jQuery

jQuery is a lightweight, "write less, do more", JavaScript library. It is simple and easy to learn. To perform any task JQuery required less code compare to javascript.

jQuery is a fast, small, and feature-rich JavaScript library.

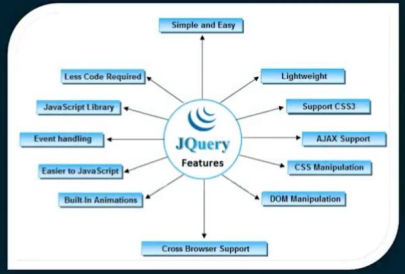
It makes things like HTML document traversal and manipulation, event handling, animation, DOM Traversal and Manipulation and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

JQuery Feature and Explanation.

- **Simple and easy**: It have predefined method using you can perform any task easily compare to JavaScript. And it is easy to learn.
- **Lightweight:** It is very lightweight library - about 19KB in size (Minified and gzipped).
- **CSS manipulation**: It have predefined css() method for manipulate style for any Html elements.
- **HTML manipulation**: The jQuery made it easy to select DOM elements, traverse them and modifying their content.
- **DOM Traversal and manipulation**:
- **Cross browser support**: It support all modern web-browser including IE-6.
- **Event handling**: It support event handling like click mouse button.
- **JavaScript Library**: It is JavaScript library.

JQuery features



Add JQuery

There are several ways to add jQuery on your web-page. You can download jquery from jquery.com or Include jQuery from a CDN. Below two methods are used for add jQuery;

- Download from www.jquery.com
- Include jQuery from a CDN (Content Delivery Network).

Downloading JQuery

In this method you download jquery library and host on your website. Jquery library is a single JavaScript file, you need to give path of jquery file on your website;

Add jquery on web-page

```
<head>
<script src="jquery-1.11.3.min.js"></script>
</head>
```

jQuery CDN

Another way to add jquery on your web-page is; direct use hosted jquery library files, both Google and Microsoft host jQuery library. Below syntax is used for add jquery library on your website;

Google CDN

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js">
</script>
</head>
```

Microsoft CDN

```
<head>
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.3.min.js">
</script>
</head
```

Which method is better?

Many people downloaded jQuery hosted files from google and Microsoft when they visit other websites; as a result when they visit your website, jQuery files loaded from cache. Due to this website load faster. So use google or Microsoft hosted files is better than host jQuery file on your own website.

What is a CDN?

A Content Delivery Network or Content Distribution Network (CDN) is a large distributed system of servers deployed in multiple data centers across the Internet.

Only for remember: Here lot of files are stored, using CDN you can directly use these files on your website. No need to upload these files on your server.

What is the advantage of using CDN?

When you use CDN get following advantage.

- It reduces the load from your server.
- It saves bandwidth.
- When any user visit on other website it save in browser, as a result next time easily load.

Editors for jQuery

Before starting with its necessary to know what are the rich text editors where you can write your jQuery code. Below are some of the common editors, you can use to write jQuery code.

- Notepad
- Notepad++
- Visual Studio
- Visual Studio Code
- CKEditor
- TinyMCE
- Summernote
- Trumbowyg
- Simditor

Jquery Syntax

With the help of jQuery you can select Html elements and perform actions on them.

Syntax

$(selector).action()

- $: This sign define jQuery
- (selector) : It select or find Html elements
- action() : It performed action on the html elements

Example

- $(this).hide() : hides the current element.
- $("p").hide() : hides all <p> elements.
- $(".list").hide() : hides all elements with class="list".
- $("#list").hide() : hides the element with id="list".

What is dollar sign ($) in jquery ?

Dollar Sign is nothing but it's an alias for JQuery. You can write jQuery at the place of $.

Syntax

```
$(document).ready(function(){
});
(or)
jQuery(document).ready(function(){
});
```

Document Ready Event

You can notice in any Jquery example or code all jQuery methods are inside a document ready event. This event prevent any jQuery code from running before the document is finished loading, It means jquery code load after complete loading of web page.

Why need Document Ready Event

Document Ready Event prevent any jQuery code from running before the document is finished loading. Suppose if your jquery code load before loading the image in this case you can't get the size of images. In below condition action are failed when method run before the document is fully loaded.

- Try to get size of any image which is not loaded yet.
- Try to hide element which is not loaded yet.

Syntax

```
$(document).ready(function()
{

  // jQuery methods

});
```

Downloading jQuery

There are two versions of jQuery available for downloading:

- Production version - this is for your live website because it has been minified and compressed
- Development version - this is for testing and development (uncompressed and readable code)

Both versions can be downloaded from jQuery.com.

How to start with jQuery

You can put any number of scripts inside an HTML document, or else you can write all your jQuery code in an external file, save it with .js file extension and call that file inside our HTML Code.

It is recommended not to write all jQuery code inside the HTML file as the file. In a HTML file jQuery code are written inside <script> and </script> tags.

Jquery Website

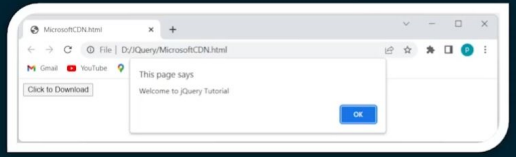
https://releases.jquery.com/

Program(GoogleCDN.html)

```
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script>
    $(document).ready(function()
    {
      $("#btnDemo").click(function()
      {
        alert("Welcome to jQuery Tutorial");
      });
    });
 </script>
</head>
<body>
  <input type = "button" id= "btnDemo" value="Click to Download"/>
</body>
</html>
```

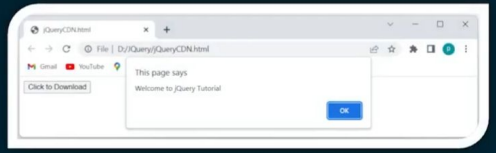
Program(MicrosoftCDN.html)

```
<html>
<head>
  <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.3.min.js"></script>
  <script>
    $(document).ready(function()
    {
      $("#btnDemo").click(function()
      {
        alert("Welcome to jQuery Tutorial");
      });
    });
  </script>
</head>
<body>
  <input type = "button" id= "btnDemo" value="Click to Download"/>
</body>
</html>
```



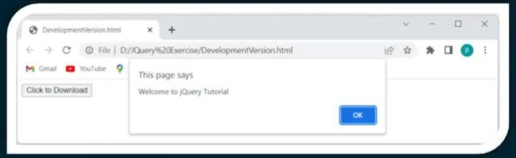
Program(jQueryCDN.html)

```
<html>
<head>
 <script src="https://code.jquery.com/jquery-1.11.3.js"></script>
 <script>
   $(document).ready(function()
   {
     $("#btnDemo").click(function()
     {
       alert("Welcome to jQuery Tutorial");
     });
   });
   /*
       (or)
      $(function()
      {
         $("#btnDemo").click(function()
         {
            alert("Welcome to jQuery Tutorial");
         });
      });
   */
 </script>
</head>
<body>
  <input type = "button" id= "btnDemo" value="Click to Download"/>
</body>
</html>
```



Program(DevelopmentVersion.html)

```
<html>
<head>
  <script src="jQuery/jquery-1.11.3.js"></script>
  <script>
    $(document).ready(function()
    {
      $("#btnDemo").click(function()
      {
        alert("Welcome to jQuery Tutorial");
      });
    });
 </script>
</head>
<body>
  <input type = "button" id= "btnDemo" value="Click to Download"/>
</body>
</html>
```



The element Selector

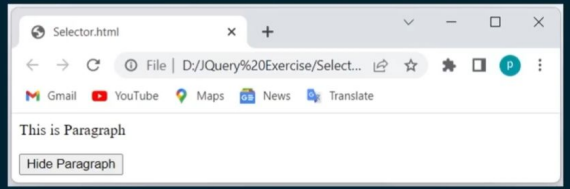
The jQuery element selector selects elements based on the element name.

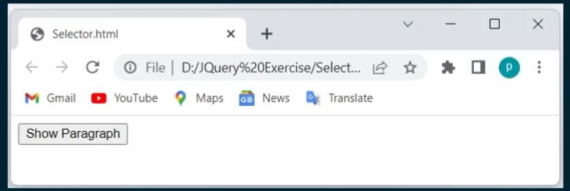
You can select all <p> elements on a page like this: $("p")

The example below to hide and show a paragraph when button click

Example(Selector.html)

```
<html>
<head>
 <script src="jQuery/jquery-1.11.3.js"></script>
 <script>
  $(function()
  {
    $("#btn").click(function()
    {
     if($("#btn").val()=='Hide Paragraph')
     {
      $("p").hide();
      $("#btn").val('Show Paragraph');
     }
     else
     {
      $("p").show();
      $("#btn").val('Hide Paragraph');
     }
    });
  });
 </script>
</head>
<body>
 <p>This is Paragraph</p>
 <input type = "button" id= "btn" value="Hide Paragraph"/>
```

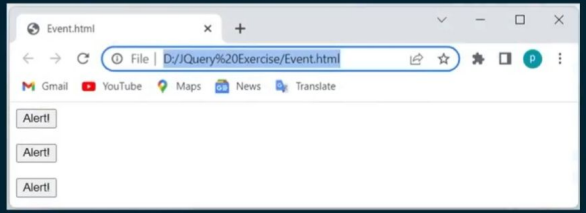




HTML Events

Program(Event.html)

```
<html>
<head>
 <script src="jQuery/jquery.js"></script>
 <script>
   $(function()
   {
      $("button.alert").on("click", function()
      {
        console.log("A button with the alert class was clicked!");
        $("<br> <br><button class='alert'>Alert!</button>").appendTo(document.body);
      });

   });
 </script>
</head>
<body>
 <button class='alert'>Alert!</button>
</body>
</html>
```



DOM Manipulation Methods in jQuery

jQuery provides various methods to add, edit or delete DOM element(s) in the HTML page.

The following table lists some important methods to add/remove new DOM elements.

Method	Description
append()	Inserts content to the end of element(s) which is specified by a selector.
before()	Inserts content (new or existing DOM elements) before an element(s) which is specified by a selector.
after()	Inserts content (new or existing DOM elements) after an element(s) which is specified by a selector.
prepend()	Insert content at the beginning of an element(s) specified by a selector.
remove()	Removes element(s) from DOM which is specified by selector.
replaceAll()	Replace target element(s) with specified element.
wrap()	Wrap an HTML structure around each element which is specified by selector.

Program

```
<html>
<head>
  <title>DOM Manipulation Methods</title>
  <script src="jQuery/jquery-1.11.3.js"></script>
  <script>
    $(function()
    {
        $("div").on("click", function()
        {
            $("div").prepend('<p>This is prepened Paragraph</p>');
            $("div").before('<p>Paragraph added First(before)</p>');
            $("div").after('<p>Paragraph added Last(after)</p>');
            $("div").append('<p>This is appended Paragraph</p>');
        });
        $("p").on("click",function()
        {
            $(this).remove();
        });
    });
 </script>
</head>
<body>
  <div>
    <label>This is Div.</label><br>
  </div>
  <p>This is Paragraph.</p>
```

Manipulate HTML Attributes using jQuery

The following table lists jQuery methods to get or set value of attribute, property, text or html.

jQuery Method	Description
attr()	Get or set the value of specified attribute of the target element(s).
prop()	Get or set the value of specified property of the target element(s).
html()	Get or set html content to the specified target element(s).
text()	Get or set text for the specified target element(s).
val()	Get or set value property of the specified target element.

jQuery attr() Method

jQuery attr() method is used to get or set the value of specified attribute of DOM element.

Syntax:
$('selector expression').attr('name','value');

jQuery prop() Method
The jQuery prop() method gets or sets the value of specified property to the DOM element(s).
Syntax:
$('selector expression').prop('name','value');

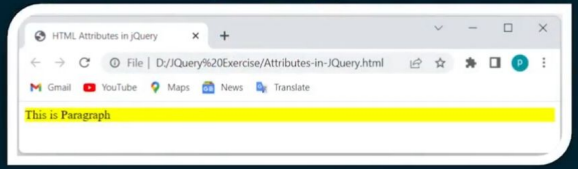
jQuery html() Method
The jQuery html() method gets or sets html content to the specified DOM element(s).
Syntax:
$('selector expression').html('content');

jQuery text() Method
The jQuery text() method gets or sets the text content to the specified DOM element(s).
Syntax: $('selector expression').text('content');

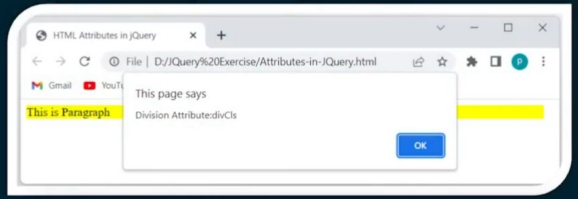
Program

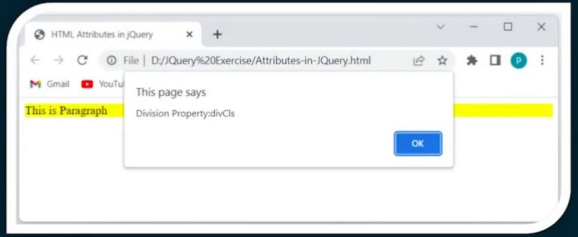
```
<html>
<head>
  <title>HTML Attributes in jQuery</title>
  <script src="jquery/jquery.js"></script>
  <script>
    $(function()
    {
       $("#myDiv").on("click", function()
       {
          var divattr=$('#myDiv').attr('class')
          alert('Division Attribute:'+divattr);
          var divprop=$('#myDiv').prop('class')
          alert('Division Property:'+divprop);
          var divHTML=$('#myDiv').html();
          alert('Division HTML:'+divHTML);
          var divText=$('#myDiv').text();
          alert('Division Text:'+divText);
       });
    });
 </script>
</head>
<body>
  <div id="myDiv" class="divCls">
    <p style="background-color:yellow;width:100%">This is Paragraph</p>
  </div>
</body>
</html>
```

Output(Before Clicking this Paragraph)



Output(After Clicking this Paragraph)





Program(Calculator.html)

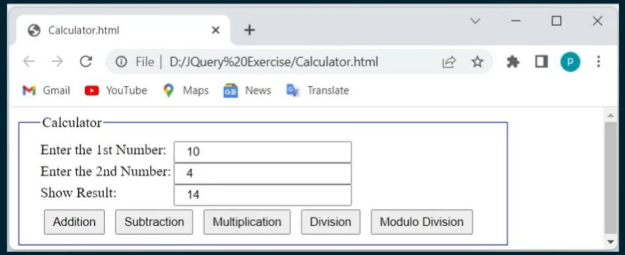
```
<html>
<head>
 <style>
   fieldset
   {
    border:1px solid blue;
    width:500px;
    height:120px;
    padding:12px 24px;
   }
            input
   {
      width:200px;
      height:20px;
      padding:10px 12px;
   }
   label
   {
     width:150px;
     float:left;
   }
 button
   {
     margin:4px;
     padding:4px 8px;
   }
```

