Unit 2: HTML5, JQuery and Ajax

Asynchronous Communication- XHR

AJAX is not a programming language, but a technique that incorporates a client-side script (i.e. a script that runs in a user's browser) that communicates with a web server. Further, its name is somewhat misleading: while an AJAX application might use XML to send data, it could also use just plain text or JSON text. But generally, it uses an XMLHttpRequest object in your browser to request data from the server and JavaScript to display the data.

XMLHttpRequest supports both synchronous and asynchronous communications. In general, however, asynchronous requests should be preferred to synchronous requests for performance reasons.

Synchronous requests block the execution of code which causes "freezing" on the screen and an unresponsive user experience.

Fetch API

The Fetch API provides an interface for fetching resources. It will seem familiar to anyone who has used XMLHTTPRequest, but this API provides a more powerful and flexible feature set.

AJAX can access the server both synchronously and asynchronously:

- **Synchronously**, in which the script stops and waits for the server to send back a reply before continuing.
- **Asynchronously**, in which the script allows the page to continue to be processed and handles the reply if and when it arrives.

Components of AJAX

The AJAX cannot work independently. It is used in combination with other technologies to create interactive Web pages that are described in the following list:

JavaScript:

- Loosely typed scripting language.
- JavaScript function is called when an event occurs in a page.
- Glue for the whole AJAX operation.

DOM:

- API for accessing and manipulating structured documents.
- Represents the structure of XML and HTML documents.

CSS:

- Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript.

XMLHttpRequest:

- JavaScript object that performs asynchronous interaction with the server.

XMLHttpRequest Methods

- **abort()**

  Cancels the current request.

- **getAllResponseHeaders()**

  Returns the complete set of HTTP headers as a string.

- **getResponseHeader(headerName)**

Returns the value of the specified HTTP header.

- **open(method, URL)**
- **open(method, URL, async)**
- **open(method, URL, async, userName)**
- **open(method, URL, async, userName, password)**

  Specifies the method, URL, and other optional attributes of a request.

  The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.

  The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.

- **send(content)**

  Sends the request.

- **setRequestHeader(label, value)**

  Adds a label/value pair to the HTTP header to be sent.

XMLHttpRequest Properties

- **onreadystatechange**

  An event handler for an event that fires at every state change.

- **readyState**

  The readyState property defines the current state of the XMLHttpRequest object.

The following table provides a list of the possible values for the readyState property −

State	Description
0	The request is not initialized.
1	The request has been set up.
2	The request has been sent.
3	The request is in process.
4	The request is completed.

readyState = 0 After you have created the XMLHttpRequest object, but before you have called the open() method.

readyState = 1 After you have called the open() method, but before you have called send().

readyState = 2 After you have called send().

readyState = 3 After the browser has established a communication with the server, but before the server has completed the response.

readyState = 4 After the request has been completed, and the response data has been completely received from the server.

- **responseText**

  Returns the response as a string.

- **responseXML**

  Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.

- **status**

  Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").

- **statusText**

  Returns the status as a string (e.g., "Not Found" or "OK").

How to choose between GET & POST?

Purpose of GET - to GET information , intended to be used when you are reading information to display on the page. Browsers will automatically cache the result from a GET request and if the same GET request is made again then they will display the cached result rather than rerunning the entire request. A GET call is retrieving data to display in the page and data is not expected to be changed on the server by such a call and so re-requesting the same data should be expected to obtain the same result.

POST method is intended to be used where you are updating information on the server . Results returned from server . A POST call will therefore always obtain the response from the server rather than keeping a cached copy of the prior response. If the value to be retrieved is expected to vary over time as a result of other processes updating it then add a current time parameter to what you are passing in your GET call. These criteria is not only for GET and POST for your Ajax calls but also to GET or POST when processing forms on your web page as well.