JavaScript

Learning outcome1:

Use Fundamental Features of JavaScript

CHAP XII:

Use of JavaScript Libraries

Usage of jQuery

Table of Contents

1. Usage of jquery

1.1. Introduction

- → **jQuery is a JavaScript Library.**
- → **jQuery greatly simplifies JavaScript programming.**
- → **jQuery is easy to learn**

1.1.1. What You Should Already Know

Before you start studying jQuery, you should have a basic knowledge of HTML, CSS, JavaScript

1.1.2. Why jQuery?

There are lots of other *JavaScript libraries* out there, but jQuery is probably the most popular, and also the most extendable. Many of the biggest companies on the Web use jQuery, such as **Google**, **Microsoft, IBM, Netflix**

1.1.3. Will jQuery work in all browsers?

The jQuery team knows all about cross-browser issues, and they have written this knowledge into the jQuery library. jQuery will run exactly the same in all major browsers.

1.2. Jquery basic

1.2.1. Adding jQuery library to a Web Pages

There are several ways to start using jQuery on your web site. You can:

Download the jQuery library from jQuery.com
Include jQuery from a CDN, like Google

The jQuery library is a single JavaScript file, and you reference it with the HTML **<script>** tag (notice that the **<script>** tag should be inside the **<head>** section)

```
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js">
</script>
</head>
```

1.2.2. jQuery Syntax

With jQuery you select (query) HTML elements and perform "***actions***" on them. The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).

Basic syntax is: **$(*selector*).*action*()**

A $ sign to define/access jQuery
A *(selector)* to "query (or find)" HTML elements
A jQuery *action()* to be performed on the element(s)

Examples:

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

1.2.3. The Document Ready Event

You might have noticed that all jQuery methods in our examples, are inside a document ready event

```
$(document).ready(function () {
    // jQuery methods go here...
});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).

It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

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

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

The jQuery team has also created an even shorter method for the document ready event:

```
$(function () {
    // jQuery methods go here...
});
```

1.2.4. jQuery Selectors

jQuery selectors are one of the most important parts of the jQuery library. *They allow you to select and manipulate HTML element(s).* They are used to "find" (or select) HTML elements based on their **name, id, classes, types, attributes, values** of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.

All selectors in jQuery start with the dollar sign and parentheses: **$()**.

Syntax	Description
`$("p")`	Element
`$("#test")`	**id selector**
`$(".test")`	**class** selector
`$("*")`	Selects **all** elements
`$(this)`	Selects the **current HTML element**
`$("p.intro")`	Selects all **<p>** elements with **class="intro"**
`$("p:first")`	Selects the first **<p>** element
`$("ul li:first")`	Selects the first ** ** element of the first ** **
`$("ul li:first-child")`	Selects the first ** ** element of every ** **
`$("[href]")`	Selects all elements with an href attribute
`$("a[target='_blank']")`	Selects all **<a>** elements with a target attribute value equal to **"_blank"**
`$("a[target!='_blank']")`	Selects all **<a>** elements with a target attribute value NOT equal to **"_blank"**
`$(":button")`	Selects all **<button>** elements and **<input>** elements of **type="button"**
`$("tr:even")`	Selects all even **<tr>** elements
`$("tr:odd")`	Selects all odd **<tr>** elements

1.3. jQuery Event Methods

Event methods trigger or attach a function to an event handler for the selected elements. The following table lists all the jQuery methods used to handle events.

Method / Property	Description
bind()	**Deprecated in version 3.0. Use the on() method instead.** Attaches event handlers to elements
blur()	Attaches/Triggers the blur event
change()	Attaches/Triggers the change event
click()	**Attaches/Triggers the click event**
dblclick()	Attaches/Triggers the double click event
delegate()	**Deprecated in version 3.0. Use the on() method instead.** Attaches a handler to current, or future, specified child elements of the matching elements
die()	**Removed in version 1.9.** Removes all event handlers added with the live() method
error()	**Removed in version 3.0.** Attaches/Triggers the error event

event.currentTarget	The current DOM element within the event bubbling phase
event.data	Contains the optional data passed to an event method when the current executing handler is bound
event.delegateTarget	Returns the element where the currently-called jQuery event handler was attached
event.isDefaultPrevented()	Returns whether **event.preventDefault()** was called for the event object
event.isImmediatePropagationStopped()	Returns whether **event.stopImmediatePropagation()** was called for the event object
event.isPropagationStopped()	Returns whether **event.stopPropagation()** was called for the event object
event.namespace	Returns the namespace specified when the event was triggered
event.pageX	Returns the mouse position relative to the left edge of the document
event.pageY	Returns the mouse position relative to the top edge of the document
event.preventDefault()	Prevents the default action of the event
event.relatedTarget	Returns which element being entered or exited on mouse movement
event.result	Contains the last/previous value returned by an event handler triggered by the specified event
event.stopImmediatePropagation()	Prevents other event handlers from being called
event.stopPropagation()	Prevents the event from bubbling up the DOM tree, preventing any parent handlers from being notified of the event
event.target	Returns which DOM element triggered the event
event.timeStamp	Returns the number of milliseconds since January 1, 1970, when the event is triggered
event.type	Returns which event type was triggered
event.which	Returns which keyboard key or mouse button was pressed for the event
focus()	Attaches/Triggers the focus event
focusin()	Attaches an event handler to the focusin event
focusout()	Attaches an event handler to the focusout event
hover()	Attaches two event handlers to the hover event
keydown()	Attaches/Triggers the keydown event
keypress()	Attaches/Triggers the keypress event
keyup()	Attaches/Triggers the keyup event
live()	**Removed in version 1.9.** Adds one or more event handlers to current, or future, selected elements
load()	**Removed in version 3.0.** Attaches an event handler to the load event
mousedown()	Attaches/Triggers the mousedown event
mouseenter()	Attaches/Triggers the mouseenter event
mouseleave()	Attaches/Triggers the mouseleave event
mousemove()	Attaches/Triggers the mousemove event
mouseout()	Attaches/Triggers the mouseout event
mouseover()	Attaches/Triggers the mouseover event

mouseup()	Attaches/Triggers the mouseup event
off()	Removes event handlers attached with the on() method
on()	Attaches event handlers to elements
one()	Adds one or more event handlers to selected elements. This handler can only be triggered once per element
$.proxy()	Takes an existing function and returns a new one with a particular context
ready()	Specifies a function to execute when the DOM is fully loaded
resize()	Attaches/Triggers the resize event
scroll()	Attaches/Triggers the scroll event
select()	Attaches/Triggers the select event
submit()	Attaches/Triggers the submit event
toggle()	**Removed in version 1.9.** Attaches two or more functions to toggle between for the click event
trigger()	Triggers all events bound to the selected elements
triggerHandler()	Triggers all functions bound to a specified event for the selected elements
unbind()	**Deprecated in version 3.0. Use the off() method instead**. Removes an added event handler from selected elements
undelegate()	Deprecated in version 3.0. Use the off() method instead. Removes an event handler to selected elements, now or in the future
unload()	**Removed in version 3.0. Use the on() or trigger() method instead.** Attaches an event handler to the unload event

1.4. jQuery Effect Methods

The following table lists all the jQuery methods for creating animation effects.

Method	Description
animate()	Runs a custom animation on the selected elements
clearQueue()	Removes all remaining queued functions from the selected elements
delay()	Sets a delay for all queued functions on the selected elements
dequeue()	Removes the next function from the queue, and then executes the function
fadeIn()	Fades in the selected elements
fadeOut()	Fades out the selected elements