```
<script src="jQuery/jquery-1.11.3.js"></script>
  <script>
    $(function()
    {
       $("button").on("click", function()
       {
         let input1=parseInt($("#input1").val());
         let input2=parseInt($("#input2").val());
         //alert(typeof(input1));
         //alert(typeof(input2));
         if(input1!=null && input2!=null)
         {
           let button_text=$(this).text();
           switch(button_text)
           {
              case 'Addition':
                result=input1+input2;
                $("#output").val(result);
                break;
                case 'Subtraction':
                result=input1-input2;
                $("#output").val(result);
                break;
```

```
            case 'Multiplication':
            result=input1*input2;
            $("#output").val(result);
            break;
            case 'Division':
            result=input1/input2;
            $("#output").val(result);
            break;
            case 'Modulo Division':
            result=input1%input2;
            $("#output").val(result);
            break;
          }
        }
        else
        {
          alert('Please Provide Input');
        }
      });
  });
 </script>
</head>
```

```
<body>
 <fieldset>
  <legend>Calculator</legend>
  <label for="input1">Enter the 1st Number:</label>
  <input type="number" id="input1"><br>
  <label for="input2">Enter the 2nd Number:</label>
  <input type="number" id="input2"><br>
  <label for="output">Show Result:</label>
  <input type="number" id="output"><br>
  <button>Addition</button>
  <button>Subtraction</button>
  <button>Multiplication</button>
  <button>Division</button>
  <button>Modulo Division</button>
 </fieldset>
</body>
</html>
```

Output:



Manipulate DOM Element's Dimensions using jQuery

The jQuery library includes various methods to manipulate DOM element's dimensions like height, width, offset, position etc.

The following table lists all the jQuery methods to get or set DOM element's dimensions.

jQuery Method	Description
height()	Get or set height of the specified element(s).
innerHeight()	Get or set inner height (padding + element's height) of the specified element(s).
outerHeight()	Get or set outer height (border + padding + element's height) of the specified element(s).
offset()	Get or set left and top coordinates of the specified element(s).
position()	Get the current coordinates of the specified element(s).
width()	Get or set the width of the specified element(s).
innerWidth()	Get or set the inner width (padding + element's width) of the specified element(s).
outerWidth()	Get or set outer width (border + padding + element's width) of the specified element(s).

jQuery height() Method

The jQuery height()method gets or sets height of the specified DOM element(s).

Syntax:

```
$('selector expression').height(value);
```

Specify a selector to get the reference of an element and call height() method to get the height in pixel. To set the height of specified elements, specify height as integer parameter in height() method.

jQuery width() Method

The jQuery width() method gets or sets width of the specified DOM element(s).

Syntax:

$('selector expression').width(value);

Specify a selector to get the reference of an element and call width() method to get the width in pixel. Specify width as integer parameter to set the width.

jQuery offset() Method

The jQuery offset() method gets or sets coordinates of the specified element(s).

Syntax:

$('selector expression').offset(options);

Specify a selector to get the reference of an element and call offset() method to get the jQuery object which has left and top property. Specify JSON object with left and top property with the coordinates where you want to move the element.

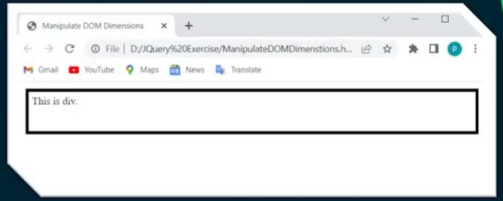
Points to Remember :

1. jQuery dimension methods allows you to manipulate dimensions of DOM elements.
2. Use the selector to get the reference of an element(s) and then call jQuery dimension methods to edit it.
3. Important DOM manipulation methods: height(), width(), offset(), position() etc.

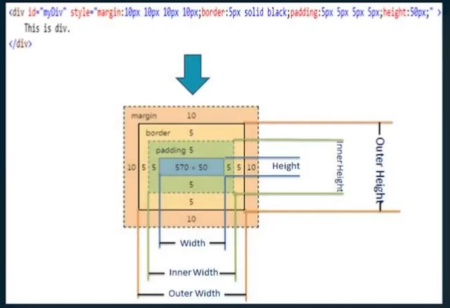
```
<html>
<head>
 <title>Manipulate DOM Dimensions</title>
 <script src="jQuery/jquery-1.11.3.js"></script>
 <script>
  $(function()
  {
    $("#myDiv").on("click", function()
    {
      var divHeight=$('#myDiv').height();
      alert('Division Height:'+divHeight);
      var divInnerHeight=$('#myDiv').innerHeight();
      alert('Division Inner Height:'+divInnerHeight);
      var divOuterHeight=$('#myDiv').outerHeight();
      alert('Division Outer Height:'+divOuterHeight);
      var divOffset=$('#myDiv').offset();
      alert('Division Offset:'+divOffset);
      var divWidth=$('#myDiv').width();
      alert('Division Width:'+divWidth);
      var divInnerWidth=$('#myDiv').innerWidth();
      alert('Division Inner Width:'+divInnerWidth);
      var divOuterWidth=$('#myDiv').outerWidth();
      alert('Division Outer Width:'+divOuterWidth);
    });
  });
 </script>
</head>
<body>
 <div id="myDiv" style="margin:10px 10px 10px 10px;border:5px solid black;padding:5px 5px 5px 5px;height:50px;">
  This is div.
 </div>
</body>
</html>
```

output

output(After clicking div)



The following figure shows various dimensions of an element.



Traversing DOM Elements using jQuery

The jQuery library includes various methods to traverse DOM elements in a DOM hierarchy.

The following table lists jQuery methods for traversing DOM elements.

jQuery Methods	Description
children()	Get all the child elements of the specified element(s)
each()	Iterate over specified elements and execute specified call back function for each element.
find()	Get all the specified child elements of each specified element(s).
first()	Get the first occurrence of the specified element.
next()	Get the immediately following sibling of the specified element.
parent()	Get the parent of the specified element(s).
prev()	Get the immediately preceding sibling of the specified element.
siblings()	Get the siblings of each specified element(s)

Let's look at some of the important jQuery traversing methods.

jQuery each() Method

The jQuery each() method iterates over each specified element (specified using selector) and executes callback function for each element.

Syntax:

$('selector expression').each(callback function);

jQuery children() Method

The jQuery children() method get the child element of each element specified using selector expression.

Syntax:

$('selector expression').children();

jQuery find() Method

The jQuery find() method returns all the matching child elements of specified element(s).

Syntax:

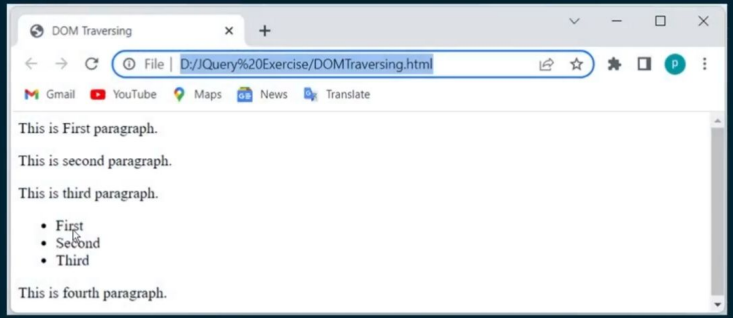
$('selector expression').find('selector expression to find child elements');

Program(DOMTraversing.html)

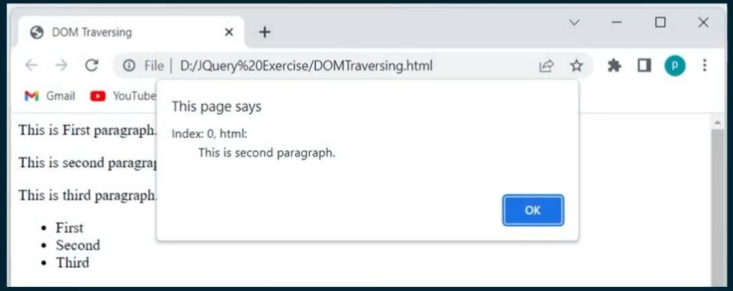
```
<html>
<head>
  <title>DOM Traversing</title>
  <script src="jQuery/jquery-1.11.3.js"></script>
  <script>
    $(function()
    {
       $("#myDiv").on("click", function()
       {
         $("#myDiv").children().each(function(index)
         {
          alert('Index: ' + index + ', html: ' + $(this).html());
         });
         alert('Siblings:'+$('#myDiv').siblings().html());
         alert('Children:'+$('#myDiv').children().html());
         alert('Previous element to #myDiv: ' + $('#myDiv').prev().html());
         alert('Next element to #myDiv: ' + $('#myDiv').next().html());
         alert('Next element to #inrDiv: ' + $('#inrDiv').next().html());
         alert('Parent of #inrDiv: ' + $('#inrDiv').parent().html());
       });

       $("p").each(function(index){
        alert('index '+index+',test:'+$(this).html());
       });
    });
</script>
```

```
<body>
  <div>
    <p>This is First paragraph.</p>
  </div>
  <div id="myDiv">
    <p>
      This is second paragraph.
    </p>
    <div id="inrDiv">
      <p>This is third paragraph.</p>
    </div>
    <div>
      <ul>
        <li>First</li>
        <li>Second</li>
        <li>Third</li>
      </ul>
    </div>
  </div>
  <div>
    <p>This is fourth paragraph.</p>
  </div>
</body>
</html>
```

Output



Output(After Clicking Second Paragraph)



Output

Points to Remember :

1. The jQuery traversing methods allow you to iterate DOM elements in a DOM hierarchy.
2. Use the selector to get the reference of an element(s) and then call jQuery traversing methods to edit.
3. Important DOM manipulation methods: each(), children(), find(), first(), parent(), next(), previous(), siblings() etc.

CSS Manipulation using jQuery

The jQuery library includes various methods to manipulate style properties and CSS class of DOM element(s).

The following table lists jQuery methods for styling and css manipulation.

jQuery Methods	Description
css()	Get or set style properties to the specified element(s).
addClass()	Add one or more class to the specified element(s).
hasClass()	Determine whether any of the specified elements are assigned the given CSS class.
removeClass()	Remove a single class, multiple classes, or all classes from the specified element(s).
toggleClass()	Toggles between adding/removing classes to the specified elements

CSS Manipulation using jQuery

The jQuery library includes various methods to manipulate style properties and CSS class of DOM element(s).

The following table lists jQuery methods for styling and css manipulation.

jQuery Methods	Description
css()	Get or set style properties to the specified element(s).
addClass()	Add one or more class to the specified element(s).
hasClass()	Determine whether any of the specified elements are assigned the given CSS class.
removeClass()	Remove a single class, multiple classes, or all classes from the specified element(s).
toggleClass()	Toggles between adding/removing classes to the specified elements

jQuery css() Method

The jQuery css() method gets or sets style properties to the specified element(s).

Syntax:

```
$('selector expression').css('style property name','value');
$('selector expression').css({
                        'style property name':'value',
 });
```

jQuery addClass() method

The jQuery addClass() method adds single or multiple css class to the specified element(s).

Syntax:

```
$('selector expression').addClass('css class name');
```

jQuery toggleClass() Method

The jQuery toggleClass() method toggles between adding/removing classes to the specified elements.

Syntax:

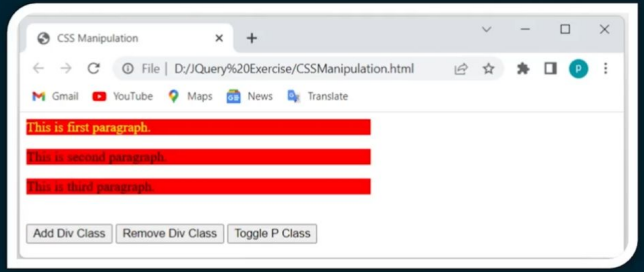
```
$('selector expression').toggleClass('css class name')
```

Program

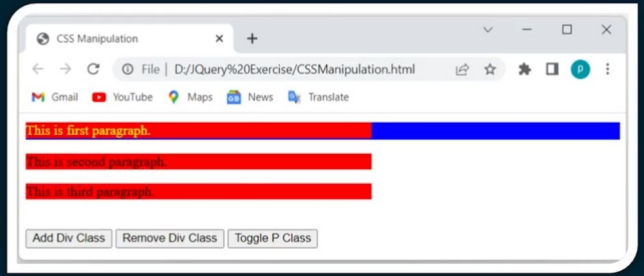
```
<html>
<head>
  <title>CSS Manipulation</title>
  <script src="jQuery/jquery-1.11.3.js"></script>
  <script>
    $(function()
    {
       $("#myDiv").on("click", function()
       {
         $('#myDiv').css('color','yellow'); //setting the text color
                                  $('p').css({'background-color': 'red','width':'400px'});
         alert($('p').css('background-color')); // returns rgb(255,255,0) for yellow color
       });
                  $("#addDiv").click(function()
       {
         $('#myDiv').toggleClass('blueDiv');
       });
                  $("#removeDiv").click(function()
       {
         $('#myDiv').removeClass('blueDiv');
       });
       $("#toggleP").click(function()
       {
         $('p').toggleClass('myStyle');
       });
    });
```

```
 <style>
   .blueDiv
   {
            background-color:blue;
            margin: 10px 0 0 0;
            height:20px;
   }
   .myStyle
   {        font-size:16px;
            font-weight:bold;
   }
 </style>
</head>
<body>
  <div id="myDiv">
    <p>This is first paragraph.</p>
  </div>
  <div>
    <p>This is second paragraph.</p>
  </div>
  <div >
    <p>This is third paragraph.</p>
  </div>
  <br>
  <button id="addDiv">Add Div Class</button>
  <button id="removeDiv">Remove Div Class</button>
  <button id="toggleP">Toggle P Class</button>
```

Output



After Clicking Add Div Class Button



jQuery Animation

jQuery includes methods which give special effects to the elements on hiding, showing, changing style properties, and fade-in or fade-out operation. These special effect methods can be useful in building an interactive user interface.

The following table lists jQuery methods for adding special effects to the DOM elements.

jQuery Methods for Special Effects	Description
animate()	Perform custom animation using element's style properties.
queue()	Show or manipulate the queue of functions to be executed on the specified element.
stop()	Stop currently running animations on the specified element(s).
fadeIn()	Display specified element(s) by fading them to opaque.
fadeOut()	Hides specified element(s) by fading them to transparent.
fadeTo()	Adjust the opacity of the specified element(s)
fadeToggle()	Display or hide the specified element(s) by animating their opacity.
hide()	Hide specified element(s).
show()	Display specified element(s).
toggle()	Display hidden element(s) or hide visible element(s).
slideUp()	Hide specified element(s) with sliding up motion.
slideDown()	Display specified element(s) with sliding down motion.
slideToggle()	Display or hide specified element(s) with sliding motion.

jQuery animate() Method

The jQuery animate() method performs custom animation using element's style properties. The animate() method changes existing style properties to the specified properties with motion.

Specify a selector to get the reference of an element to which you want to add animation effect and then call animate() method with JSON object for style properties, speed of animation and other options.

Syntax:

```
$('selector expression').animate({ stylePropertyName : 'value'},
                duration,
                easing,
                callback);
$('selector expression').animate({ propertyName : 'value'},{ options });
```

Set Animation Duration

You can apply animation duration in millisecond's as a second parameter of animate() method.

Example: Set Duration

```
$('img').animate({
        height: '100px',
        width: '100px'
        },5000);
```

Apply Easing Method

Specify a string parameter indicating which easing function to use for the transition. The jQuery library provides two easing function: linear and swing.

Example: Apply Easing Method

```
$('img').animate({
        height: '100px',
        width: '100px'
      },5000,'swing');
```

Callback Function on Animation Complete

Specify a callback function to execute when animation is complete.

Example: Specify Callback Function

```
$('img').animate({
                height: '100px',
                width: '100px'
            },5000,
function ()
{
            $('#msg').text('Animation completed..');
            });
<img src="cat.jpg"></img>
<p id="msg"></p>
```

Specify Animation Options

You can specify various options as JSON object. The options include duration, easing, queue, step, progress, complete, start, done and always.

```
function myAnimation()
 {
   $('img').animate({
                    height: '100px',
                    width: '100px'
                },
                {   // options parameter
                    duration: 5000,
                    complete: function () {
                        $(this).animate({
                            height: '200px',
                            width: '200px'
                        }, 5000,
                        function () {
                        $('#msg').text('Animation completed..');
                        });
                    },
                    start: function () {
                        $('#msg').text('starting animation..');
                    }
                });
   }
```

jQuery queue() Method

The jQuery queue() method shows or manipulates the queue of special effect functions to be executed on the specified element.

Syntax: $('selector expression').queue();

jQuery fadeIn() Method

The jQuery fadeIn() method displays specified element(s) by fading them to opaque.

Syntax: $('selector expression').fadeIn(speed, easing, callback);

jQuery fadeOut() Method

The jQuery fadeOut() method hides specified element(s) by fading them to transparent.

Syntax: $('selector expression').fadeOut(speed, easing, callback);

jQuery hide() and show() Method

The jQuery hide() method hides and show() method displays the specified element. You can specify speed, easing and callback function which will be executed when hide/show completes.

Syntax:

$('selector expression').hide(speed, easing, callback);

$('selector expression').show(speed, easing, callback);

jQuery toggle() Method

The jQuery toggle() method hides or displays specified element(s).

Syntax: $('selector expression').toggle(speed, easing, callback)

jQuery Event Methods

The jQuery library provides methods to handle DOM events. Most jQuery methods correspond to native DOM events.

The following table lists all jQuery methods and corresponding DOM events.

Category	jQuery Method	DOM Event
Form events	blur	onblur
	change	onchange
	focus	onfocus
	focusin	onfocusin
	select	onselect
	submit	onsubmit
Keyboard events	keydown	onkeydown
	keypress	onkeypress
	keyup	onkeyup
	focusout	
Mouse events	click	onclick
	dblclick	ondblclick
	focusout	
	hover	
	mousedown	onmousedown
	mouseenter	onmouseenter
	mouseleave	onmouseleave
	mousemove	onmousemove
	mouseout	onmouseout
	mouseover	onmouseover
	mouseup	onmouseup
	Toggle	
Browser events	Error	onerror()
	Resize	onresize
	Scroll	onscroll
Document loading	Load	onload
	Ready	
	Unload	Onunload

Event Handling

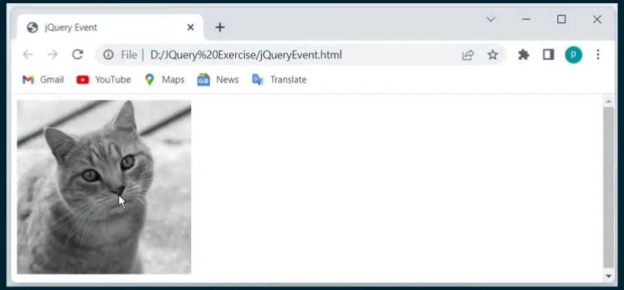
To handle DOM events using jQuery methods, first get the reference of DOM element(s) using jQuery selector and invoke appropriate jQuery event method.

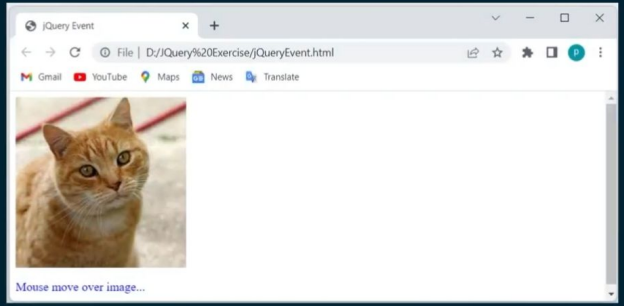
Program(jQueryEvent.html)

```
<html>
 <head>
   <title>jQuery Event</title>
   <script src="jQuery/jquery-1.11.3.js"></script>
   <script>
     $(function()
     {
       $('img').click(function()
       {
          $('#msg').text('Image Clicked...');
       });
       $('img').dblclick(function()
       {
          $(this).css('filter','invert(100%)');
          $('#msg').text('Image Double Clicked...');
       });
       $('img').mousemove(function()
       {
          $(this).css('filter','grayscale(0%)');
          $('#msg').text('Mouse move over image...');
```

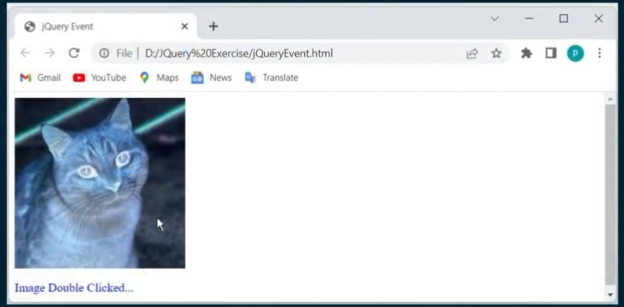
```
$('img').mouseout(function()
        {
            $(this).css('filter','grayscale(100%)');
            $('#msg').text('Mouse out image...');
        });
    });
  </script>
  <style>
        /*change image color to grayscale mode*/
        img
         {
            -webkit-filter: grayscale(100%);
            filter: grayscale(100%);
         }
  </style>
 </head>
 <body>
   <img src="images/cat.jpg"></img><br>
   <p id="msg" style="color:blue;"></p>
 </body>
</html>
```

Output


jQuery Event
File | D:/JQuery%20Exercise/jQueryEvent.html
Gmail
YouTube
Maps
News
Translate


jQuery Event
File | D:/JQuery%20Exercise/jQueryEvent.html
Gmail
YouTube
Maps
News
Translate
Mouse move over image...

Output


jQuery Event
File | D:/JQuery%20Exercise/jQueryEvent.html
Gmail
YouTube
Maps
News
Translate
Image Double Clicked...

Event Object

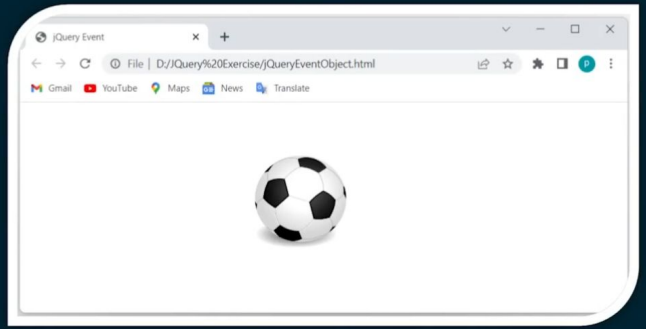
jQuery passes an event object to every event handler function. The event object includes important properties and methods for cross-browser consistency e.g. target, pageX, pageY, relatedTarget etc.

Program(jQueryEventObject.html)

```
<html>
 <head>
  <title>jQuery Event</title>
  <script src="jQuery/jquery-1.11.3.js"></script>
  <script>
   $(function()
   {
     $('img').mousemove(function(e)
     {
        var x = e.pageX;
        var y = e.pageY;
        var newposX = x - 60;
        var newposY = y - 60;
        $(this).css("transform","translate3d("+newposX+"px,"+newposY+"px,0px)");
     });
   });
  </script>
```

