AJAX and JQUERY

Course Code: CSC 3222 Course Title: Web Technologies

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**12**	**Week No:**	**12**	**Semester:**	**Summer 19-20**
Lecturer:					

Lecture Outline

1. Introduction to AJAX
2. AJAX Request and Response
3. Access JSON using AJAX
4. Introduction to jQuery
5. jQuery Selectors
6. jQuery Events

Introduction to AJAX

What is AJAX

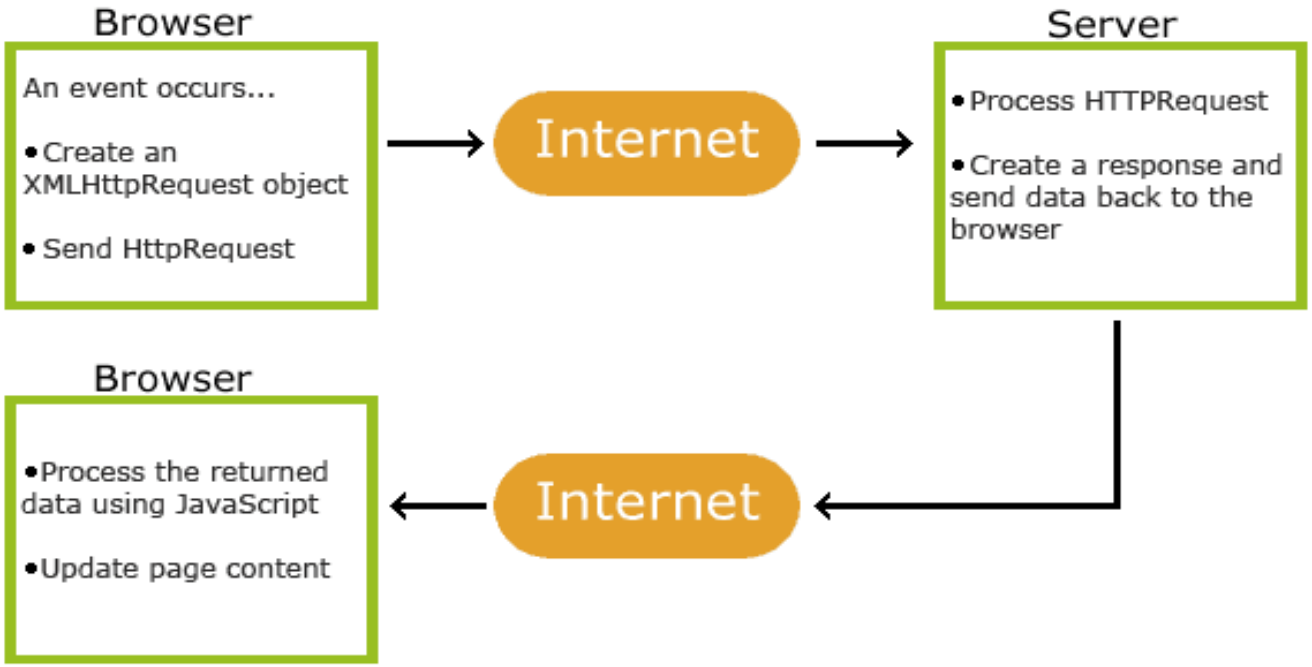
AJAX = Asynchronous JavaScript And XML.

- AJAX is not a programming language.
- AJAX is a technique for accessing web servers from a web page.
- A browser built-in XMLHttpRequest object to request data from a web server
  - Read data from a web server - after the page has loaded
  - Update a web page without reloading the page
  - Send data to a web server - in the background
- JavaScript and HTML DOM to display or use the data

Advantages of AJAX

- AJAX is the most viable Rich Internet Application (RIA) technology so far. It is getting tremendous industry momentum and several tool kit and frameworks are emerging. But at the same time, AJAX has browser incompatibility and it is supported by JavaScript, which is hard to maintain and debug. AJAX is based on the following open standards
- Browser-based presentation using HTML and Cascading Style Sheets (CSS)
- Data is stored in XML format and fetched from the server.
- Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.
- JavaScript to make everything happen.