fadeTo()	Fades in/out the selected elements to a given opacity
fadeToggle()	Toggles between the fadeIn() and fadeOut() methods
finish()	Stops, removes and completes all queued animations for the selected elements
hide()	Hides the selected elements
queue()	Shows the queued functions on the selected elements
show()	Shows the selected elements
slideDown()	Slides-down (shows) the selected elements
slideToggle()	Toggles between the slideUp() and slideDown() methods
slideUp()	Slides-up (hides) the selected elements

stop()	Stops the currently running animation for the selected elements
toggle()	Toggles between the hide() and show() methods

1.5. Jquery manipulation

1.5.1. jQuery DOM Manipulation

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.

1.5.2. 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.

1.5.3. 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.

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).

1.5.4. 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.

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)

1.5.5. 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.

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

1.5.6. jQuery Animations

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.

1.6. Jquery advanced

1.6.1. 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 jQuery library includes various methods to send Ajax requests. These methods internally use XMLHttpRequest object of JavaScript. The following table lists all the Ajax methods of jQuery.

jQuery Ajax Methods	Description
ajax()	Sends asynchronous http request to the server.
get()	Sends http GET request to load the data from the server.
Post()	Sends http POST request to submit or load the data to the server.
getJSON()	Sends http GET request to load JSON encoded data from the server.
getScript()	Sends http GET request to load the JavaScript file from the server and then executes it.
load()	Sends http request to load the html or text content from the server and add them to DOM element(s).

The jQuery library also includes following events which will be fired based on the state of the Ajax request.

jQuery Ajax Events	Description
ajaxComplete()	Register a handler function to be called when Ajax requests complete.
ajaxError()	Register a handler function to be called when Ajax requests complete with an error.
ajaxSend()	Register a handler function to be called before Ajax request is sent.
ajaxStart()	Register a handler function to be called when the first Ajax request begins.
ajaxStop()	Register a handler function to be called when all the Ajax requests have completed.
ajaxSuccess()	Register a handler function to be called when Ajax request completes successfully.

1.6.2. jQuery ajax() Method

The jQuery ajax() method provides core functionality of Ajax in jQuery. It sends asynchronous HTTP requests to the server. Syntax: `$.ajax(url); or $.ajax(url,[options]);`

Parameter description:

- **url**: A string URL to which you want to submit or retrieve the data
- **options**: Configuration options for Ajax request. An options parameter can be specified using JSON format. This parameter is optional.

The following table list all the options available for configuring Ajax request.

Options	Description
accepts	The content type sent in the request header that tells the server what kind of response it will accept in return.
async	By default, all requests are sent asynchronously. Set it false to make it synchronous.
beforeSend	A callback function to be executed before Ajax request is sent.
cache	A boolean indicating browser cache. Default is true.
complete	A callback function to be executed when request finishes.
contentType	A string containing a type of content when sending MIME content to the server.Default is "application/x-www-form-urlencoded; charset=UTF-8"
crossDomain	A boolean value indicating whether a request is a cross-domain.
data	A data to be sent to the server. It can be JSON object, string or array.
dataType	The type of data that you're expecting back from the server.
error	A callback function to be executed when the request fails.
global	A Boolean indicating whether to trigger a global Ajax request handler or not. Default is true.
headers	An object of additional header key/value pairs to send along with request.

ifModified	Allow the request to be successful only if the response has changed since the last request. This is done by checking the Last-Modified header. Default value is false.
isLocal	Allow the current environment to be recognized as local.
jsonp	Override the callback function name in a JSONP request. This value will be used instead of 'callback' in the 'callback=?' part of the query string in the url.
jsonpCallback	String containing the callback function name for a JSONP request.
mimeType	String containing a mime type to override the XMLHttpRequest mime type.
password	A password to be used with XMLHttpRequest in response to an HTTP access authentication request.
processData	A Boolean indicating whether data assigned to data option will be converted to a query string. Default is true.
statusCode	A JSON object containing numeric HTTP codes and functions to be called when the response has the corresponding code.
success	A callback function to be executed when Ajax request succeeds.
timeout	A number value in milliseconds for the request timeout.
type	A type of http request e.g. POST, PUT and GET. Default is GET.
url	A string containing the URL to which the request is sent.
username	A username to be used with XMLHttpRequest in response to an HTTP access authentication request.
xhr	A callback for creating the XMLHttpRequest object.
xhrFields	An object of fieldName-fieldValue pairs to set on the native XMLHttpRequest object.

Let's see how to send http requests using $.ajax() (or jQuery.ajax()) method.

Send Ajax Request

The ajax() methods performs asynchronous http request and gets the data from the server. The following example shows how to send a simple Ajax request.

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js">
  </script>
  <script type="text/javascript">
    $(document).ready(function () {

      $('#ajaxBtn').click(function () {
        $.ajax('/jquery/getdata',   // request url
          {
            success: function (data, status, xhr) {// success callback function
              $('p').append(data);
            }
```