```
<style>
        img
         {
            width:120px;
                    height:120px;
                    position:absolute;
            transform:translate3d(-10%,-10%,0); //translate3d(tx, ty, tz)
            /* cubic-bezier(x1,y1,x2,y2)
            x1,y1,x2,y2 Required. Numeric values. x1 and x2 must be a number from 0 to 1 */
            transition:transform 1s cubic-bezier(.02,1.23,.79,1.08);
          }
              .box
        {
            width:100%;
            height:100%;
            position:relative;
            overflow:hidden;
            cursor:crosshair;
         }
   </style>
  </head>
  <body>
    <div class="box">
      <img src="images/ball.png"></img>
    </div>
  </body>
</html>
```

Output


jQuery Event
File | D:/JQuery%20Exercise/jQueryEventObject.html
Gmail
YouTube
Maps
News
Translate

this Keyword in Event Handler

this keyword in an event handler represents a DOM element which raised an event.

Hover Events

jQuery provides various methods for mouse hover events e.g. mouseenter, mouseleave, mousemove, mouseover, mouseout and mouseup.

Note: You can use **hover()** method instead of handling mouseenter and mouseleave events separately.

Event Binding using on()

jQuery allows you to attach an event handler for one or more events to the selected elements using **on()** method.

Internally all of the shorthand methods uses on() method. The on() method gives you more flexibility in event binding.

Syntax:

on(types, selector, data, fn)

- Types = One or more space-separated event types and optional namespaces
- Selector = selector string
- Data = data to be passed to the handler in event. data when an event is triggered.
- Fn = A function to execute when the event is triggered.

Binding Multiple Events

You can also specify multiple event types separated by space.

Example: Multiple Events Binding

```
$('myDiv').on('mouseenter mouseleave', function() {
    $(this).text('The mouse entered or left from the div' );
});
<div id="myDiv" style="width:100px;height:100px">
</div>
```

Specify Named Function as Event Handler

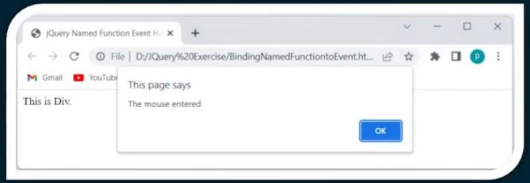
You can create separate functions and specify that as a handler. This is useful if you want to use the same handler for different events or events on different elements.

jQuery **on()** method is replacement of **live()** and **delegate()** method.

Example:Binding Named Function to Event

```
<html>
 <head>
  <title>jQuery Named Function Event Handler</title>
  <script src="jQuery/jquery-1.11.3.js"></script>
  <script>
   $(function()
   {
     $('#myDiv').on('mouseenter', mouseHandler);
   });
   var mouseHandler = function()
   {
     alert("The mouse entered" );
   };
  </script>
 </head>
 <body>
  <div id="myDiv">
   This is Div.
  </div>
 </body>
</html>
```

Output(When Mouse Enter to the Div)



Program(EventBubbling.html)

```
<html>
  <head>
    <title>jQuery Event Bubbling</title>
    <script src="jQuery/jquery.js"></script>
    <script>
      $(function()
      {
        $('div').click(function(event)
        {
          alert(event.target.tagName + ' clicked');
        });
      });
    </script>
    <style>
      div
      {
        width:200px;
        height:100px;
        background-color:red;
      }
      p
      {
        background-color:green;
      }
      span
      {
        background-color:yellow;
      }
    </style>
  </head>
```

jQuery DOM Manipulation

jQuery provides methods such as attr(), html(), text() and val() which act as getters and setters to manipulate the content from HTML documents.

Document Object Model (DOM) - is a W3C (World Wide Web Consortium) standard that allows us to create, change, or remove elements from the HTML or XML documents.

Here are some basic operations which you can perform on DOM elements with the help of jQuery standard library methods –

- Extract the content of an element
- Change the content of an element
- Adding a child element under an existing element
- Adding a parent element above an existing element
- Adding an element before or after an existing element
- Replace an existing element with another element
- Delete an existing element
- Wrapping content with-in an element

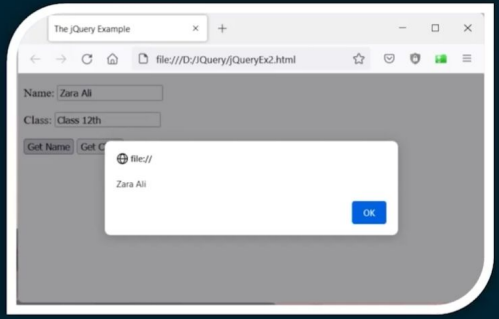
Get Form Fields

jQuery val() method is used to get the value from any form field. Following is simple syntax of this method.

```
$(selector).val();
```

Example: Following is an example to show how to get the value an input field with jQuery method val() –

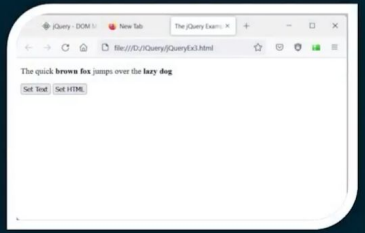
```
<!doctype html>
<html>
<head>
<title>The jQuery Example</title>
<script src="jquery/jquery.js"></script>
<script>
  $(document).ready(function() {
    $("#b1").click(function(){
      alert($("#name").val());
    });
    $("#b2").click(function(){
      alert($("#class").val());
    });
  });
</script>
</head>
<body>
  <p>Name: <input type="text" id="name" value="Zara Ali"/></p>
  <p>Class: <input type="text" id="class" value="Class 12th"/></p>
    <button id="b1">Get Name</button>
  <button id="b2">Get Class</button>
</body>
</html>
```



Example

Following example shows how to set the content of an element with the jQuery text() and html() methods:

```
<!doctype html>
<html>
<head>
<title>The jQuery Example</title>
<script src="https://www.tutorialspoint.com/jquery/jquery-3.6.0.js"></script>
<script>
  $(document).ready(function() {
    $("#text").click(function(){
      $("p").text("The quick <b>brown fox</b> jumps over the <b>lazy dog</b>");
    });
    $("#html").click(function(){
      $("p").html("The quick <b>brown fox</b> jumps over the <b>lazy dog</b>");
    });
  });
</script>
</head>
<body>
  <p>Click on any of the two buttons to see the result</p>
    <button id="text">Set Text</button>
  <button id="html">Set HTML</button>
</body>
</html>
```



Set Form Fields

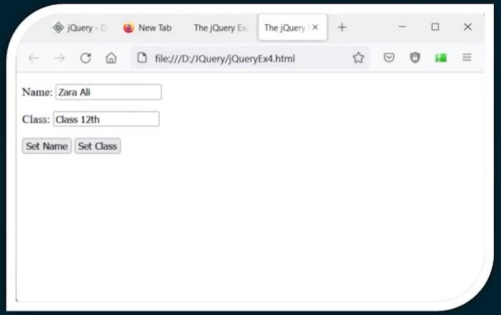
jQuery val() method is also used to set the value from any form field. Following is simple syntax of this method when it is used to set the value.

```
$(selector).val(val);
```

Here val is he value to be set for the input field. We can provide an optional callback function which will be called when the value of the field will be set.

Example: Following is another example to show how to set the value an input field with jQuery method val() <!doctype html>

```
<html>
<head>
<title>The jQuery Example</title>
<script src="jquery/jquery.js"></script>
<script>
  $(document).ready(function() {
    $("#b1").click(function(){
      $("#name").val("Zara Ali");
    });
    $("#b2").click(function(){
      $("#class").val("Class 12th");
    });
  });
</script>
</head>
<body>
  <p>Name: <input type="text" id="name" value=""/></p>
  <p>Class: <input type="text" id="class" value=""/></p>

  <button id="b1">Set Name</button>
  <button id="b2">Set Class</button>
</body>
</html>
```



jQuery - Replacement Elements

The jQuery replaceWith() method can be used to replace a complete DOM element with another HTML or DOM element. Following is the syntax of the method:

```
$(selector).replaceWith(val);
```

Here val is what you want to have instead of original element. This could be HTML or simple text.

Example

Following is an example where we will replace a <p> element with an <h1> element and then further we replace <h1> element with <h2> element.

```
<!doctype html>
<html>
<head>
<title>The jQuery Example</title>
<script src="https://www.tutorialspoint.com/jquery/jquery-3.6.0.js"></script>
<script>
  $(document).ready(function() {
    $("#b1").click(function(){
      $("p").replaceWith("<h1>This is new paragraph</h1>");
    });
    $("#b2").click(function(){
      $("h1").replaceWith("<h2>This is New heading</h2>");
    });
  });
</script>
</head>
<body>
  <p>Click below button to replace me</p>


  <button id="b1">Replace Paragraph</button>
  <button id="b2">Replace Heading</button>
</body>
</html>
```

Output


file:///D:/JQuery/jQueryEx5.html
Click below button to replace me
Replace Paragraph
Replace Heading


file:///D:/JQuery/jQueryEx5.html
This is new heading
Replace Paragraph
Replace Heading


file:///D:/JQuery/jQueryEx5.html
This is another heading
Replace Paragraph
Replace Heading

jQuery - Add Elements

jQuery provides various methods to add new DOM elements in the existing HTML document. You can add these new elements at various locations (before, after any of the existing tags) based on your requirements.

jQuery append() Method

The jQuery append() method adds the content at the end of the matched each element(s). You can also append multiple elements in a single function call.

Following is the syntax of the append() method:

```
$(selector).append(content, [content]);
```

jQuery after() Method

The jQuery after() method adds the content after the matched each element(s). You can also insert multiple elements in a single function call.

Following is the syntax of the after() method:

```
$(selector).after(content, [content]);
```

Here content parameter could be a HTML string, a DOM element, text node, array of elements and text nodes or jQuery object to insert at the end of each element in the set of matched elements.

jQuery - Traversing Ancestors

jQuery provides methods to traverse upwards inside the DOM tree to find ancestor(s) of a given element. These methods can be used to find a parent, grandparent, great-grandparent, and so on for a given element inside the DOM.

There are following three methods to traverse upward inside the DOM tree:

parent() - returns the direct parent of the each matched element.

parents() - returns all the ancestor elements of the matched element.

parentsUntil() - returns all ancestor elements until it finds the element given as selector argument.

jQuery - Traversing Descendants

jQuery provides methods to traverse downwards inside the DOM tree to find descendant(s) of a given element. These methods can be used to find a child, grandchild, great-grandchild, and so on for a given element inside the DOM.

There are following three methods to traverse downwards inside the DOM tree:

- children() - returns all the direct children of the matched element.
- find() - returns all the descendant elements of the matched element.

The children() method differs from find() in that children() only travels a single level down the DOM tree where as find() method can traverse down multiple levels to select descendant elements (children, grandchildren, great-grandchildren etc.) as well.

jQuery children() Method

The jQuery children() method returns all the direct children of the matched element. Following is a simple syntax of the method:

jQuery find() Method

The jQuery find() method returns all descendants of the matched element. Following is a simple syntax of the method:

$(selector).find([ilter)

Here filter selector is mandatory for this method. To return all the descendants of the matched element, we need to pass * as a filter otherwise if the filter is supplied as an element, the elements will be filtered by testing whether they match it.

```
<!doctype html>
<html>
<head>
<title>The jQuery Example</title>
<script src="jquery/jquery-3.6.0.js"></script>
<script>
  $(document).ready(function() {
    $("button").click(function(){
      $( ".grand-parent" ).find("li").css( "border", "2px solid red" );
    });
  });
</script>
<style>
  .great-grand-parent *{display:block; border:2px solid #aaa; color:#aaa; padding:5px; margin:5px;}
</style>
</head>
```

Example without Callback Function

First let's take a jQuery program which does not make use of callback function so here alert message is being displayed even before the hide effect is getting completed.

```
<!doctype html>
<html>
<head>
<title>The jQuery Example</title>
<script src="jquery/jquery-3.6.0.js"></script>
<script>
  $(document).ready(function() {
    $("div").click(function(){
      $(this).hide(1000);
      alert("I'm hidden now");
    });
  });
</script>
<style>
  div{ margin:10px;padding:12px; border:2px solid #666; width:60px; cursor:pointer}
</style>
</head>
<body>
  <p>Click on any of the squares to see the result:</p>

  <div>Hide Me</div>
  <div>Hide Me</div>
  <div>Hide Me</div>
</body>
</html>
```

jQuery Callback Functions

jQuery callback functions are required due to asynchronous nature of Javascript (jQuery) code execution. jQuery effects may take sometime to complete, so there is a chance that the next lines of code may get executed while the effects are still being executed. To handle asynchronous execution of the code, jQuery allows to pass a callback in all the effect methods and the purpose of this callback function is to be executed only when the effect gets completed.

Example –rewrite the above program with call back functions.

