Unit 2: HTML5, JQuery and Ajax

AJAX load() Method

• The load() method loads data from a server and puts the returned data into the selected element.

Syntax: $(selector).load(URL, data, callback);

- The required URL parameter specifies the URL you wish to load.
- The optional data parameter specifies a set of query string key/value pairs to send along with the request.
- The optional callback parameter is the name of a function to be executed after the load() method is completed The following example loads the content of the file "test.txt" into a specific element:

$("#div1").load("test.txt");

It is also possible to add a jQuery selector to the URL parameter. The following example loads the content of the element with id="p1", inside the file "test.txt", into a specific element:

$("#div1").load("test.txt #p1");

The optional callback parameter specifies a callback function to run when the load() method is completed. The callback function can have different parameters:

- responseTxt - contains the resulting content if the call succeed
- statusTXT - contains the status of the call
- xhr - contains the XMLHttpRequest object

The following example displays an alert box after the load() method completes. If the load() method has succeed, it displays "External content loaded successfully!", and if it fails it displays an error message:

```
$("button").click(function(){
$("#div1").load("demo_test.txt",function(responseTxt,statusTxt,xhr){
if(statusTxt=="success

alert("External content loaded successfully!");

 if(statusTxt=="error")

  alert("Error: "+xhr.status+": "+xhr.statusText);  });

});
```

HTTP Request: GET vs. POST

Two commonly used methods for a request-response between a client and server.

- GET- Requests data from a specified resource
- POST - Submits data to be processed to a specified resource

- **$.get(URL,callback);**

  - The required URL parameter specifies the URL you wish to request.
  - The optional callback parameter is the name of a function to be executed if the request succeeds.

- **$.post(URL,data,callback);**

  - The required URL parameter specifies the URL you wish to request.

- The optional data parameter specifies some data to send along with the request.
- The optional callback parameter is the name of a function to be executed if the request succeeds.

Example

```
$("button").click(function(){

$.post("test_post.jsp",

{

name:“Bill Gates",

city:“Seatle"

},

function(data,status)

{

alert("Data: " + data + "\nStatus: " + status);

});

});
```

- The first parameter of $.post() is the URL we wish to request ("test_post.jsp").
- Then we pass in some data to send along with the request (name and city).

- The JSP script in "test_post.jsp" reads the parameters, process them, and return a result.
- The third parameter is a callback function. The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

The $.ajax(settings) or $.ajax(url, settings)

Used for sending an Ajax request. The settings is an object of key-value pairs. The frequently-used keys are:

- *url*: The request URL, which can be placed outside the *settings* in the latter form.
- *type*: GET or POST.
- *data*: Request parameters (name=value pairs). Can be expressed as an object (e.g., {name:"peter", msg:"hello"}), or query string (e.g., "name=peter&msg=hello").
- *dataType*: Expected response data type, such as text, xml, json, script or html.
- *headers*: an object for request header key-value pairs. The header X-Requested-With:XMLHttpRequest is always added.

Ajax request, by default, is asynchronous. In other words, once the .ajax() is issued, the script will not wait for the response, but continue into the next statement, so as not to lock up and freeze the screen.

NOTE: $ is a shorthand (alias) for the jQuery object. $() is an alias for jQuery() function for Selector. $.ajax() is a global function (similar to class method in an OO language).

The Fetch API

The Fetch API provides a fetch() method defined on the window object, which you can use to perform requests. This method returns a Promise that can be used to retrieve the response of the request.

The fetch method only has one mandatory argument, which is the URL of the resource you wish to fetch. The Fetch API is a modern interface that allows you to make HTTP requests in the web browsers. The fetch() method is available in the global scope that instructs the browser to send a request to a provided URL.

Sending a Request

The fetch() has only one parameter which most of the time is the URL of the resource that you want to fetch:

```
let response = fetch(url);
```

The fetch() method returns a Promise so you can use the then() and catch() methods to handle it:

```
fetch(url)
  .then(response => {
    // handle the response
  })
  .catch(error => {
    // handle the error
  });
```

When the request completes, the resource is available. At this time, the promise will resolve into a Response object.

The Response object is the API wrapper for the fetched resource. The Response object has a number of useful properties and methods to inspect the response.

Reading a Response

If the contents of the response are in the raw text format, you can use the text() method. The text() method returns a Promise that resolves with the complete contents of the fetched resource:

```
fetch('/readme.txt')
   .then(response => response.text())
   .then(data => console.log(data));
```

Or async/await can be used:

```
async function fetchText() {
   let response = await fetch('/readme.txt');
   let data = await response.text();
   console.log(data);
}
```

Besides the text() method, the Response object has other methods such as json(), blob(), formData() and arrayBuffer() to handle respective data.

Handling status codes of a response

The Response object provides the status code and status text via the status and statusText properties. When a request is successful, the status code is 200 and status text is OK:

```
async function fetchText() {
   let response = await fetch('/readme.txt');
```

```
    console.log(response.status); // 200

    console.log(response.statusText); // OK

    if (response.status === 200) {

        let data = await response.text();

        // handle data

    }

}

fetchText();
```

Output:

200

OK

If the requested resource doesn't exist, the response code is 404:

```
let response = await fetch('/non-existence.txt');

console.log(response.status); // 400

console.log(response.statusText); // OK
```

Output:

400

Not Found

If the requested URL throws a server error, the response code will be 500.

If the requested URL is redirected to the new one with the response 300-309, the status of the Response object is set to 200. In addition the redirected property is set to true.

The fetch() returns a promise that rejects when a real failure occurs such as a web browser timeout, a loss of network connection, and a CORS violation.