How AJAX Works



1. An event occurs in a web page when the page is loaded, a button is clicked
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action like page update is performed by JavaScript

Example

- The HTML page contains a section and a <button>.
- The section is used to display information from a server.
- The <button> calls a function loadDoc().
- The function requests data from a web server and displays it.

```
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();
}
```

The XMLHttpRequest Object

- The keystone of AJAX is the XMLHttpRequest object.
- The XMLHttpRequest object can be used to exchange data with a web server behind the scenes.
- it is possible to update parts of a web page, without reloading the whole page.

Syntax for creating an XMLHttpRequest object:
[Variable] = new XMLHttpRequest();
var xhttp = new XMLHttpRequest();

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 Request

- To send a request to a server, 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

- simple GET request:

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

To get some specific data using GET method, need to use below example

Example 1:

```
xhttp.open("GET", "demo_get.asp?t=" + Math.random(), true);
xhttp.send();
```

Example 2:

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

POST Requests

simple **POST** request

```
xhttp.open("POST", "demo_post.asp", true);
xhttp.send();
```

To POST data like an HTML form, add an HTTP header with setRequestHeader().

To get specific data using POST method need to use

```
xhttp.open("POST", "ajax_test.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

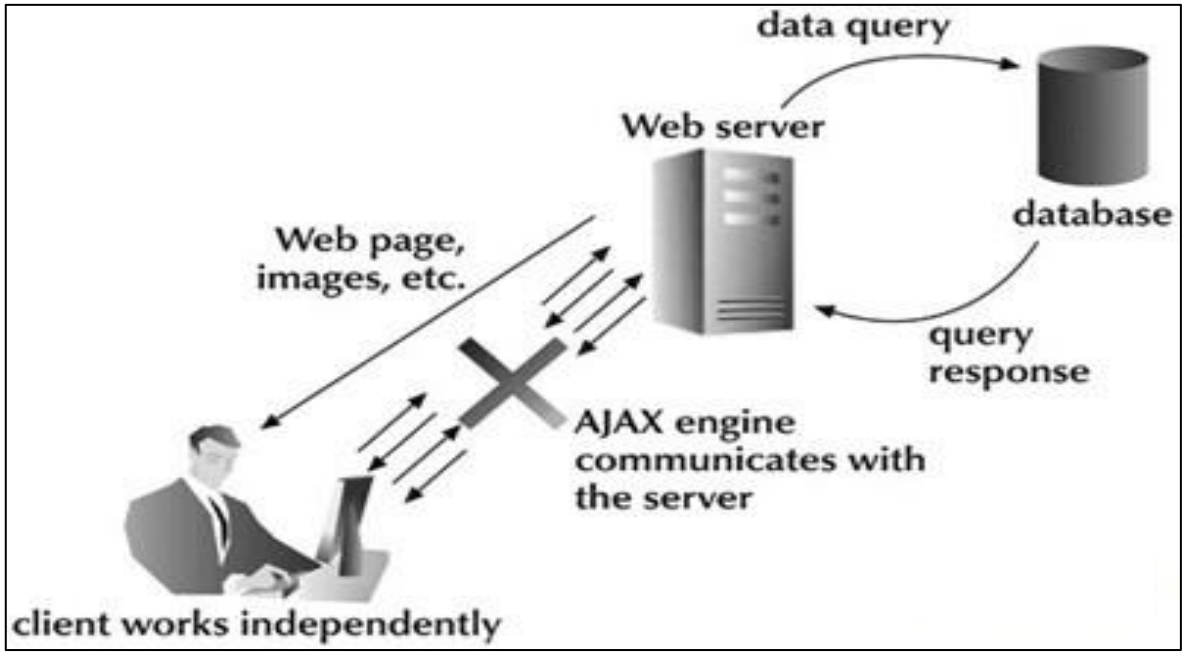
Method	Description
setRequestHeader(*header, value*)	Adds HTTP headers to the request *header*: specifies the header name *value*: specifies the header value

Asynchronous Requests

Server requests should be sent asynchronously.The async parameter of the open() method should be set to true:
xhttp.open("GET", "ajax_test.asp", true);

By sending asynchronously, the JavaScript does not have to wait for the server response.

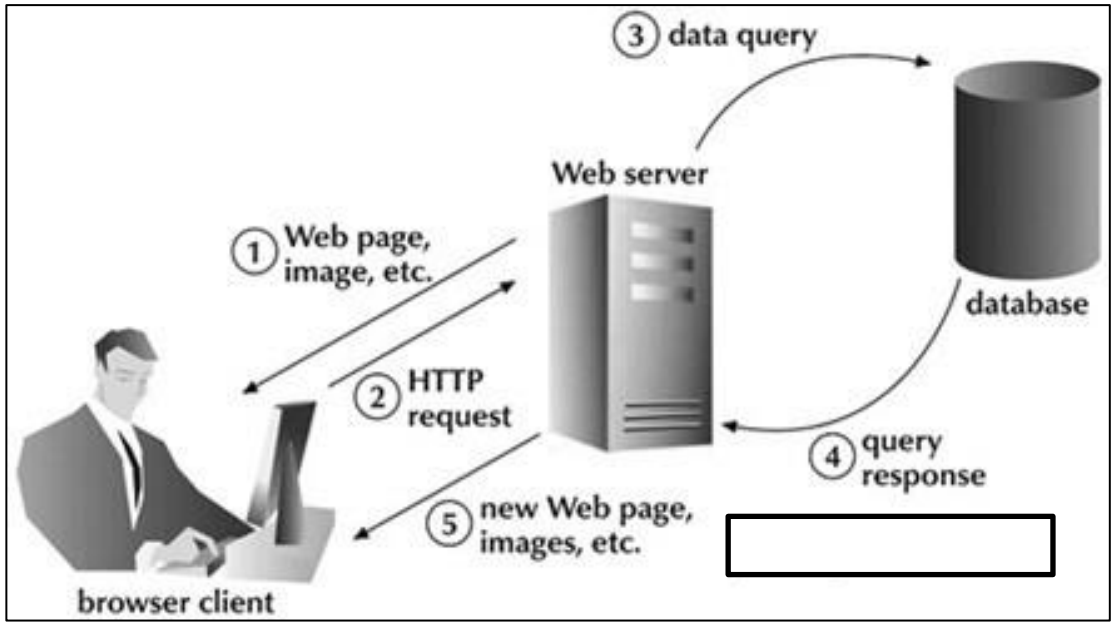
- execute other scripts while waiting for server response
- deal with the response after the response is ready



Synchronous Requests

A synchronous request blocks the client until operation completes i.e. browser is unresponsive.
To execute a synchronous request, change the third parameter in the open() method to false:
xhttp.open("GET", "ajax_info.txt", false);

Sometimes async = false are used for quick testing.



AJAX Response

- The readyState property holds the status of the XMLHttpRequest.
- The onreadystatechange property defines a function to be executed when the readyState changes.
- The status property and the statusText property holds the status of the XMLHttpRequest object.
- The onreadystatechange function is called every time the readyState changes.
- When readyState is 4 and status is 200, the response is ready:

```
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();
}
```

AJAX Response

Server Response Methods

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

Server Response Properties

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

The Flow of an AJAX Request

Event Happens (User Action)

User clicks a button, types in a box, scrolls, etc.

Example: Typing in Google's search box.

JavaScript Creates a Request

Browser (via JS) creates an AJAX request object.

Old way: XMLHttpRequest

Modern way: fetch()

Request Sent to the Server

The request contains:

- Method: GET (read data) or POST (send data)
- URL: where the request goes (e.g., server.php)
- Data: what you send (like a search term, form input)

Server Processes the Request

The server (PHP, Node.js, Python, etc.) receives the request.

It can read from a database, file, or generate some response.

Server Sends a Response Back

Response can be in:

- Text (plain)
- HTML (markup)
- JSON (most common today)
- XML (older style)

JavaScript Handles the Response

JS receives the data (via .responseText or response.json()).

Updates only part of the web page (like , <table>, etc.), without refreshing.

Introduction to JSON

- JSON: JavaScript Object Notation.
- JSON is a syntax for storing and exchanging data.
- JSON is text, written with JavaScript object notation.

Sending Data

```
var myObj = {name: "John", age: 31, city: "New York"};
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

Receiving Data

```
var myJSON = '{"name":"John", "age":31, "city":"New York"}';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

JSON with AJAX

```
<script>
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myArr = JSON.parse(this.responseText);
        myFunction(myArr);
    }
};
xmlhttp.open("GET", myTutorials.txt, true);
xmlhttp.send();

function myFunction(arr) {
    var out = "";
    var i;
    for(i = 0; i < arr.length; i++) {
        out += '<a href="' + arr[i].url + '">' +
        arr[i].display + '</a><br>';
    }
    document.getElementById("id01").innerHTML = out;
}
</script>
```

Example of JSON

myTutorials.txt

```
[
{
"display": "JavaScript Tutorial",
"url": "https://www.w3schools.com/js/default.asp"
},
{
"display": "HTML Tutorial",
"url": "https://www.w3schools.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "https://www.w3schools.com/css/default.asp"
}
]
```

Introduction to jQuery

What is jQuery?

- The purpose of jQuery is to make it much easier to use JavaScript.
- jQuery is a lightweight JavaScript library.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish and wraps them into methods that you can call with a single line of code.
- jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

jQuery

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

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

Example

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

$(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)

$(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".

jQuery Selectors

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

The element Selector

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

To select all <p> elements on a page $("p")

Example

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

When a user clicks on a button, all <p> elements will be hidden:

The #id Selector

- jQuery #id selector uses the id attribute of an HTML tag to find the specific element.
- An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.
- To find an element with a specific id, write a hash character, followed by the id of the HTML element $("#test")

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

When a user clicks on a button, the element with id="test" will be hidden.

The .class Selector

- jQuery #id selector uses the id attribute of an HTML tag to find the specific element.
- An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.
- To find an element with a specific id, write a hash character, followed by the id of the HTML element $("#test")

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

When a user clicks on a button, the element with id="test" will be hidden.

The .class Selector

- The jQuery .class selector finds elements with a specific class.
- To find elements with a specific class, write a period character, followed by the name of the class $(".test")

Example

```
$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide();
  });
});
```

When a user clicks on a button, the elements with class="test" will be hidden:

Syntax	Description
$("*")	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"

jQuery Event Methods

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

Examples:

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

The term "fires/fired" is often used with events. Example: "The keypress event is fired, the moment you press a key".

jQuery Event Methods

some common DOM events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

click()

- The click() method attaches an event handler function to an HTML element.
- The function is executed when the user clicks on the HTML element.

The following example says: When a click event fires on a <p> element; hide the current <p> element

```
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
```

dblclick()

- The dblclick() method attaches an event handler function to an HTML element.
- The function is executed when the user double-clicks on the HTML element

```
$(document).ready(function(){
  $("p").dblclick(function(){
    $(this).hide();
  });
});
```

mouseenter()

- The mouseenter() method attaches an event handler function to an HTML element.
- The function is executed when the mouse pointer enters the HTML element

```
$(document).ready(function(){
  $("#p1").mouseenter(function(){
    alert("You entered p1!");
  });
});
```

mouseleave()

- The mouseleave() method attaches an event handler function to an HTML element.
- The function is executed when the mouse pointer leaves the HTML element

```
$(document).ready(function(){
  $("#p1").mouseleave(function(){
    alert("Bye! You now leave p1!");
  });
});
```

Output

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
jQuery Tutorial
SQL Tutorial
PHP Tutorial
XML Tutorial

References

- MySQL - www.mysql.com
- W3Schools Online Web Tutorials- www.w3schools.com
- PHP Manual - www.php.net
- Free Online Tutorials - www.javatpoint.com

Books

- Sams Teach Yourself Ajax JavaScript and PHP All in One; Phil Ballard and Michael Moncur;
- Sams Publishing; 2010
- JavaScript Phrasebook; Christian Wenz; Sams Publishing; 2007
- PHP and MySQL Web Development, 4/E; Luke Welling and Laura Thomson; AddisonWesley Professional; 2009
- JavaScript for Programmers Paul J. Deitel and Harvey M. Deitel; Prentice Hall; 2009