```
<!doctype html>
<html>
<head>
<title>The jQuery Example</title>
<script src="jquery/jquery-3.6.0.js"></script>
<script>
  $(document).ready(function() {
    $("div").click(function(){
      $(this).hide(1000, function(){
        alert("I'm hidden now");
      });
    });
  });
</script>
<style>
  div{ margin:10px;padding:12px; border:2px solid #666; width:60px; cursor:pointer}
</style>
</head>
<body>
  <p>Click on any of the squares to see the result:</p>
  <div>Hide Me</div>
  <div>Hide Me</div>
  <div>Hide Me</div>
</body>
</html>
```

jQuery – Utilities

JQuery provides several utilities in the format of $(name space). These methods are helpful to complete the programming tasks. A few of the utility methods are as show below.

$.trim()

$.trim() is used to Removes leading and trailing whitespace

```
$.trim( "    lots of extra whitespace    " );
```

$.each()

$.each() is used to Iterates over arrays and objects

```
$.each([ "foo", "bar", "baz" ], function( idx, val ) {
  console.log( "element " + idx + " is " + val );
});
$.each({ foo: "bar", baz: "bim" }, function( k, v ) {
  console.log( k + " : " + v );
});
```

.each() can be called on a selection to iterate over the elements contained in the selection. .each(), not $.each(), should be used for iterating over elements in a selection.

$.inArray()

$.inArray() is used to Returns a value's index in an array, or -1 if the value is not in the array.

```
var myArray = [ 1, 2, 3, 5 ];
 if ( $.inArray( 4, myArray ) !== -1 ) {
  console.log( "found it!" );
}
```

$.extend()

$.extend() is used to Changes the properties of the first object using the properties of subsequent objects.

```
var firstObject = { foo: "bar", a: "b" };
var secondObject = { foo: "baz" };
var newObject = $.extend( firstObject, secondObject );
 console.log( firstObject.foo );
console.log( newObject.foo );
```

$.proxy()

$.proxy() is used to Returns a function that will always run in the provided scope – that is, sets the meaning of this inside the passed function to the second argument

```
var myFunction = function() {
  console.log( this );
};
var myObject = {
  foo: "bar"
};

myFunction(); // window
var myProxyFunction = $.proxy( myFunction, myObject );
myProxyFunction();
```

$.browser

$.browser is used to give the information about browsers

```
jQuery.each( jQuery.browser, function( i, val ) {
  $( "<div>" + i + " : <span>" + val + "</span>" )
  .appendTo( document.body );
});
```

$.contains()

$.contains() is used to returns true if the DOM element provided by the second argument is a descendant of the DOM element provided by the first argument, whether it is a direct child or nested more deeply.

```
$.contains( document.documentElement, document.body );
$.contains( document.body, document.documentElement );
```

$.data()

$.data() is used to give the information about data

```
<html lang = "en">
  <head>
    <title>jQuery.data demo</title>
    <script src = "https://code.jquery.com/jquery-1.10.2.js">
    </script>
  </head>
  <body>
    <div>
      The values stored were <span></span>
        and <span></span>
    </div>
    <script>
      var div = $( "div" )[ 0 ];
      jQuery.data( div, "test", {
        first: 25,
        last: "tutorials"
      });

      $( "span:first" ).text( jQuery.data( div, "test" ).first );
      $( "span:last" ).text( jQuery.data( div, "test" ).last );
    </script>
  </body>
</html>
```

output: The values stored were 25 and tutorials

$.fn.extend()

$.fn.extend() is used to extends the jQuery prototype

```
<html lang = "en">
  <head>
    <script src = "https://code.jquery.com/jquery-1.10.2.js">
    </script>
  </head>
  <body>
    <label><input type = "checkbox" name = "android">
      Android</label>
    <label><input type = "checkbox" name = "ios"> IOS</label>
    <script>
      jQuery.fn.extend({
          check: function() {
          return this.each(function() {
            this.checked = true;
          });
        },
          uncheck: function() {
          return this.each(function() {
            this.checked = false;
          });
        }
      });
       // Use the newly created .check() method
      $( "input[type = 'checkbox']" ).check();

    </script>
  </body>
</html>
```

$.isWindow()

$.isWindow() is used to recognise the window

```
<!doctype html>
<html lang = "en">
  <head>
    <meta charset = "utf-8">
    <title>jQuery.isWindow demo</title>
    <script src = "https://code.jquery.com/jquery-1.10.2.js">
    </script>
  </head>

  <body>
    Is 'window' a window? <b></b>

    <script>
      $( "b" ).append( "" + $.isWindow( window ) );
    </script>
  </body>
</html>
```

$.now()

The $.now() method is an alias for Date.now() .
It returns a number(of milliseconds) which is representing the current time
(new Date).getTime()

$.isXMLDoc()

$.isXMLDoc() checks whether a file is an xml or not
jQuery.isXMLDoc(document)
jQuery.isXMLDoc(document.body)

$.globalEval()

$.globalEval() is used to execute the javascript globally

```
function test() {
   jQuery.globalEval( "var new Var = true;" )
}
test();
```

$.dequeue()

$.dequeue() is used to execute the next function in the queue

$.now()

The $.now() method is an alias for Date.now() .
It returns a number(of milliseconds) which is representing the current time
(new Date).getTime()

$.isXMLDoc()

$.isXMLDoc() checks whether a file is an xml or not
jQuery.isXMLDoc(document)
jQuery.isXMLDoc(document.body)

$.globalEval()

$.globalEval() is used to execute the javascript globally

```
function test() {
   jQuery.globalEval( "var new Var = true;" )
}
test();
```

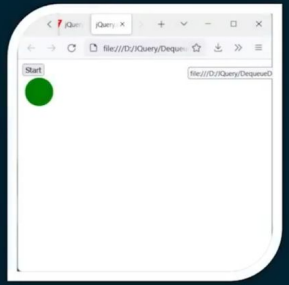
$.dequeue()

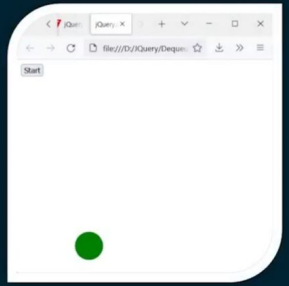
$.dequeue() is used to execute the next function in the queue

```
<!doctype html>
<html lang = "en">
  <head>
    <meta charset = "utf-8">
    <title>jQuery.dequeue demo</title>

    <style>
      div {
        margin: 3px;
        width: 50px;
        position: absolute;
        height: 50px;
        left: 10px;
        top: 30px;
        background-color: green;
        border-radius: 50px;
      }
      div.red {
        background-color: blue;
      }
    </style>

    <script src = "https://code.jquery.com/jquery-1.10.2.js"></script>
  </head>
```

```
<body>
<div>
    <button>Start</button>
</div>

    <script>
      $( "button" ).click(function() {
        $( "div" )
        .animate({ left: '+ = 20000px' }, 1000 )
        .animate({ top: '100px' }, 600 )

        .queue(function() {
          $( this ).toggleClass( "red" );
          $.dequeue( this );
        })

        .animate({ left:'100px', top:'300px' }, 700 );
      });
    </script>   </body>
</html>
```

It provides the output as shown below –





jQuery – Interactions

Interactions could be added basic mouse-based behaviours to any element. Using with interactions, We can create *sortable lists, resizeable elements, drag & drop behaviours*. Interactions also make great building blocks for more complex widgets and applications.

SL.No.	Interactions & Description
1	Drag able Enable drag able functionality on any DOM element.
2	Drop able Enable any DOM element to be drop able.
3	Resize able Enable any DOM element to be resize-able.
4	Select able Enable a DOM element (or group of elements) to be selectable.
5	Sort able Enable a group of DOM elements to be sortable.

jQuery Interaction -Selectable()

```
<!doctype html>
<html lang = "en">
  <head>
    <meta charset = "utf-8">
    <title>jQuery UI Selectable - Default functionality</title>

    <link rel = "stylesheet"
      href = "//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">

    <script type = "text/javascript"
      src = "https://www.tutorialspoint.com/jquery/jquery-3.6.0.js">
    </script>

    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-ui.min.js">
    </script>

    <style>
      #feedback { font-size: 1.4em; }
      #selectable .ui-selecting { background: #FECA40; }
      #selectable .ui-selected { background: #F39814; color: white; }
      #selectable { list-style-type: none; margin: 0; padding: 0; width: 60%; }
      #selectable li { margin: 3px; padding: 0.4em; font-size: 1.4em; height: 18px; }
    </style>
```

jQuery serializeArray()

The jQuery serializedArray() Method is used to create a JavaScript array of objects by serializing form values. It operates on a jQuery collection of forms and form controls. You can select one or more form elements such as <input>, <textarea> or the form element itself.

Syntax:

```
$(selector).serializeArray()
```

jQuery serializeArray() example

Let's take an example of serializeArray() method.

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        var x = $("form").serializeArray();
        $.each(x, function(i, field){
            $("#results").append(field.name + ":" + field.value + " ");
        });
    });
});
</script>
</head>
```

jQuery Form -serialize()

jQuery serialize() method is used to create a text string in standard URL-encoded notation. It is used in form controls like <input>, <textarea>, <select> etc. It serializes the form values so that its serialized values can be used in the URL query string while making an AJAX request.

Syntax:

```
$ (selector).serialize()
```

jQuery serialize() example

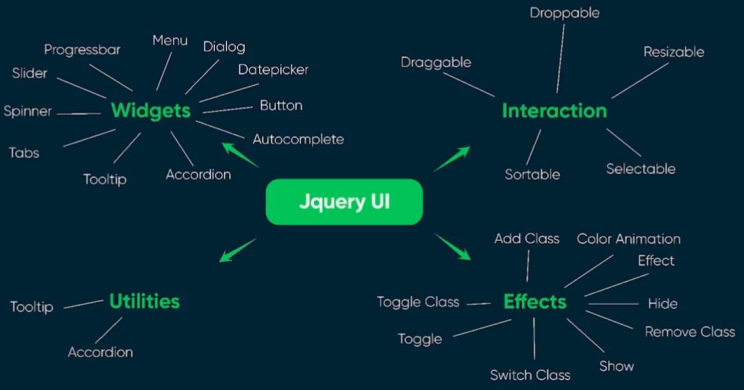
Let's take an example which serializes a form values.

```
<!DOCTYPE html>
<html>
<head>
 <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
<script>
$(document).ready(function(){
   $("button").click(function(){
      $("div").text($("form").serialize());
   });
});
</script>
</head>
<body>
<form action="">
 First name: <input type="text" name="FirstName" value="Kadhir"><br>
 Last name: <input type="text" name="LastName" value="Aadhanl"><br>
</form>
<button>Serialize form values</button>
<div></div>
</body>
</html>
```

jQuery UI

jQuery UI is categorized into four groups: interactions, widgets, effects, utilities. These will be discussed in detail here. The structure of the library is as shown in the image below –

- Interactions – These are the interactive plugins like drag, drop, resize and more which give the user the ability to interact with DOM elements.
- Widgets – Using widgets which are jQuery plugins, you can create user interface elements like accordian,datepicker etc.
- Effects – These are built on the internal jQuery effects. They contain a full suite of custom animations and transitions for DOM elements.
- Utilities – These are a set of modular tools the jQuery UI library uses internally.



Benefits of jQuery UI

The below are some of the benefits of jQuery UI –

- Cohesive and Consistent APIs.
- Comprehensive Browser Support.
- Open Source and Free to Use.
- Good Documentation.
- Powerful Theming Mechanism.
- Stable and Maintenance Friendly.

jQuery UI Interactions

Sr.No.	Interactions	Description
1	Dragable	Enable drag able functionality on any DOM element.
2	Dropable	Enable any DOM element to be drop able.
3	Resizeable	Enable any DOM element to be resize-able.
4	Selectable	Enable a DOM element (or group of elements) to be selectable.
5	Sortable	Enable a group of DOM elements to be sortable.

Dragable

The Drag-able function can be used with interactions in jQuery UI. This function can enable drag-able functionality on any DOM element. We can drag the drag-able element by clicking on it with mouse.

Syntax

Here is the simple syntax to use dragable –

```
$("#draggable").draggable();
```

Droppable

The Drop-able function can be used with interactions in jQuery UI. This function can be enabled drop-able functionality on any DOM element. We can drop the drag-able element by clicking on it with mouse.

