

AJAX

Topics Covered:

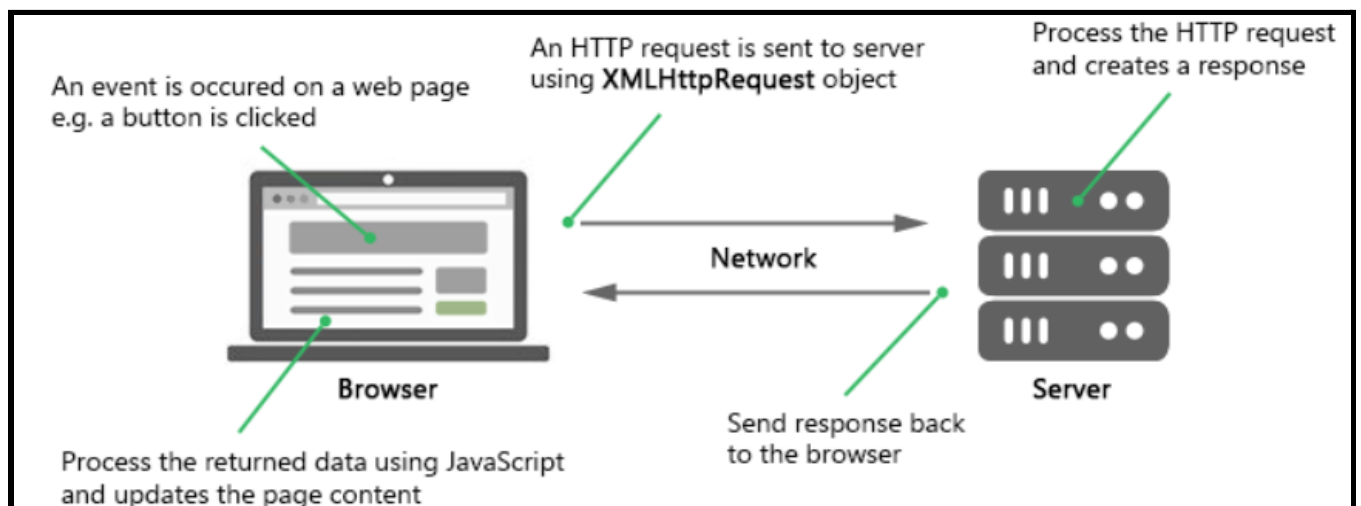
- AJAX Introduction
- XMLHttpRequest
- XMLHttpRequest Response
- AJAX Application
- fetch()

Topics in Detail:

AJAX Introduction

- AJAX stands for **A**synchronous JavaScript **A**nd **X**ML.
- It is a combination of **XMLHttpRequest**, **JavaScript** and **HTML DOM**.
- The web pages are updated **asynchronously** by **exchanging data** with the **web server**.
- Instead of **reloading** the whole web page, **AJAX** helps in updating **parts of a web page**.

How AJAX works



XMLHttpRequest

- To request data from a server, **XMLHttpRequest object** is used.
- Open()** and **send()** methods of XMLHttpRequest object are used to **send a request** to the server.

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Method	Description
open (method, url, async)	Specifies the type of request method - GET or POST url - file location async - true (asynchronous) or false (synchronous)
send()	Sends request to the server(GET)
send (string)	Sends request to the server (POST)

- url - A file on a server**

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

- The file can be of any kind like **.txt**, **.xml** or **server scripting files** like **.asp** and **.php**.

Asynchronous - True or False?

- Asynchronous** parameter of open is set to **true**.
- JavaScript **does not wait** for server **response**, instead
 - Start **executing other scripts** while waiting.
 - After the **response is ready**, deal with the responses.
- The **default** value of async parameter is **true**.
- async = false** is **not recommended**.

GET or POST

- **GET** is used in **most cases**, and it is **simpler** and **faster** than POST.
- **POST** is **more secure** and **robust** than **GET**.
- **POST** is used in the following situations
 - When **sending large amount of data** to the server.
 - When a **cached file** is **not an option**.
 - When sending **user input**.

GET Requests

- A simple GET request.

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

- To **avoid** getting **cached result**, add a **unique ID** to the URL.

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

- To **send information** with the GET method, **add the information to the URL**.

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

POST Requests

- A simple POST request.

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

- The Data-like HTML form are requested by adding an HTTP header with **setRequestHeader()**.
- The data want to be sent is specified in the send() method.

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

Method	Description
setRequestHeader(header, value)	<p>Adds HTTP headers to the request</p> <p>header: specifies the header name value: specifies the header value</p>

Synchronous Request

- **Synchronous request** is achieved by setting the **async** parameter to **false**.
- For quick testing, `async = false` is set.
- The `onreadystatechange` function is not needed as the server waits for server completion.

```
xhttp.open("GET", "ajax_info.txt", false);
xhttp.send();
document.getElementById("demo").innerHTML = xhttp.responseText;
```

XMLHttpRequest Response

Server Response Properties

Property	Description
responseText	Get the response data as a string
responseXML	Get the response data as an XML data

The responseText Property

- The **responseText** property returns the **server response** as a **JavaScript string**.

