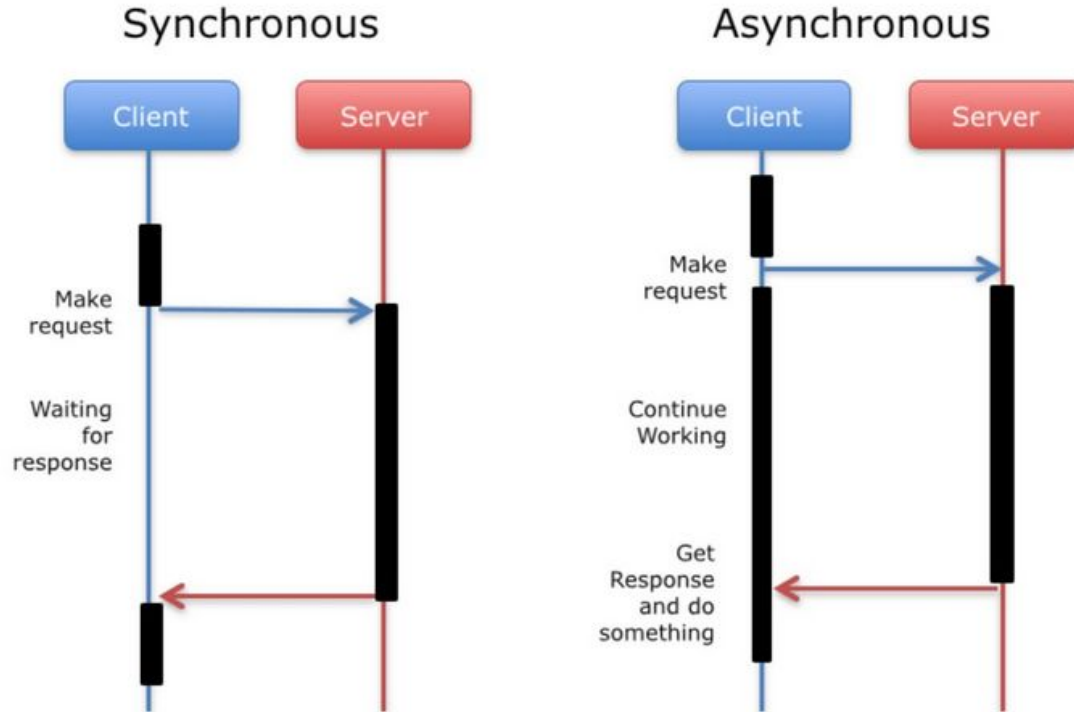# AJAX

AJAX

For the European soccer fans, it is not not Ajax Amsterdam, it is **Asynchronous JavaScript and Xml.**

Do you know what asynchronous means?

Synchronous - Asynchronous

Synchronous - Asynchronous

Asynchronous: It is is a technique that enables your program to start a potentially long-running task and still be able to be responsive to other events while that task runs, rather than having to wait until that task has finished. Once that task has finished, your program is presented with the result.

Synchronous: As the name suggests synchronous means to be in a sequence, i.e. every statement of the code gets executed one by one. So, basically a statement has to wait for the earlier statement to get executed.

Synchronous Example

First let's setup the color blue on a –primary-color:

document.documentElement.style.setProperty('--primary-color', 'blue')

console.log('I am starting synchronous, and I am going to wait until you click on the alert')
alert('Hey, the code is blocking right now')
console.log("ok I am back working here")
document.documentElement.style.setProperty('--primary-color', 'purple')

The alert box was blocking the code, you did not see the console log 'ok I am back….'.

Let's set the color blue again:
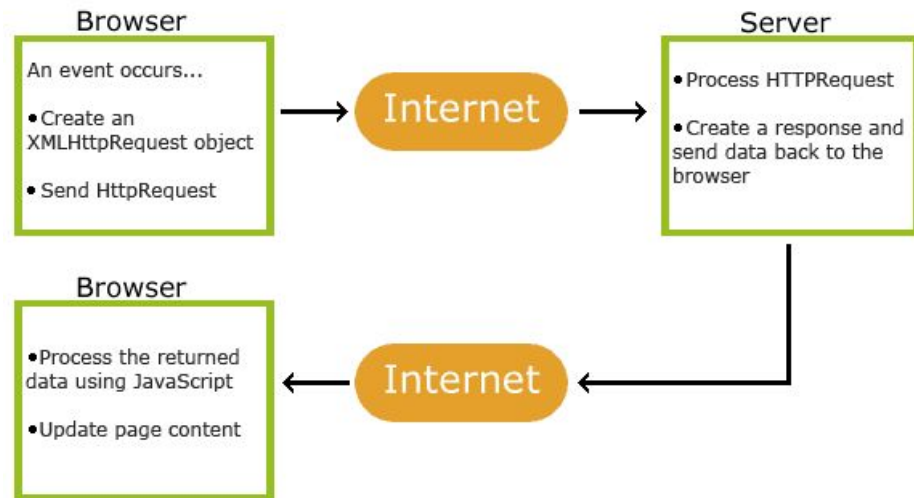document.documentElement.style.setProperty('--primary-color', 'blue')