Syntax

Here is the simple syntax to use dropp-able –

```
$("#droppable").droppable();
```

Resizable

The Resize-able function can be used with interactions in jQuery UI. This function can be enabled Resize- able functionality on any DOM element. With the cursor grab the right or bottom border and drag to the desired width or height.

Syntax

Here is the simple syntax to use resiza-able –

```
$("#resizable").resizable();
```

Selectable

The Select-able function can be used with interactions in jQuery UI. This function can be enabled select able functionality on any DOM element. Draw a box with your cursor to select items. Hold down the Ctrl key to make multiple non-adjacent selections.

Syntax

Here is the simple syntax to use select-able –

```
$("#selectable").selectable();
```

Sortable

The Sort-able function can be used with interactions in jQuery UI. This function can be enabled sortable functionality on any DOM element. Click on and drag an element to a new spot within the list, and the other items will adjust to fit. By default, sortable items share draggable properties.

Syntax

Here is the simple syntax to use sort-able –

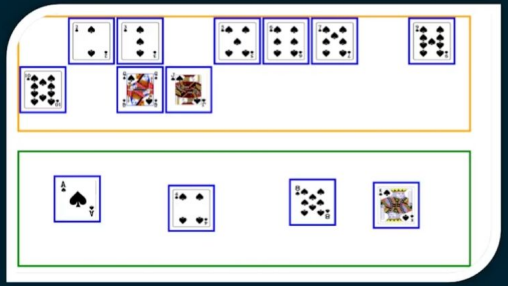
```
$(function() {
  $("#sortable").sortable();
  $("#sortable").disableSelection();
});
```

Program(PlayingCards.html)

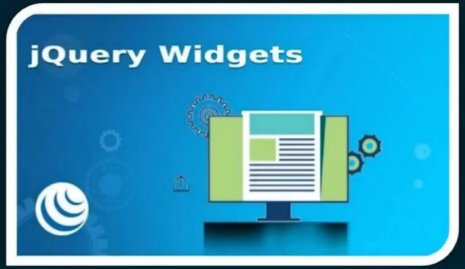
```
<html>
<head>
  <title>Playing Cards</title>
  <script src="jQuery/jquery.js"></script>
  <script src="jQuery/jquery-ui.js"></script>
  <script src="jQuery/jquery-ui.min.js"></script>
  <link rel="stylesheet" href="jquery-ui.css">
  <style>
       .cards { width: 100px; height: 100px; padding: 0.5em; border:5px solid blue;}
       #droplist{ margin:50 0px; width:80%; height:300px; border:5px solid green;}
       .cards: hover{border-color:red;}
       #cardlist{width:80%; height:300px; border:5px solid orange;}
   <style>
</head>
```

```
<body>
  <div id="cardlist">
    <img src="images/playingCards/CardAce.png" id="card1" class="cards"></img>
    <img src="images/playingCards/Card2.png" id="card2" class="cards"></img>
    <img src="images/playingCards/Card3.png" id="card3" class="cards"></img>
    <img src="images/playingCards/Card4.png" id="card4" class="cards"></img>
    <img src="images/playingCards/Card5.png" id="card5" class="cards"></img>
    <img src="images/playingCards/Card6.png" id="card6" class="cards"></img>
    <img src="images/playingCards/Card7.png" id="card7" class="cards"></img>
    <img src="images/playingCards/Card8.png" id="card8" class="cards"></img>
    <img src="images/playingCards/Card9.png" id="card9" class="cards"></img>
    <img src="images/playingCards/Card10.png" id="card10" class="cards"></img>
    <img src="images/playingCards/CardKing.png" id="cardking" class="cards"></img>
    <img src="images/playingCards/CardQueen.png" id="cardqueen" class="cards"></img>
    <img src="images/playingCards/CardJack.png" id="cardjack" class="cards"></img>
  </div>

  <div id="droplist">
  </div>
</body>
```

```
<script>
    $(function()
    {
      $(".cards").sortable({helper: 'clone'}).disableSelection()        ;
      $(".cards").draggable();
      $(".cards").selectable();
      //$(".cards").draggable({containment:"#droplist"});
      $("#droplist").resizable();

});
  </script>
</html>
```



Introduction to jQuery Widgets

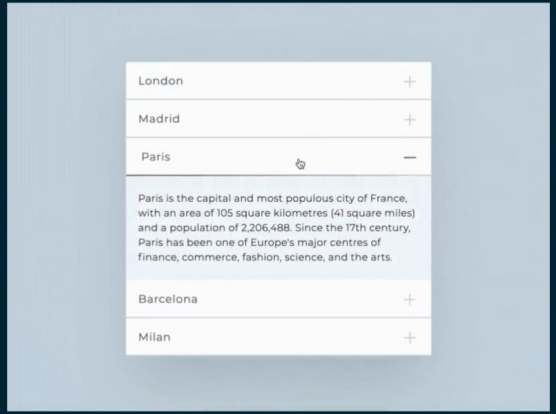


jQuery Widgets are specialized, platform-independent, cross browser compatible, *stateful* plugins rich in features, with a full life cycle along with some methods and events, built on an extensible base called jQuery UI Widget Factory (facilitates state management and provides conventions for tasks such as exposing plugin methods, changing options after instantiation, thus, allowing us to make some here and there changes in the existing functionality of the jQuery Widgets).

And in combination with animated visual effects, CSS, HTML, and jQuery themes, form a complete jQuery UI, thus, taking JavaScript and HTML UI development to a completely new level and becoming one of the fastest growing JavaScript UI frameworks.

Top jQuery Widgets

1. Accordion Widget
2. Autocomplete Widget
3. Button Widget
4. Datepicker Widget
5. Dialog Widget
6. Menu Widget
7. Progressbar Widget
8. Select the Menu Widget
9. Slider Widget
10. Spinner Widget
11. Tabs Widget
12. Tooltip Widget



1. jQuery UI – Accordion

Accordion Widget in jQuery UI is a jQuery based expandable and collapsible content holder that is broken into sections and probably looks like tabs. jQuery UI provides accordion() method to achieve this.

Syntax

The accordion() method can be used in two forms –

- $(selector, context).accordion(options) Method
- $(selector, context).accordion("action", params) Method

$ (selector, context).accordion (options) Method

The accordion (options) method declares that an HTML element and its contents should be treated and managed as accordion menus. The options parameter is an object that specifies the appearance and behavior of the menu involved.

Syntax

$(selector, context).accordion(options);

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows –

$(selector, context).accordion({option1: value1, option2: value2..... });

The following table lists the different options that can be used with this method

Sr.No.	Option	Description
1	Active	Indicates the index of the menu that is open when the page is first accessed. By default its value is 0, i.e the first menu.
2	Animate	This option is used to set how to animate changing panels. By default its value is {}.
3	collapsible	This option when set to true, it allows users to close a menu by clicking on it. By default, clicks on the open panel's header have no effect. By default its value is false.
4	disabled	This option when set to true disables the accordion. By default its value is false.
5	Event	This option specifies the event used to select an accordion header. By default its value is click.
6	Header	This option specifies a selector or element to override the default pattern for identifying the header elements. By default its value is > li > :first-child,> :not(li):even.
7	heightStyle	This option is used to control the height of accordion and panels. By default its value is auto.
8	Icons	This option is an object that defines the icons to use to the left of the header text for opened and closed panels. The icon to use for closed panels is specified as a property named header, whereas the icon to use for open panels is specified as a property named headerSelected. By default its value is { "header": "ui-icon-triangle-1-e", "activeHeader": "ui-icon-triangle-1-s" }.

2. jQuery UI - Autocomplete

Auto completion is a mechanism frequently used in modern websites to provide the user with a list of suggestions for the beginning of the word, which he/she has typed in a text box. The user can then select an item from the list, which will be displayed in the input field. This feature prevents the user from having to enter an entire word or a set of words.

jQuery UI provides an autocomplete widget – a control that acts a lot like a <select> dropdown, but filters the choices to present only those that match what the user is typing into a control. jQuery UI provides the autocomplete() method to create a list of suggestions below the input field and adds new CSS classes to the elements concerned to give them the appropriate style.

Any field that can receive input can be converted into an Autocomplete, namely, <input> elements, <textarea> elements, and elements with the contenteditable attribute.

Syntax

The autocomplete() method can be used in two forms −

- $(selector, context).autocomplete (options) Method
- $(selector, context).autocomplete ("action", params) Method

$ (selector, context).autocomplete (options) Method

The autocomplete (options) method declares that an HTML <input> element must be managed as an input field that will be displayed above a list of suggestions. The options parameter is an object that specifies the behavior of the list of suggestions when the user is typing in the input field.

Syntax

$(selector, context).autocomplete (options);

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows −

$(selector, context).autocomplete({option1: value1, option2: value2..... });

The following table lists the different options that can be used with this method –

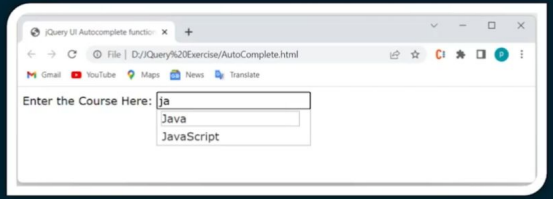
Sr.No.	Option	Description
1	appendTo	This option is used append an element to the menu. By default its value is null.
2	autoFocus	This option when set to true, the first item of the menu will automatically be focused when the menu is shown. By default its value is false.
3	delay	This option is an Integer representing number of milliseconds to wait before trying to obtain the matching values (as specified by the source option). This can help reduce thrashing when non-local data is being obtained by giving the user time to enter more characters before the search is initiated. By default its value is 300.
4	disabled	This option if specified and true, the autocomplete widget is initially disabled. By default its value is false.
5	minLength	The number of characters that must be entered before trying to obtain the matching values (as specified by the source option). This can prevent too large a value set from being presented when a few characters isn't enough to whittle the set down to a reasonable level. By default its value is 1.
6	Position	This option identifies the position of the suggestions menu in relation to the associated input element. The of option defaults to the input element, but you can specify another element to position against. By default its value is {my: "left top", at: "left bottom", collision: "none"}.
7	Source	This option specifies the manner in which the data that matches the input data is obtained. A value must be provided or the autocomplete widget won't be created. By default its value is none; must be specified.

The following table lists the different options that can be used with this method –

Sr.No.	Option	Description
1	appendTo	This option is used append an element to the menu. By default its value is null.
2	autoFocus	This option when set to true, the first item of the menu will automatically be focused when the menu is shown. By default its value is false.
3	delay	This option is an Integer representing number of milliseconds to wait before trying to obtain the matching values (as specified by the source option). This can help reduce thrashing when non-local data is being obtained by giving the user time to enter more characters before the search is initiated. By default its value is 300.
4	disabled	This option if specified and true, the autocomplete widget is initially disabled. By default its value is false.
5	minLength	The number of characters that must be entered before trying to obtain the matching values (as specified by the source option). This can prevent too large a value set from being presented when a few characters isn't enough to whittle the set down to a reasonable level. By default its value is 1.
6	Position	This option identifies the position of the suggestions menu in relation to the associated input element. The of option defaults to the input element, but you can specify another element to position against. By default its value is {my: "left top", at: "left bottom", collision: "none"}.
7	Source	This option specifies the manner in which the data that matches the input data is obtained. A value must be provided or the autocomplete widget won't be created. By default its value is none; must be specified.

```
Program
<!doctype html>
<html lang = "en">
  <head>
    <meta charset = "utf-8">
    <title>jQuery UI Autocomplete functionality</title>
    <link href = "CSS/jquery-ui.css" rel = "stylesheet">
    <script src = "jQuery/jquery-3.6.0.js"></script>
    <script src = "jQuery/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function()
      {
        var availableTutorials =
                                [
                                        "AutoCAD",
                                        "ActionScript",
                                "Bootstrap",
                                "C",
                                "C++",
                                "C#",
                                "Derby",
                                "ECMA Script",
                                "Java",
                                "JavaScript",
                                        "MongoDB",
                                "MariaDB",
                                "Perl",
```

Output



3. jQuery UI – Button

jQuery UI provides button() method to transform the HTML elements (like buttons, inputs and anchors) into themeable buttons, with automatic management of mouse movements on them, all managed transparently by jQuery UI.

In order to group radio buttons, Button also provides an additional widget, called Buttonset. Buttonset is used by selecting a container element (which contains the radio buttons) and calling .buttonset().

Syntax

The button() method can be used in two forms –

- $(selector, context).button (options) Method
- $(selector, context).button ("action", params) Method

$ (selector, context).button (options) Method

The button (options) method declares that an HTML element should be treated as button. The options parameter is an object that specifies the behavior and appearance of the button.

