AJAX Fundamentals

.NET

Ajax (Asynchronous JavaScript and XML) is a set of web development techniques using many web technologies on the client-side to create asynchronous web applications.

HTTPS://EN.WIKIPEDIA.ORG/WIKI/AJAX_(PROGRAMMING)

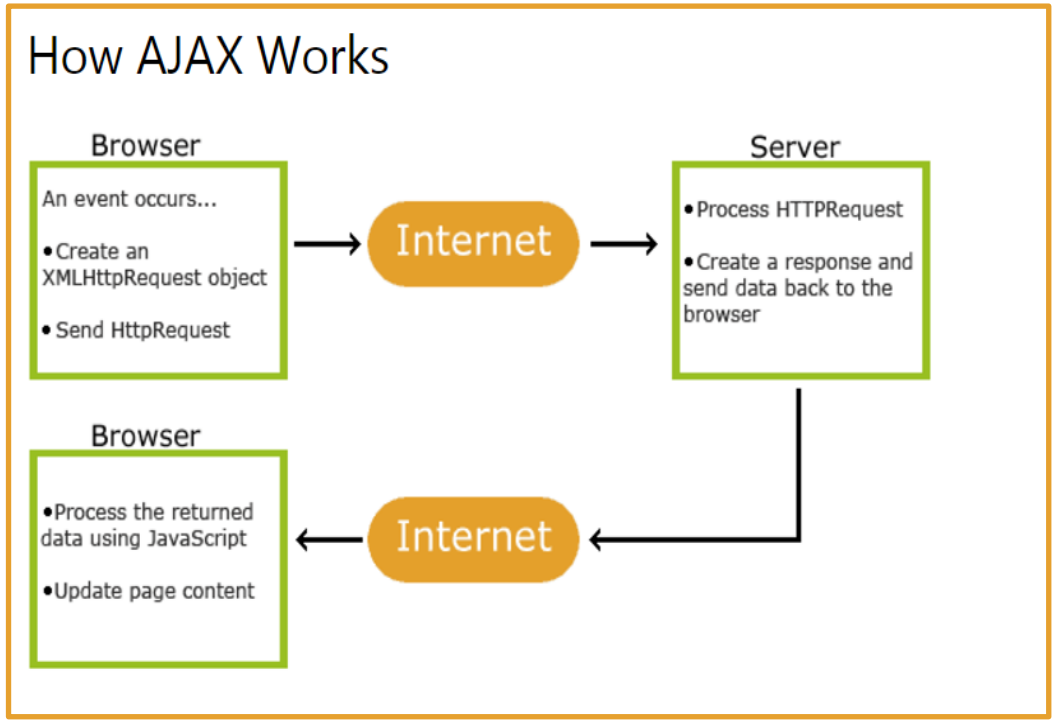
AJAX – Overview

https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started
https://www.w3schools.com/xml/ajax_intro.asp
https://www.tutorialspoint.com/ajax/ajax_quick_guide.htm

AJAX (***Asynchronous JavaScript And XML***) is the use of the JavaScript ***XMLHttpRequest*** object to communicate with servers.

A JS ***XMLHttpRequest*** object can send and receive information in various formats (JSON, XML, HTML, text). It has many built-in functions to help send and receive data.

Because ***AJAX*** is "asynchronous" it can communicate with the server, exchange data, and update a page without having to refresh.



AJAX – XmlHttpRequest vs. Fetch

https://medium.com/beginners-guide-to-mobile-web-development/the-fetch-api-2c962591f5c

XMLHttpRequest.readyState

https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/readyState

---

The XMLHttpRequest.readyState property returns the state that an XMLHttpRequest client is currently in. An ***XMLHttpRequest*** exists in one of four states during the HTTP request/response cycle:

Value	State	Description
0	UNSENT	Client has been created. open() not called yet.
1	OPENED	open() has been called.
2	HEADERS_RECEIVED	send() has been called, and headers and status are available.
3	LOADING	Downloading; ***.responseText*** holds partial data.
4	DONE	The operation is complete.

Checking Readystate Status

https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/readyState

Readystate can be verified by checking its integer value.

```
if(req.readyState < 4) {
    //handle problem like check the  status code, log it, retry, etc.
}
else if(req.readyState === 4){
    //success! Do something.
}
```

```
var xhr = new XMLHttpRequest();
console.log('UNSENT', xhr.readyState); // readyState will be 0

xhr.open('GET', '/api', true);
console.log('OPENED', xhr.readyState); // readyState will be 1

xhr.onprogress = function () {
    console.log('LOADING', xhr.readyState); // readyState will be 3
};

xhr.onload = function () {
    console.log('DONE', xhr.readyState); // readyState will be 4
};

xhr.send(null);
```

AJAX Step-By-Step (1/4)

https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started
https://www.tutorialspoint.com/ajax/ajax_quick_guide.htm

---

1. In order to make an HTTP request to the server with JavaScript, you need an XMLHttpRequest object.

```
let httpRequest = new XMLHttpRequest();
```

2. The response indicates a ‘state change’ of the object. You need to give the XMLHttpRequest object a JS function that will handle the response. Set the .onreadystatechange property of the XMLHttpRequest object equal to the function to call (without parenthesis) when the response is received.

```
httpRequest.onreadystatechange = CallbackFunction;
```

AJAX Step-By-Step (2/4)

https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started

---

3. Ready the request with the open() function of the XMLHttpRequest object. The first parameter of open() is the ***HTTP request method*** (all caps). The second parameter is the URL you're sending the request to. The (optional) third parameter sets asynchronicity. It's true by default.

```
httpRequest.open('GET', 'http://www.example.org/dir/file', true);
```

4. Send the request. The parameter to the send() method can be any data you want to send to the server. Send data in a format that the server can parse (query string, JSON, XML).

```
httpRequest.send();
```

AJAX Step-By-Step (3/4)

https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started

5. In step 2, we set httpRequest.onreadystatechange to a callback function. .onreadystatechange() is invoked every time the state changes. The first thing that function needs to do is verify that the response was completed.

```
if (httpRequest.readyState == XMLHttpRequest.DONE) {
   // whatever actions you want.
} else {
   // Not ready yet.
}
```

6. You also must check that the HTTP response status code is acceptable.

```
if (httpRequest.status == 200) {
   // NOW you can act on the data received.
} else {
   // There was a problem with the request.
}
```

AJAX Step-By-Step (4/4)

https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started

7. You can now send async requests. You have two ways to access the data:
   - httpRequest.responseText – Get the response as a string.
   - httpRequest.responseXML – Get the response as an XMLDocument object and traverse it with JavaScript DOM functions (see walking the DOM)

Now you can access the data and do what you need with it.