03. AJAX Request to Server.

3.1. Send a Request to a Server

The XMLHttpRequest object is used to exchange data with a server.

To send a request to a server, we use the 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)

3.2. GET or POST?

GET is simpler and faster than POST, and can be used in most cases.

However, always use POST requests when:

- A cached file is not an option (update a file or database on the server).
- Sending a large amount of data to the server (POST has no size limitations).
- Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

3.3. GET Requests

A simple GET request:

Example:

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

In the example above, you may get a cached result. To avoid this, add a unique ID to the URL:

Example:

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

If you want to send information with the GET method, add the information to the URL:

Example:

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

3.4. POST Requests

A simple POST request:

Example:

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

To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method:

Example:

```
xhttp.open("POST", "ajax_test.php", 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

3.5. The URL – A File On a Server

The url parameter of the open() method, is an address to a file on a server:

Example:

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

The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and .php (which can perform actions on the server before sending the response back).

3.6. Asynchronous - True or False?

Server requests should be sent asynchronously.

The async parameter of the open() method should be set to true:

Example:

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

By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead:

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

3.7. The onreadystatechange Property

With the XMLHttpRequest object you can define a function to be executed when the request receives an answer.

The function is defined in the **onreadystatechange** property of the XMLHttpResponse object:

Example:

```
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();
```

You will learn more about onreadystatechange in a later chapter.

3.8. Synchronous Request

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. You will also find synchronous requests in older JavaScript code.

Since the code will wait for server completion, there is no need for an onreadystatechange function:

Example:

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

Synchronous XMLHttpRequest (async = false) is not recommended because the JavaScript will stop executing until the server response is ready. If the server is busy or slow, the application will hang or stop.

Synchronous XMLHttpRequest is in the process of being removed from the web standard, but this process can take many years.

Modern developer tools are encouraged to warn about using synchronous requests and may throw an ***InvalidAccessError*** exception when it occurs.