Asynchronous - Example

We are going to insert an asynchronous function setTimeout. The setTimeout will wait in the background, and the following line is going to proceed, and when the background timer is done (and when javascript has finished what it had next), then it will proceed with the instruction inside the setTimeout (refer to the picture of the previous page)

```
console.log('I am going to go asynchronous, and I am not going to wait like a benet this time')
setTimeout(function () {
        document.documentElement.style.setProperty('--primary-color', 'purple')
        console.log("it is purple now, what? You logged something in the console while I was waiting")
}, 5000)
console.log("ok I finished here, quicker that the settimeout, now I am going to wait, because I finished first, lalalala")
```

AJAX Steps

1. An event occurs in a web page (the page is loaded, a button is clicked)

2. An XMLHttpRequest object is created by JS

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 JS

7. Proper action (like page update) is performed by JS

Browser
An event occurs...
- Create an XMLHttpRequest object
- Send HttpRequest

Internet

Server
- Process HTTPRequest
- Create a response and send data back to the browser

Browser
- Process the returned data using JavaScript
- Update page content

Internet

XMLHttpRequest Object

It is an object that can be used to exchange data with a server behind the scenes.

It is possible to update parts of a web page without reloading the whole page.

Browsers should have a built-in XMLHttpRequest object.

var xhttp = new XMLHttpRequest();

XMLHttpRequest Object - Methods

getAllResponseHeaders(): It is used to get the header information

getResponseHeader(): It is used to get the specific header information

open(method, url, async, user, psw): It is used to set the request parameters.

       <u>method:</u> request type GET or POST
       <u>url:</u> file location for example
       <u>async</u>: for the asynchronous set to true or for synchronous set to false. It is true by default.

send(): It is used to send requests to the web server. It is generally used for GET requests

send(string): It is used to send requests to the server. It is generally used for POST requests.

setRequestHeader(): It is used to add key/value pair to the header

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

XMLHttpRequest Object - Properties

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

XMLHttpRequest Object - Properties

status: Returns the status-number of a request

      200: "OK"

      403: "Forbidden"

      404: "Not Found"

      Check here for more https://www.w3schools.com/tags/ref_httpmessages.asp

statusText: Returns the status-text (e.g. "OK" or "Not Found")

Send a request to the server

Use the open() and send() methods of the XMLHttpRequest object:

```
const request_countries = new XMLHttpRequest();

request_countries.open('GET', 'https://restcountries.com/v3.1/all?fields=name,flag')

request_countries.send()
```

Receive a response

You may use **responseText/responseXML**

Example:

const myVar = JSON.parse(**this.responseText**);

Example and Practice

Create and ajax.html webpage:

```
<!DOCTYPE html>
<html>
  <body>
      <select
      class="countries_select"
      name="countries"></select>
      <div class="result"></div>
      <script src="js/ajax.js"></script>
  </body>
</html>
```

Example and Practice

```
// Creating XMLHttpRequest object
const request_countries = new XMLHttpRequest();
// Open the given file
request_countries.open( "GET", "https://restcountries.com/v3.1/all?fields=name,flag");
// Sending the request
request_countries.send();
let countries_list;
// Creating a callback function
request_countries.addEventListener("load",  function () {
 countries_list = JSON.parse(this.responseText);
 populateSelectCountries(countries_list);
});

const selectCountriesContainer = document.querySelector(".countries_select");
const populateSelectCountries = function (data) {
 for (const country of data.sort((a, b) => a.name.common > b.name.common)) {
     let optionHtml = `<option value="${country.name.common}">${country.flag}
${country.name.common}</option>`;
     selectCountriesContainer.insertAdjacentHTML("beforeend", optionHtml);
 }
};
```

Practice

Using the previous example, display the following information about a country when it is selected in the dropdown menu:

Name: France
Capital: Paris
Language: French
Population: 67391582
Currency: EUR - Euro
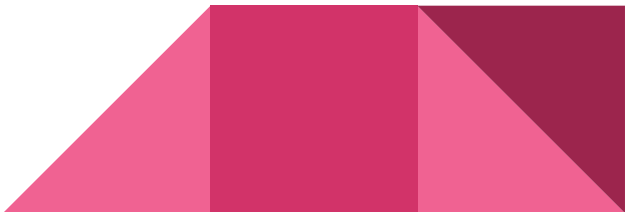Flag: Load the flag image (no the little icon)

Data by country is available here (change country_name):

https://restcountries.com/v3.1/name/**country_name**

https://restcountries.com/v3.1/name/France

Commit your work to GitHub (homework/unit_2/05_ajax/your_files_here)

Fetch request

Fetch is an interface to implement an AJAX request. It is supported by modern browsers it is normally use to call an API.

const promise = fetch(url)

Fetch returns a promise, and you can handle errors in case of problems while connecting to the server or receiving a response.

fetch(url)

  .then(response => response.json())

  .catch(err => console.log(err))

Example using fetch

```
fetch("https://restcountries.com/v3.1/all?fields=name,flag")

  .then(function (response) {

      return response.json();

  })

  .then(populateSelectCountries)

  .catch(err => console.log(err));
```

The then() method in JS has been defined in the Promise API and is used to deal with asynchronous tasks such as an API call.