Synt$(selector, context).button (options);

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows –

$(selector, context).button({option1:value1,option2: value2..... });

The following table lists the different options that can be used with this method –

Sr.No.	Option	Description
1	disabled	This option deactivates the button is set to true. By default its value is false.
2	icons	This option specifies that one or two icons are to be displayed in the button: primary icons to the left, secondary icons to the right. The primary icon is identified by the primary property of the object, and the secondary icon is identified by the secondary property. By default its value is primary: null, secondary: null.
3	label	This option specifies text to display on the button that overrides the natural label. If omitted, the natural label for the element is displayed. In the case of radio buttons and checkboxes, the natural label is the <label> element associated with the control. By default its value is null.
4	text	This option specifies whether text is to be displayed on the button. If specified as false, text is suppressed if (and only if) the icons option specifies at least one icon. By default its value is true.

4. JQuery UI – Datepicker

Datepickers in jQueryUI allow users to enter dates easily and visually. You can customize the date format and language, restrict the selectable date ranges and add in buttons and other navigation options easily.

jQueryUI provides datepicker() method that creates a datepicker and changes the appearance of HTML elements on a page by adding new CSS classes. Transforms the <input>, , and elements in the wrapped set into a datepicker control.

Syntax

The datepicker() method can be used in two forms –

- $(selector, context).datepicker (options) Method
- $(selector, context).datepicker ("action", [params]) Method

$ (selector, context).datepicker (options) Method

The datepicker (options) method declares that an <input> element (or , or , depending on how you choose to display the calendar) should be managed as a datepicker. The options parameter is an object that specifies the behavior and appearance of the datepicker elements.

Syntax

$(selector, context).datepicker(options);

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows –

$(selector, context).datepicker({option1: value1, option2: value2..... });

The following table lists the different options that can be used with this method (some of the methods)–

Sr.No.	Option	Description
1	altField	This option specifies a jQuery selector for a field that is also updated with any date selections. The altFormat option can be used to set the format for this value. This is quite useful for setting date values into a hidden input element to be submitted to the server, while displaying a more user-friendly format to the user. By default its value is "".
2	altFormat	This option is used when an altField option is specified. It provides the format for the value to be written to the alternate element. By default its value is "".
3	appendText	This option is a String value to be placed after the <input> element, intended to show instructions to the user. By default its value is "".
4	autoSize	This option when set to true resizes the <input> element to accommodate the datepicker's date format as set with the dateFormat option. By default its value is false.
5	beforeShow	This option is a callback function that's invoked just before a datepicker is displayed, with the <input> element and datepicker instance passed as parameters. This function can return an options hash used to modify the datepicker. By default its value is "".
6	beforeShowDay	This option is a callback function which takes a date as parameter, that's invoked for each day in the datepicker just before it's displayed, with the date passed as the only parameter. This can be used to override some of the default behavior of the day elements. This function must return a three-element array. By default its value is null.
7	buttonImage	This option specifies the path to an image to be displayed on the button enabled by setting the showOn option to one of buttons or both. If buttonText is also provided, the button text becomes the alt attribute of the button. By default its value is "".
8	buttonImageOnly	This option if set to true, specifies that the image specified by buttonImage is to appear standalone (not on a button). The showOn option must still be set to one of button or both for the image to appear. By default its value is false.
9	buttonText	This option specifies the caption for the button enabled by setting the showOn option to one of button or both. By default its value is "...".
10	calculateWeek	This option is a custom function to calculate and return the week number for a date passed as the lone parameter. The default implementation is that provided by the $.datepicker.iso8601Week() utility function.

11	changeMonth	This option if set to true, a month dropdown is displayed, allowing the user to directly change the month without using the arrow buttons to step through them. By default its value is false.
12	changeYear	This option if set to true, a year dropdown is displayed, allowing the user to directly change the year without using the arrow buttons to step through them. Option yearRange can be used to control which years are made available for selection. By default its value is false.
13	closeText	This option specifies the text to replace the default caption of Done for the close button. It is used when the button panel is displayed via the showButtonPanel option. By default its value is "Done".
14	constrainInput	This option if set true (the default), text entry into the <input> element is constrained to characters allowed by the current dateformat option. By default its value is true.
15	currentText	This option specifies the text to replace the default caption of Today for the current button. This is used if the button panel is displayed via the showButtonPanel option. By default its value is Today.
16	dateFormat	This option specifies the date format to be used. By default its value is mm/dd/yy.
17	dayNames	This option is a 7-element array providing the full day names with the 0th element representing Sunday. Can be used to localize the control. By default its value is ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"].
18	dayNamesMin	This option is a 7-element array providing the minimal day names with the 0th element representing Sunday, used as column headers. Can be used to localize the control. By default its value is ["Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"].
19	dayNamesShort	This option specifies a 7-element array providing the short day names with the 0th element representing Sunday. Can be used to localize the control. By default its value is ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"].
20	defaultDate	This option sets the initial date for the control, overriding the default value of today, if the <input> element has no value. This can be a Date instance, the number of days from today, or a string specifying an absolute or relative date. By default its value is null.

21	Duration	This option specifies the speed of the animation that makes the datepicker appear. Can be one of slow, normal, or fast, or the number of milliseconds for the animation to span. By default its value is normal.
22	firstDay	This option specifies which day is considered the first day of the week, and will be displayed as the left-most column. By default its value is 0.
23	gotoCurrent	This option when set to true, the current day link is set to the selected date, overriding the default of today. By default its value is false.
24	hideIfNoPrevNext	This option if set to true, hides the next and previous links (as opposed to merely disabling them) when they aren't applicable, as determined by the settings of the minDate and maxDate options. By default its value is false.
25	isRTL	This option when set to true, specifies the localizations as a right-to-left language. By default its value is false.
26	maxDate	This option sets the maximum selectable date for the control. This can be a Date instance, the number of days from today, or a string specifying an absolute or relative date. By default its value is null.
27	minDate	This option sets the minimum selectable date for the control. This can be a Date instance, the number of days from today, or a string specifying an absolute or relative date. By default its value is null.
28	monthNames	This option is a 12-element array providing the full month names with the 0th element representing January. Can be used to localize the control. By default its value is ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"].
29	monthNamesShort	This option specifies a 12-element array providing the short month names with the 0th element representing January. Can be used to localize the control. By default its value is ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"].
30	navigationAsDateFormat	This option if set to true, the navigation links for nextText, prevText, and currentText are passed through the $.datepicker.formatDate() function prior to display. This allows date formats to be supplied for those options that get replaced with the relevant values. By default its value is false.

$ (selector, context).datepicker ("action", [params]) Method

The datepicker (action, params) method can perform an action on the calendar, such as such as selecting a new date. The action is specified as a string in the first argument and optionally, one or more params can be provided based on the given action.

Basically, here actions are nothing but they are jQuery methods which we can use in the form of string.

Syntax

```
$(selector, context).datepicker ("action", [params]);
```

5. JqueryUI – Dialog

Dialog boxes are one of the nice ways of presenting information on an HTML page. A dialog box is a floating window with a title and content area. This window can be moved, resized, and of course, closed using "X" icon by default.

jQueryUI provides dialog() method that transforms the HTML code written on the page into HTML code to display a dialog box.

Syntax

The dialog() method can be used in two forms –

- $(selector, context).dialog (options) Method
- $(selector, context).dialog ("action", [params]) Method

$ (selector, context).dialog (options) Method

The dialog (options) method declares that an HTML element can be administered in the form of a dialog box. The options parameter is an object that specifies the appearance and behavior of that window.

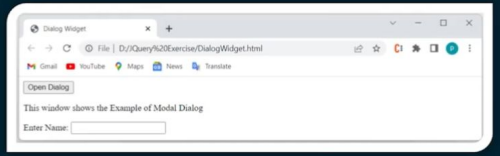
Syntax

```
$(selector, context).dialog(options);
```

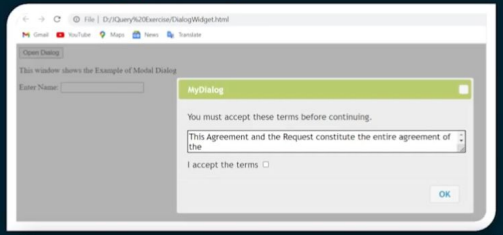
You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows –

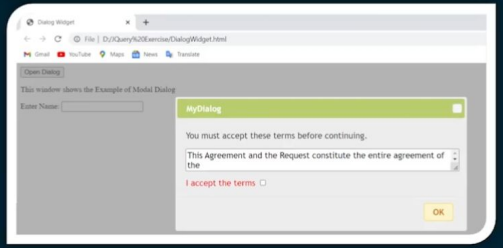
```
$(selector, context).dialog({option1:value1,option2:value2..... });
```

Output



Output(After Clicking Open Dialog Button)





6. JqueryUI – Menu

A menu widget usually consists of a main menu bar with popup menus. Items in popup menus often have sub popup menus. A menu can be created using the markup elements as long as the parent-child relation is maintained (using <ul> or <ol>). Each menu item has an anchor element.

The Menu Widget in jQueryUI can be used for inline and popup menus, or as a base for building more complex menu systems. For example, you can create nested menus with custom positioning.

jQueryUI provides menu() methods to create a menu.

Syntax

The menu() method can be used in two forms −

- $(selector, context).menu (options) Method
- $(selector, context).menu ("action", params) Method

7. JqueryUI – Progressbar

Progress bars indicate the completion percentage of an operation or process. We can categorize progress bar as determinate progress bar and indeterminate progress bar.

Determinate progress bar should only be used in situations where the system can accurately update the current status. A determinate progress bar should never fill from left to right, then loop back to empty for a single process.

If the actual status cannot be calculated, an indeterminate progress bar should be used to provide user feedback.

jQueryUI provides an easy-to-use progress bar widget that we can use to let users know that our application is hard at work performing the requested operation. jQueryUI provides progressbar() method to create progress bars.

Syntax

The progressbar() method can be used in two forms −

- $(selector, context).progressbar (options) Method
- $(selector, context).progressbar ("action", params) Method

$ (selector, context).progressbar (options) Method

The progressbar (options) method declares that an HTML element can be managed in the form of a progress bar. The options parameter is an object that specifies the appearance and behavior of progress bars.

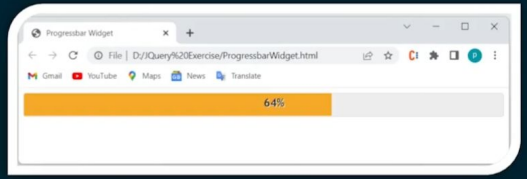
Syntax

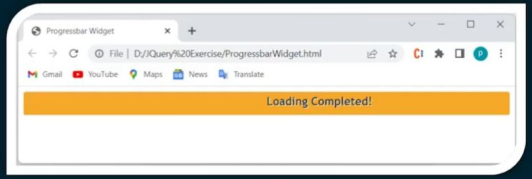
$(selector, context).progressbar (options);

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows −

$(selector,context).progressbar({option1:value1,option2: value2.....});

```
$("#progressbar-1").progressbar({
          value: false,
          change: function() {
            progressLabel.text(
              prgStatus.progressbar("value") + "%");
          },
          complete: function() {
            progressLabel.text("Loading Completed!");
          }
        });
        function progress() {
          var val = prgStatus.progressbar("value") || 0;
          prgStatus.progressbar("value", val + 1);
          if (val < 99) {
            setTimeout(progress, 100);
          }
        }
        setTimeout(progress, 100);
      });

    </script>
  </head>

  <body>
    <!-- HTML -->
    <div id="progressbar-1">
      <div class = "progress-label">
        Loading...
      </div>
    </div>
  </body>
</html>
```

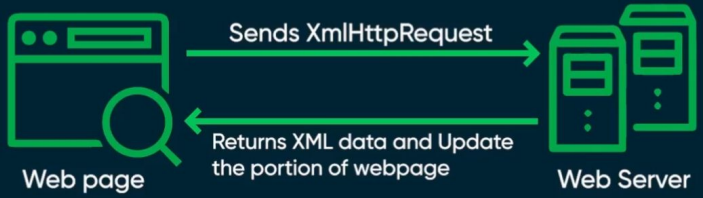




jQuery AJAX Introduction

AJAX stands for "Asynchronous JavaScript and XML". JavaScript includes features of sending asynchronous http request using XMLHttpRequest object. AJAX is about using this ability of JavaScript to send asynchronous http request and get the xml data as a response (also in other formats) and update the part of a web page (using JavaScript) without reloading or refreshing entire web page.

The following figure illustrates the Ajax functionality.



NORMAL HTTP REQUEST

AJAX XMLHTTP REQUEST

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

AJAX is a developer's dream, because you can:

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

What is AJAX?

AJAX = Asynchronous JavaScript And XML.

AJAX is not a programming language.

AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

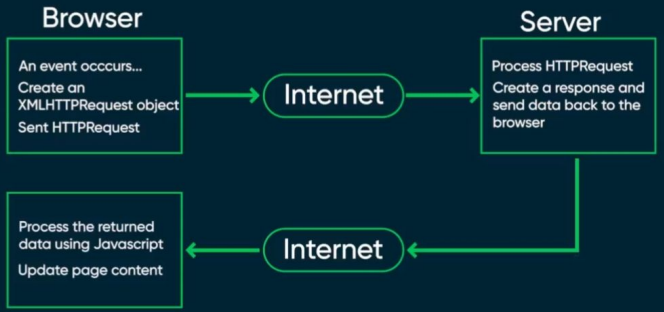
How AJAX Works:

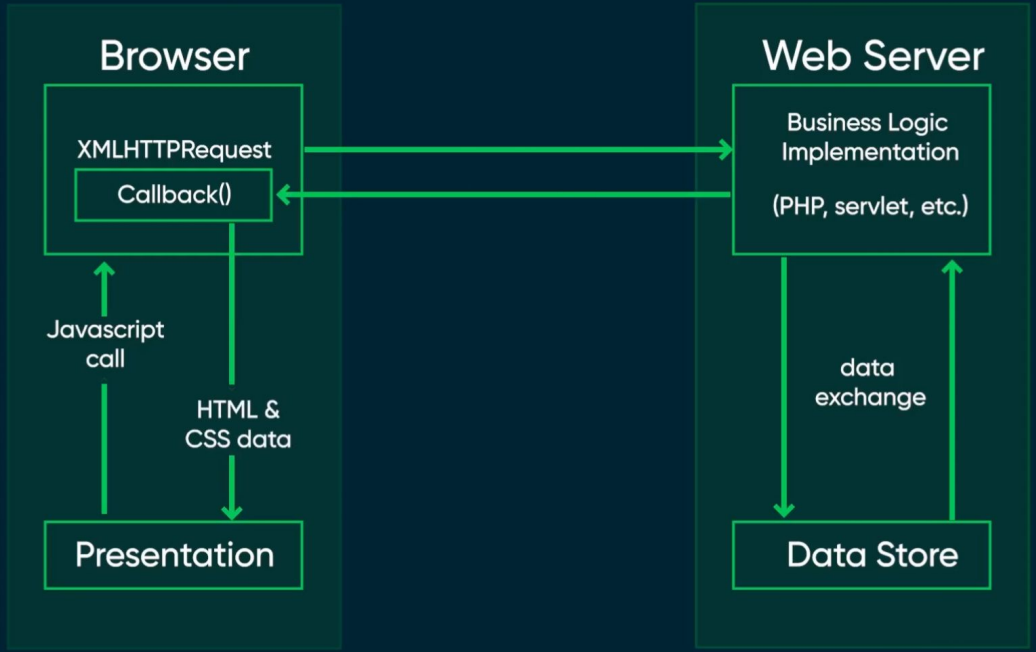
AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

How AJAX Works:




Browser
XMLHTTPRequest
Callback()
Javascript call
HTML & CSS data
Presentation
Web Server
Business Logic Implementation
(PHP, servlet, etc.)
data exchange
Data Store

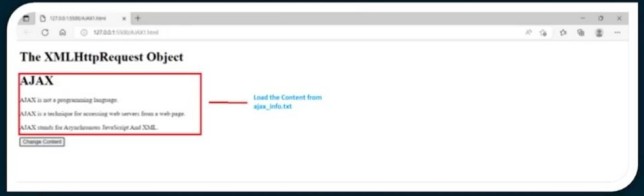
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    <h1>The XMLHttpRequest Object</h1>
    <p id="demo">Let AJAX change this text.</p>
   <button type="button" onclick="loadDoc()">Change Content</button>
  <script>
    function loadDoc()
    {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function()
      {
       if (this.readyState == 4 && this.status == 200)
       {
         document.getElementById("demo").innerHTML = this.responseText;
       }
      };
      xhttp.open("GET", "ajax_info.txt", true);
      xhttp.send();
    }
    </script>
</body>
</html>
```

ajax_info.txt

```
<h1>AJAX</h1>
<p>AJAX is not a programming language.</p>
<p>AJAX is a technique for accessing web servers from a web page.</p>
<p>AJAX stands for Asynchronous JavaScript And XML.</p>
```

Output(Before Clicking the Button)

Output(After Clicking the Button)



Access Across Domains

For security reasons, modern browsers do not allow access across domains.

This means that both the web page and the XML file it tries to load, must be located on the same server.

If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.

XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(method,url,async,user,psw)	Specifies the request method: the request type GET or POST, url: the file location async: true (asynchronous) or false (synchronous) user: optional user name psw: optional password
send()	Sends the request to the server
Used for GET requests	
send(string)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

AJAX - Send a Request To a Server

The XMLHttpRequest object is used to exchange data with a server.

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Method	Description
open(method, url, async)	Specifies the type of request method: the type of request: GET or POST url: the server (file) location async: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(string)	Sends the request to the server (used for POST)

GET Requests

A simple GET request:

Example

```
xhttp.open("GET", "demo_get.jsp", true);
xhttp.send();
```

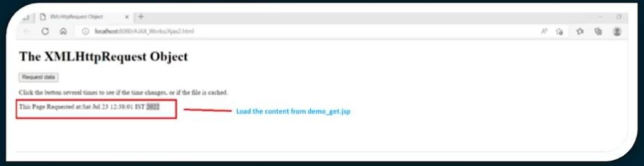
In the example above, you may get a cached result. To avoid this, add a unique ID to the URL:

Program(Ajax2.html)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
  <h1>The XMLHttpRequest Object</h1>
  <button type="button" onclick="loadDoc()">Request data</button>
  <p>Click the button several times to see if the time changes, or if the file is cached.</p>
  <p id="demo"></p>
  <script>
    function loadDoc()
    {  var xhttp = new XMLHttpRequest();
       xhttp.onreadystatechange=function()
       {
        if (this.readyState == 4 && this.status == 200)
        {
          document.getElementById("demo").innerHTML = this.responseText;
        }
       };
       xhttp.open("GET", "demo_get.jsp?t="+Math.random() , true);
       xhttp.send();
    }
</script>
</body>
```

Output(Before Button Clicking)

Output(After Button Clicking)



If you want to send information with the GET method, add the information to the URL:

Example

```
xhttp.open("GET", "demo_get2.jsp?fname=Henry&lname=Ford", true);
xhttp.send();
```

Program(Ajax3.html)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Request data</button>
<p id="demo"></p>
<script>
function loadDoc()
{
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function()
  {
   if (this.readyState == 4 && this.status == 200)
   {
     document.getElementById("demo").innerHTML = this.responseText;
   }
  };
  xhttp.open("GET", "demo_get2.jsp?fname=Henry&lname=Ford", true);
  xhttp.send();
}
</script>
</body>
</html>
```

POST Requests

A simple POST request:

Example

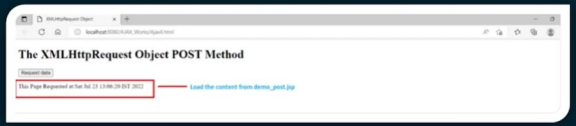
```
xhttp.open("POST", "demo_post.jsp", true);
xhttp.send();
Program(Ajax4.html)
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>XMLHttpRequest Object</title>
</head>
<body>
 <h1>The XMLHttpRequest Object Post Method</h1>
   <button type="button" onclick="loadDoc()">Request data</button>
   <p id="demo"></p>
   <script>
     function loadDoc()
     {
       var xhttp = new XMLHttpRequest();
       xhttp.onreadystatechange=function()
       {
         if (this.readyState == 4 && this.status == 200)
         {
           document.getElementById("demo").innerHTML = this.responseText;
         }
       };
       xhttp.open("POST", "demo_post.jsp?" , true);
       xhttp.send();
   }
</script>
</body>
</html>
```

demo_post.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="java.util.Date"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>XML HTTP Post Method</title>
</head>
<body>
 <%
      Date today=new Date();
      out.println("This Page Requested at:"+today);
 %>
</body>
</html>
```

Output(Before Button Clicking)

Output(After Button Clicking)



Server Response Properties

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

Server Response Methods

Method	Description
getResponseHeader()	Returns specific header information from the server resource
getAllResponseHeaders()	Returns all the header information from the server resource

The responseText Property

The responseText property returns the server response as a JavaScript string, and you can use it accordingly:

Example

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

The responseXML Property

The XML HttpRequest object has an in-built XML parser.

The responseXML property returns the server response as an XML DOM object.

Using this property you can parse the response as an XML DOM object:

Program(Ajax7.html)

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="ISO-8859-1">
  <title>The XMLHttpRequest Object</title>
 </head>
 <body>
  <h1>The XMLHttpRequest Object</h1>
  <div id="demo"></div>
  <script>
    var xhttp, xmlDoc, txt, x, i;
    xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function()
    {
       if(this.readyState == 4 && this.status == 200)
       {
          xmlDoc = this.responseXML;
          txt = "<table border=1 ><tr
            bgcolor='gray'><th>Empid</th><th>EmpName</th><th>Designation</th><th>Salary</th></tr>";
          empidNode = xmlDoc.getElementsByTagName("empid");
          empNameNode = xmlDoc.getElementsByTagName("empname");
          designationNode = xmlDoc.getElementsByTagName("designation");
          salaryNode=xmlDoc.getElementsByTagName("salary");
```

```
for (i = 0; i < empidNode.length; i++)
        {
                        txt=txt +
                            "<tr><td>"+empidNode[i].childNodes[0].nodeValue + "</td>"+
                            "<td>"+empNameNode[i].childNodes[0].nodeValue + "</td>"+
                            "<td>"+designationNode[i].childNodes[0].nodeValue + "</td>"+
                            "<td>"+salaryNode[i].childNodes[0].nodeValue + "</td>"+
                            "</tr>";
        }
        document.getElementById("demo").innerHTML = txt+"</table>";
      }
    };
    xhttp.open("GET", "Employee_details.xml", true);
    xhttp.send();
  </script>
 </body>
</html>
```

AJAX Database Example

AJAX can be used for interactive communication with a database.

Example(Ajax10.html)

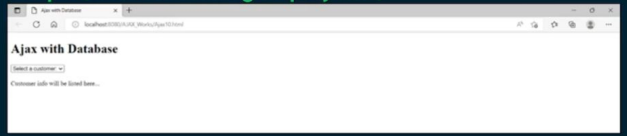
```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Ajax with Database</title>
</head>
<body>
  <h1>Ajax with Database</h1>
  <form action="">
    <select name="customers" onchange="showCustomer(this.value)">
      <option value="">Select a customer:</option>
      <option value="1000">1000</option>
      <option value="1001">1001</option>
      <option value="1002">1002</option>
      <option value="1003">1003</option>
      <option value="1004">1004</option>
    </select>
  </form>
  <br>
  <div id="txtHint">Customer info will be listed here...</div>
```

```
<script>
    function showCustomer(str)
    {
      var xhttp;
      if (str == "")
      {
        document.getElementById("txtHint").innerHTML = "";
        return;
      }
      xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function()
      {
        if (this.readyState == 4 && this.status == 200)
        {
          document.getElementById("txtHint").innerHTML = this.responseText;
        }
      };
      xhttp.open("GET", "getcustomer.jsp?customerid="+str, true);
      xhttp.send();
    }
  </script>
 </body>
</html>
```

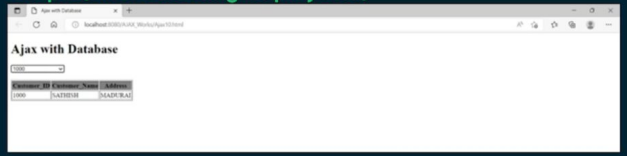
getcustomer.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="java.sql.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<table border="1">
<tr bgcolor="gray"><th>Customer_ID</th><th>Customer_Name</th><th>Address</th></tr>
 <%
 try
 {
    //Class.forName("com.mysql.jdbc.Driver");
    Class.forName("com.mysql.cj.jdbc.Driver");
    Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/MyDB","root","root");
    Statement stmt=con.createStatement();
    int customer_id=Integer.parseInt(request.getParameter("customerid"));
                ResultSet rs=stmt.executeQuery("select * from customer where customerid="+customer_id);
    while(rs.next())
    {  out.println("<tr><td>");
       out.println(rs.getInt(1)+"</td><td>"+rs.getString(2)+"</td><td>"+rs.getString(3)+"</td></tr>");
    }
    out.println("</table>");
    con.close();
 }
 catch(Exception e)
 {
     System.out.println(e);
 }

 %>
 </table>
</body>
</html>
```

Output(Before Selecting Employeeid)



Output(After Selecting EmployeeId)



Drag from top and touch the back button to exit full screen.