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

The responseXML Property

- An in-built **XML parser** is there in the XMLHttpRequest **object**.
- The **responseXML** property returns the **server response** as an **XML DOM object**.

```
const xmlDoc = xhttp.responseXML;
const x = xmlDoc.getElementsByTagName("ARTIST");

let txt = "";
for (let i = 0; i < x.length; i++) {
    txt += x[i].childNodes[0].nodeValue + "<br>";
}
document.getElementById("demo").innerHTML = txt;

xhttp.open("GET", "cd_catalog.xml");
xhttp.send();
```

Server Response Methods

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

The getAllResponseHeaders() Method

All the **header information** from the server response is returned by the **getAllResponseHeaders()**.

```
const xhttp = new XMLHttpRequest();
xhttp.onload = function() {
    document.getElementById("demo").innerHTML =
        this.getAllResponseHeaders();
}
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

The `getResponseHeader()` Method

The **specific header information** from the server response is returned by `getResponseHeader()` method.

```
const xhttp = new XMLHttpRequest();
xhttp.onload = function() {
    document.getElementById("demo").innerHTML =
        this.getResponseHeader("Last-Modified");
}
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

XML Applications

Display XML data in an HTML Table

- The values of `<ARTIST>` and `<TITLE>` elements are displayed in an HTML table.

```
<table id="demo"></table>

<script>
function loadXMLDoc() {
    const xhttp = new XMLHttpRequest();
    xhttp.onload = function() {
        const xmlDoc = xhttp.responseXML;
        const cd = xmlDoc.getElementsByTagName("CD");
        myFunction(cd);
    }
    xhttp.open("GET", "cd_catalog.xml");
    xhttp.send();
}

function myFunction(cd) {
    let table="<tr><th>Artist</th><th>Title</th></tr>";
    for (let i = 0; i < cd.length; i++) {
        table += "<tr><td>" +
            cd[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
            "</td><td>" +
            cd[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
            "</td></tr>";
    }
    document.getElementById("demo").innerHTML = table;
}
</script>

</body>
</html>
```

Display the first CD element in an HTML div

```
const xhttp = new XMLHttpRequest();
xhttp.onload = function() {
    const xmlDoc = xhttp.responseXML;
    const cd = xmlDoc.getElementsByTagName("CD");
    myFunction(cd, 0);
}
xhttp.open("GET", "cd_catalog.xml");
xhttp.send();

function myFunction(cd, i) {
    document.getElementById("showCD").innerHTML =
    "Artist: " +
    cd[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
    "<br>Title: " +
    cd[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
    "<br>Year: " +
    cd[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue;
}
```

Navigate between CDs

To navigate between CDs, create next() and previous() functions.

```
function next() {
    // display the next CD, unless you are on the last CD
    if (i < len-1) {
        i++;
        displayCD(i);
    }
}

function previous() {
    // display the previous CD, unless you are on the first CD
    if (i > 0) {
        i--;
        displayCD(i);
    }
}
```

Show Album information when clicking on a CD

To show album information when clicking on a CD, call displayCD() function on the onclick event.

```
function displayCD(i) {  
    document.getElementById("showCD").innerHTML =  
        "Artist: " +  
        cd[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +  
        "<br>Title: " +  
        cd[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +  
        "<br>Year: " +  
        cd[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue;  
}
```

fetch() Method

- The **fetch()** method in JavaScript **sends request to the server** and the fetched **information is being loaded** in the webpages.

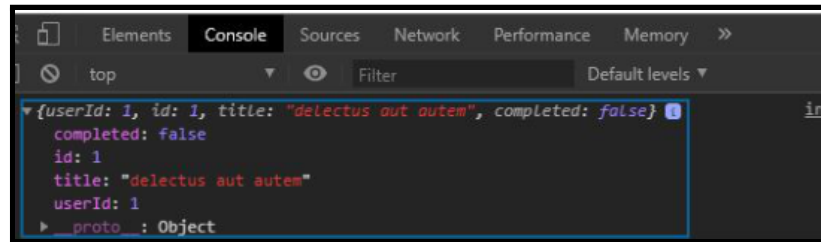
- **Syntax**

```
fetch( url, options )
```

- **url** - url to which the request is to be made.
- **options** - It is an **optional parameter**. It is an array of properties.
- **Returns value** - The return data is of **JSON** or **XML format**. It can be a single object or array of objects.
- fetch() method without options will act as a get request.

```
<script>  
    // API for get requests  
    let fetchRes = fetch(  
        "https://jsonplaceholder.typicode.com/todos/1");  
    // fetchRes is the promise to resolve  
    // it by using.then() method  
    fetchRes.then(res =>  
        res.json()).then(d => {  
            console.log(d)  
        })  
</script>
```


Output



- fetch() method with options given below will act as a post request.

```

<script>
    user = {"name": "Geeks for Geeks",
            "age": "23"}
    // Options to be given as parameter in fetch for making requests
    other than GET
    let options = {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json;charset=utf-8'},
        body: JSON.stringify(user)}
    // Fake api for making post requests
    let fetchRes = fetch(
    "http://dummy.restapiexample.com/api/v1/create",options);
    fetchRes.then(res =>
        res.json()).then(d => {console.log(d)})
</script>
  
```

Output

