button>
<button class="in">Set "state" to "hello"</button>
<button class="in">Set "state" to 86</button>
<button class="in">Remove "state" from the div</button>
<p>The "state" value of this div is </p>
<span> ?</span>
```

Script: -

```
$(document).ready(function(){
   $( "button.in" ).on( "click", function() {
      var value;
      switch ( $( "button.in" ).index( this ) ) {
         case 0 :
            value = $( "div" ).data( "state" );
            break;
         case 1 :
            $( "div" ).data( "state", "hello" );
            value = "Stored! string";
            break;
         case 2 :
            $( "div" ).data( "state", 86 );
            value = "Stored! number";
            break;
         case 3 :
            $( "div" ).removeData( "state" );
            value = "Removed!";
            break;
      }
      $("span").text(""+value);
   });
})
```

Get method: - Retrieve one of the elements matched by the jQuery object.
Syntax: - $(selector).get(index), $(selector).get()
Ex: -
Html: -

```
<ul>
   <li>good 1</li>
   <li>good 2</li>
   <li>good 3</li>
   <li>good 4</li>
   <li>good 5</li>
</ul>
```

Script: -

```
$(document).ready(function(){
   console.log($("li").get(1))
   console.log($("li").get())
})
```

Length property: - return length of total elements matched by the jQuery object.
Syntax: - $(selector).length
Ex: - console.log($("li").length)

SECURITY AND AUTHORIZATION USING AJAX: -

Param method: - create a serialized representation of an array, a plain object
Syntax: - let data=$.param(obj/arr)
Ex: -

```
var data = {
   userId: 11,
   title: "good",
   body: "good day 1"
};
var recursiveEncoded = $.param( data );
console.log( "Encoded:- ",recursiveEncoded );

var recursiveDecoded = decodeURIComponent(recursiveEncoded);
console.log( "Decoded:- ",recursiveDecoded );
```

Note: - use decodeURIComponent() method for decode serialized param

encodeURIComponent method: - it encodes URI(uniform resource identifiers) by replacing each instance of certain characters.
Syntax: - encodeURIComponent(string/array)
Ex: -

```
a="ram@1234gmail.com"
console.log("  data:- ",a)
en=encodeURIComponent(a) // encode data
console.log("encode:- ",en)
de=decodeURIComponent(en) // decode data
console.log("decode:- ",de)
```

Note: - use decodeURIComponent() method for decode encoded data

Btoa and atob methods: - use btoa method encode binary string to ASCII base-64 representation, and atob method decode base-64 encoded string by btoa method
Ex: -

```
let name="ram"
let encode=btoa(name) //encode string
let decode=atob(encode)  //decode string
console.log("name:- ",name)
console.log("encode:- ",encode)
console.log("decode:- ",decode)
```

Basic authorization: - use headers in AJAX setting object with headers pair with object of authorization key and value pair, for set basic authorization
Ex: -

```
$.ajax({
    …
    headers:{
        'authorization':'Basic '+btoa(user+":"+pass)
    }
    ...
}
```

Ex: - create client-side authorization

```
$("#btn").on("click", function(){
    let user=$("input[type='text']").val()
    let pass=$("input[type='password']").val()
    $.ajax({
        url:"https://jsonplaceholder.typicode.com/posts",
        type: 'GET',
        headers:{
            'authorization':'Basic '+btoa(user+":"+pass)
        },
        success:function(data,s){
            console.log(data)
            console.log(s)
        },
        error:function(s,e){
            console.log(s)
            console.log(e)
        }
    })
})
```

Security notes: -

- **ready method: -** write code in $(document).ready() function, for run jQuery script when document is loaded, use anonymous function in that method
- **use text method: -** use text() method instead of html() method for set data in html
- **Don't rely on client logic for security: -** Don't forget that the user controls the client-side logic. A number of browser plugins are available to set breakpoints, skip code, change values, etc. Never rely on client logic for security.
- **Don't rely on client business logic: -** Just like the security one, make sure any interesting business rules/logic is duplicated on the server side lest a user bypasses needed logic and does something silly, or worse, costly.
- **Never transmit secrets to the client: -** Anything the client knows the user will also know, so keep all that secret stuff on the server please.