In the above example, first parameter '/getData' of ajax() method is a url from which we want to retrieve the data. The second parameter is options parameter in JSON format where we have specified callback function that will be executed when request succeeds. You can configure other options as mentioned in the above table. **By default ajax() method performs http GET request if option parameter does not include method option.**

```
<script type="text/javascript">
  $(document).ready(function () {
    $('#ajaxBtn').click(function () {
      $.ajax('/jquery/getjsondata',
        {
          dataType: 'json', // type of response data
          timeout: 500,     // timeout milliseconds
          success: function (data, status, xhr) {   // success callback function
            $('p').append(data.firstName + ' ' + data.middleName + ' ' +  data.lastName);
          },
          error: function (jqXhr, textStatus, errorMessage) { // error callback
            $('p').append('Error: ' + errorMessage);
          }
        });
    });
```

In the above example, first parameter is a request url which will return JSON data. In the options parameter, we have specified dataType and timeout options. The dataType option specifies the type of response data, in this case it is JSON. The timeout parameter specifies request timeout in milliseconds. We have also specified callback functions for error and success.

The ajax() method returns an object of jQuery XMLHttpRequest. The following example shows how to use jQuery XMLHttpRequest object.

```
var ajaxReq = $.ajax('GetJsonData', {
   dataType: 'json',
   timeout: 500
});

ajaxReq.success(function (data, status, jqXhr) {
   $('p').append(data.firstName + ' ' + data.middleName + ' ' + data.lastName);
})

ajaxReq.error(function (jqXhr, textStatus, errorMessage) {
   $('p').append('Error: ' + errorMessage);
})
< p ></p >
```

Send Http POST request using ajax()

The ajax() method can send all type of http requests. The following example sends http POST request to the server.

```
$.ajax('/jquery/submitData', {
    type: 'POST',  // http method
    data: { myData: 'This is my data.' },  // data to submit
    success: function (data, status, xhr) {
        $('p').append('status: ' + status + ', data: ' + data);
    },
    error: function (jqXhr, textStatus, errorMessage) {
        $('p').append('Error' + errorMessage);
    }
});
<p></p>
```

In the above example, first parameter is a url which is used to submit the data. In the options parameter, we have specified a type option as a POST, so ajax() method will send http POST request. Also, we have specified data option as a JSON object containing data which will be submitted to the server. So this way you can send **GET, POST or PUT** request using **ajax()** method.

Points to Remember :

- **$.ajax()** method allows you to send asynchronous http requests to submit or retrieve data from the server without reloading the whole page.
- **$.ajax()** can be used to send http **GET, POST, PUT, DELETE** etc. request. It can retrieve any type of response from the server.
- Syntax: **$.ajax(url,[options])**
- Use option parameter to customize ajax request as per your need.

1.6.3. jQuery get() Method

The jQuery get() method sends asynchronous http GET request to the server and retrieves the data. Syntax: **$.get(url, [data],[callback]);**

Parameters Description:

- **url**: request url from which you want to retrieve the data
- **data**: data to be sent to the server with the request as a query string
- **callback**: function to be executed when request succeeds

The following example shows how to retrieve data from a text file.

```
$.get('/data.txt',  // url
    function (data, textStatus, jqXHR) {  // success callback
        alert('status: ' + textStatus + ', data:' + data);
    });
```

In the above example, first parameter is a url from which we want to retrieve the data. Here, we want to retrieve data from a txt file located at mydomain.com/data.txt. Please note that you don't need to give base address.

The second parameter is a callback function that will be executed when this GET request succeeds. This callback function includes three parameters data, textStatus and jQuery wrapper of XMLHttpRequest object. Data contains response data, textStatus contains status of request and jqXHR is a jQuery XMLHttpRequest object which you can use for further process.

The following example shows how to retrieve JSON data using get() method

```
$.get('/jquery/getjsondata', {name:'Steve'}, function (data, textStatus, jqXHR) {
    $('p').append(data.firstName);
});
<p></p>
```

In the above example, first parameter is a url from which we want to retrieve JSON data. This url can be a web service or any other url that returns data in JSON format.

The second parameter is data to be sent to the server as a query string. We have specified name parameter with value 'Steve' in the JSON format. So now, the request url would look like http://mydomain.com/jquery/getjsondata?name=Steve. The third parameter is a callback function that will be executed when this GET request succeeds.

jQuery getJSON() Method

The jQuery getJSON() method sends asynchronous http GET request to the server and retrieves the data in JSON format by setting accepts header to application/json, text/javascript. This is same as get() method, the only difference is that getJSON() method specifically retrieves JSON data whereas get() method retrieves any type of data. It is like shortcut method to retrieve JSON data.

Syntax: **$.getJSON(url,[data],[callback]);**

Parameter Description:

- **url**: request url from which you want to retrieve the data
- **data**: JSON data to be sent to the server as a query string
- **callback**: function to be executed when request succeeds

The following example shows how to retrieve JSON data using getJSON() method.

```
$.getJSON('/jquery/getjsondata', {name:'Steve'}, function (data, textStatus, jqXHR){
    $('p').append(data.firstName);
});
<p></p>
```

In the above example, first parameter is a url from which we want to get JSON data. This can be a web service or any other url that returns JSON data.

The second parameter is data to pass as query string with the GET request. So now, the request url would look like http://mydomain.com/jquery/getjsondata?name=Steve. The third parameter is a callback function which will be executed when request succeeds. The data parameter will be in the JSON format because getJson() method automatically converts server response into a JSON object.

You can attach fail and done callback methods to getJson() method as shown below.

```
$.getJSON('/jquery/getjsondata', { name:'Steve'}, function(data, textStatus, jqXHR){
    alert(data.firstName);
})
.done(function () { alert('Request done!'); })
.fail(function (jqxhr,settings,ex) { alert('failed, '+ ex); });
```

1. jQuery getScript() Method

The jQuery getScript() method sends http GET request to the server, retrieves the JavaScript file and then executes it. Internally, jQuery getScript() method calls get() method and sets dataType to script.

Syntax: **$.getScript(url, [callback]);**

Parameter Description:

- **url**: request url from which you want to download JavaScript file
- **callback**: function to be executed when request succeeds

The following example shows how to download script file using getScript() method.

```
$.getScript('/Scripts/myJavaScriptFile.js', function(script,status,jqxhr){
    alert(status);
});
```

In the above example, first parameter is a url from which we want to download script file. Here, we download myJavaScriptFile.js file for demo purpose. The second parameter is a callback function which will be executed when request succeeds. Thus, you can use different get methods to retrieve different types of resources using http get request.

Points to Remember :

- **$.get(), $.getJSON()** method allows you to send asynchronous http GET request to retrieve the data from the server without reloading whole page.
- **$.get()** can be used to retrieve any type of response from the server.
- **$.getJSON()** method is a short form method to retrieve JSON response from the server.
- **$.getScript()** sends asynchronous http GET request to retrieve the script files from the server and execute it.
- Syntax:
    - **$.get(url,[data],[callback])**
    - **$.getJSON(url,[data],[callback])**
    - **$.getScript(url,[callback])**

1.6.4. jQuery post() Method

The jQuery post() method sends asynchronous http POST request to the server to submit the data to the server and get the response.

Syntax: **$.post(url,[data],[callback],[type]);**

Parameter Description:

- **url**: request url from which you want to submit & retrieve the data.
- **data**: json data to be sent to the server with request as a form data.
- **callback**: function to be executed when request succeeds.
- **type**: data type of the response content.

Let's see how to submit data and get the response using post() method. Consider the following example.

```
$.post('/jquery/submitData',    // url
    { myData: 'This is my data.' }, // data to be submitted
    function (data, status, jqXHR) {// success callback
        $('p').append('status: ' + status + ', data: ' + data);
    })
< p ></p >
```

In the above example, first parameter is a url to which we want to send http POST request and submit the data.

The second parameter is a data to submit in JSON format, where key is the name of a parameter and value is the value of parameter.

The third parameter is a success callback function that will be called when request succeeds. The callback function can have three parameters; data, status and jqXHR. The data parameter is a response coming from the server. The following example shows how to submit and retrieve JSON data using post() method.

```
$.post('/submitJSONData',  // url
    { myData: 'This is my data.' }, // data to be submit
    function (data, status, xhr) {   // success callback function
        alert('status: ' + status + ', data: ' + data.responseData);
    },
    'json'); // response data format
```

In the above example, please notice that last parameter is a type of response data. We will get JSON data as a server response. So post() method will automatically parse response into JSON object. Rest of the parameters are same as first example. You can also attach fail and done callback methods to post() method as shown below.

```
$.post('/jquery/submitData',
        { myData: 'This is my data.' },
        function(data, status, xhr) {

            $('p').append('status: ' + status + ', data: ' + data);

        }).done(function() { alert('Request done!'); })
        .fail(function(jqxhr, settings, ex) { alert('failed, ' + ex); });

<p></p>
```

Points to Remember :

- $.post() method allows you to send asynchronous http POST request to submit and retrieve the data from the server without reloading whole page.
- Syntax: **$.post(url,[data],[callback],[type])**
- Specify type parameter for the type of response data e.g. specify 'JSON' if server return JSON data.
- Internally post() method calls ajax() method only by passing method='POST' as option.

1.6.5. jQuery load() Method

The jQuery load() method allows HTML or text content to be loaded from a server and added into a DOM element.

Syntax: **$.load(url,[data],[callback]);**

Parameters Description:

- → **url**: request url from which you want to retrieve the content
- → **data**: JSON data to be sent with request to the server.
- → **callback**: function to be executed when request succeeds

The following example shows how to load html content from the server and add it to div element.

```
$('#msgDiv').load('/demo.html');
<div id="msgDiv"></div>
```

In the above example, we have specified html file to load from the server and add its content to the div element.

Note : If no element is matched by the selector then Ajax request will not be sent.

The load() method allows us to specify a portion of the response document to be inserted into DOM element. This can be achieved using url parameter, by specifying selector with url separated by one or multiple space characters as shown in the following example.

```
$('#msgDiv').load('/demo.html #myHtmlContent');
<div id="msgDiv"></div>
```

In the above example, content of the element whose id is myHtmlContent, will be added into msgDiv element. The following is a demo.html.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <h1>This is demo html page.</h1>
    <div id="myHtmlContent">This is my html content.</div>
</body>
</html>
```

The load() method also allows us to specify data to be sent to the server and fetch the data.

```
$('#msgDiv').load('getData', // url
    { name: 'bill' },    // data
    function (data, status, jqXGR) {  // callback function
        alert('data loaded')
    });
<div id="msgDiv"></div>
```

In the above example, first parameter is a url from which we want to fetch the resources. The second parameter is data to be sent to the server. The third parameter is a callback function to execute when request succeeds.

Points to Remember :

- → $.load() method allows HTML or text content to be loaded from a server and added into a DOM element.
- → Syntax:$.post(url,[data],[callback])
- → Specify selector with url to specify a portion of the response document to be inserted into DOM element. $('#msgDiv').load('/demo.html #myHtmlContent');

1.7. Exercises

Exercises

1. How do you add the jQuery library to a web page?

There are two ways to add the jQuery library to a web page:

- → Inline: You can add the jQuery library code directly to the web page, between the <head> and </head> tags.
- → External: You can download the jQuery library file from the jQuery website and link to it from your web page.

2. What are the advantages of using an external jQuery library?

There are several advantages to using an external jQuery library:

- → It is **more efficient,** as the jQuery code only needs to be downloaded once, when the web page is first loaded.
- → It is **easier to update**, as you only need to update the external jQuery file, rather than the code in your web page.
- → It is **more secure,** as the jQuery code is hosted on a trusted server.

3. What are the disadvantages of using an external jQuery library?

There are two disadvantages to using an external jQuery library:

- → It adds an extra HTTP request to the web page, which can **slow down the loading time of the page.**
- → If the external jQuery library is unavailable, your web page will not work.

4. **What is the Document Ready event in jQuery?**

→ The Document Ready event in jQuery executes JavaScript code when the DOM of a web page is fully loaded and ready to be manipulated. It ensures that your JavaScript code is executed only after all the HTML content on the page has been parsed and is available for manipulation.

```
$(document).ready(function() {
    // Your code here
});
//Alternatively, you can use the shorthand notation:
$(function() {
    // Your code here
});
```

5. **Why is using the Document Ready event important?**

Using the Document Ready event is important because web pages are parsed and loaded progressively. When your JavaScript code runs before the DOM is fully loaded, it might fail to find the elements it needs to manipulate, leading to unexpected behavior or errors. By using the Document Ready event, you ensure that your code won't execute until all the necessary elements are present in the DOM.

6. **How do you attach an event handler to an element using jQuery?**

→ You can use jQuery's **.on()** method to attach event handlers to elements. This method allows you to specify the event type and the function to be executed when the event occurs.

```
$('#myButton').on('click', function() {
    // Code to execute when the button is clicked
    alert('Button clicked!');
});
```

7. **How can you delegate events using jQuery?**

→ Delegating events in jQuery involves attaching an event handler to a parent element rather than directly to the target element. This is particularly useful when dealing with dynamically added elements. Suppose you have an HTML structure bellow, and you want to handle a click event on the list items, even if new items are added dynamically. You can delegate the click event using the .on() method

```
<ul id="parentList">
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
$(document).ready(function() {
    $('#parentList').on('click', 'li', function() {
        alert('Clicked: ' + $(this).text());
    });
});
```

8. How do you prevent the default behavior of an event using jQuery?

In some cases, you might want to prevent the default behavior of an event, such as preventing a link from navigating to a new page when clicked. jQuery provides the .preventDefault() method to achieve this.

```
$('a').on('click', function(event) {
    // Prevent the default behavior of the click event (navigation)
    event.preventDefault();
    alert('Link clicked, but navigation prevented.');
});
```

9. How do you create sliding animations using jQuery?

→ jQuery provides methods like **.slideUp(), .slideDown(), and .slideToggle()** to create sliding animations that hide or show elements by adjusting their height.

```
$(document).ready(function() {
    // Slide down an element with a specified duration
    $('#slideDownButton').click(function() {
        $('#slideDownElement').slideDown(1000);
    });
    // Slide-up element with specified duration and a callback function
    $('#slideUpButton').click(function() {
        $('#slideUpElement').slideUp(1000, function() {
            alert('Element slid up!');
        });
    });
    // Slide toggle an element with a specified duration
    $('#slideToggleButton').click(function() {
        $('#slideToggleElement').slideToggle(1000);
    });
});
```

10. How can you append new content to an HTML element using jQuery?

You can use the .append() method in jQuery to add new content, such as text, HTML, or other elements, to the end of a selected element.

```
$(document).ready(function() {
    $('#targetElement').append('<p>New content added using
jQuery.</p>');
});
```

11. How do you change the text content of an HTML element using jQuery?

→ To change the text content of an HTML element, you can use the **.text()** method in jQuery.

```
$(document).ready(function() {
    $('#myElement').text('New text content using jQuery.');
});
```

12. How can you modify the attributes of an HTML element using jQuery?

→ You can use the **.attr()** method in jQuery to modify the attributes of an HTML element. This method allows you to get or set attributes like **src, href, class, and more.**

```
$(document).ready(function() {
    // Get the value of an attribute
    var linkHref = $('#myLink').attr('href');
    console.log('Current link href:', linkHref);

    // Set a new value for an attribute
    $('#myLink').attr('href', 'https://www.example.com');
});
```

13. How do you retrieve the value of an HTML attribute using jQuery?

→ You can use the .**attr()** method in jQuery to retrieve the value of an HTML attribute from a selected element.

```
$(document).ready(function() {
    var imageUrl = $('#myImage').attr('src');
    console.log('Image source:', imageUrl);
});
```

14. How do you remove an HTML attribute using jQuery?

→ To remove an HTML attribute from a selected element, you can use the .removeAttr() method in jQuery.

```
$(document).ready(function() {
    $('#myButton').removeAttr('disabled');
});
```

15. How can you animate changes in an element's dimensions using jQuery?

→ You can use the **.animate()** method in jQuery to create animations, including changing the dimensions (width and height) of an element.

```
$("button").click(function(){
    $("div").animate({
      left: '250px',
      opacity: '0.5',
      height: '150px',
      width: '150px'
    });
  });
```

16. Is it possible to manipulate ALL CSS properties with the animate() method?

→ Yes, almost! However, there is one important thing to remember: all property names must be camel-cased when used with the animate() method: You will need to write paddingLeft instead of padding-left, marginRight instead of margin-right, and so on.

→ Also, color animation is not included in the core jQuery library.

17. How can you adjust an element's dimensions based on its current dimensions using jQuery?

→ You can use the .width() and .height() methods with functions as arguments to adjust an element's dimensions based on its current dimensions.

```
$(document).ready(function(){
    $('button').click(function() {
      $('div').width(function(index, currentWidth) {
          return currentWidth * 1.5; // Increase width by 50%
      });

      $('div').height(function(index, currentHeight) {
          return currentHeight * 0.8; // Reduce height by 20%
      });
  });
});
```

18. How can you find the parent element of a selected element using jQuery?

→ You can use the .**parent()** method in jQuery to traverse up the DOM and select the direct parent element of a selected element.

```
let parentElement = $('#childElement').parent();
parentElement.css('border', '1px solid red');
```

19. How do you find all the child elements of a selected element using jQuery?

→ You can use the .**children**() method in jQuery to select all the direct child elements of a selected element.

```
$(document).ready(function() {
    var childElements = $('#parentElement').children();
    childElements.css('background-color', 'yellow');
});
```

20. How can you traverse to the siblings of a selected element using jQuery?

→ You can use the .**siblings**() method in jQuery to select all the siblings (elements with the same parent) of a selected element.

```
$(document).ready(function() {
    var siblingElements = $('#selectedElement').siblings();
    siblingElements.css('font-weight', 'bold');
});
```

→ jQuery provides various methods for traversing DOM elements, such as .parent(), .children(), .siblings(), .next(), .prev(), and more.

21. What is the jQuery .ajax() method used for?

→ The .**ajax**() method in jQuery is used to perform asynchronous HTTP requests to a server. It provides a way to send and receive data from a server without requiring a page refresh, making it ideal for creating dynamic and interactive web applications.

```
$.ajax({
    url: 'https://api.example.com/data',
    method: 'GET',
    success: function(response) {
        console.log('Data received:', response);
    },
    error: function(xhr, status, error) {
        console.error('Error:', error);
    }
});
```

22. How do you send data to the server using the jQuery .ajax() method?

→ You can send data to the server using the data option in the **.ajax()** method. This is commonly used with HTTP methods like POST.

```
$.ajax({
    url: 'https://api.example.com/submit',
    method: 'POST',
    data: {
        username: 'john_doe',
        email: 'john@example.com'
    },
    success: function(response) {
        console.log('Data submitted:', response);
    },
    error: function(xhr, status, error) {
        console.error('Error:', error);
    }
});
```

23. How can you set headers and handle cross-origin requests using the jQuery .ajax() method?

→ You can set custom headers and handle cross-origin requests using the headers option in the .**ajax**() method. This is useful for sending authorization tokens, handling **CORS** (Cross-Origin Resource Sharing), and more.

```
$.ajax({
    url: 'https://api.example.com/data',
    method: 'GET',
    headers: {
        'Authorization': 'Bearer your_token'
    },
    crossDomain: true,
    success: function(response) {
        console.log('Data received:', response);
    },
    error: function(xhr, status, error) {
        console.error('Error:', error);
    }
});
```

24. What is the purpose of the jQuery .get() method?

→ The **.get()** method in jQuery is used to send an HTTP GET request to a server and retrieve data from it. It simplifies the process of making asynchronous requests and handling the responses.

```
$.get('https://api.example.com/data', function(response) {
    console.log('Data received:', response);
});
```

25. How can you pass query parameters with the jQuery .get() method?

→ You can pass query parameters to the server by including them in the URL of the .**get**() method. Query parameters are used to send additional information to the server.

```
var userId = 123;
$.get('https://api.example.com/user', { id: userId },
function(response) {
    console.log('User data:', response);
```

26. How do you handle errors when using the jQuery .get() method?

→ The **.get()** method provides an optional second argument for handling errors using a callback function. This function is executed if the request encounters an error.

```
$.get('https://api.example.com/data', function(response) {
    console.log('Data received:', response);
}, function(error) {
    console.error('Error:', error.statusText);
});
```

27. What is the purpose of the jQuery .post() method?

→ The **.post()** method in jQuery is used to send an HTTP POST request to a server to submit data and retrieve a response. It's commonly used when you need to send data to the server for processing or database operations.

```
$.post('https://api.example.com/submit', { username: 'john_doe',
email: 'john@example.com' }, function(response) {
    console.log('Response:', response);
});
```

28. How can you send JSON data using the jQuery .post() method?

→ You can send JSON data using the **.post()** method by setting the appropriate headers and specifying the data format as JSON.

```
var jsonData = {
    name: 'Alice',
    age: 28,
    city: 'New York'
};

$.post('https://api.example.com/submit', JSON.stringify(jsonData),
function(response) {
    console.log('Response:', response);
}, 'json');
```

29. How can you handle errors when using the jQuery .post() method?

→ Similar to other jQuery AJAX methods, the **.post()** method provides an optional third argument to handle errors using a callback function.

```
$.post('https://api.example.com/submit', { data: 'value' },
function(response) {
    console.log('Response:', response);
}, 'json')
.fail(function(error) {
    console.error('Error:', error.statusText);
});
```

30. What is the purpose of the jQuery .load() method?

→ The **.load()** method in jQuery is used to fetch HTML content from a URL and insert it into a selected element on a web page. It's a convenient way to load remote content and update parts of a page without a full page reload.

```
$('#result').load('https://api.example.com/data #content');
```

31. **How can you pass data to the server using the jQuery .load() method?**

→ You can pass data to the server while using the **.load()** method by including data parameters in the URL or by specifying data using the second argument.

```
$('#result').load('https://api.example.com/data',
{ type: 'json', limit: 10 });
```

32. **How can you execute a callback function after loading content using the jQuery .load() method?**

→ You can provide a callback function as the second argument to the **.load()** method. This function will be executed after the content is loaded and inserted into the selected element.

```
$('#result').load('https://api.example.com/data', function(response,
status, xhr) {
    if (status === 'success') {
        console.log('Content loaded successfully.');
    } else {
        console.error('Failed to load content.');
    }
});
```