

The background is a solid teal color with various faint, white line drawings and sketches overlaid. These include architectural elements like a brick wall, a staircase, and a building; scientific or technical diagrams like a circular structure with concentric rings, a ladder, and a grid; and a crescent moon in the upper right corner.

JavaScript

Working with http and XHR

Peter Kassenaar – info@kassenaar.com

JavaScript HTTP, or 'AJAX'

Communicating with web servers

“AJAX:

Asynchronous JavaScript And XML”

Ajax in pure JavaScript

- AJAX: as a term, it is **old** now and mostly replaced by simply something like '**http-communication**' or '**fetching** data'.
- Pure, old school Ajax calls are using the object XMLHttpRequest
 - 'XHR'-object
 - Inventors: IE 5, 1998 (!)
- Pass a URL
- Set RequestHeader
- Catch event onreadystatechange

```

const xhr = new XMLHttpRequest();

// 3. Configure it: GET-request for the URL / random user API
xhr.open('GET', 'https://randomuser.me/api/', true);

// 4. Set up the callback function
xhr.addEventListener('readystatechange', function() {
  // 5. Check if the request is complete (readyState 4) and successful (status 200)
  if (xhr.readyState === 4 && xhr.status === 200) {
    // 6. Parse the JSON response
    const response = JSON.parse(xhr.responseText);
    const user = response.results[0];

    // 7. Display the user data
    const userDataDiv = document.getElementById('userData');
    userDataDiv.innerHTML = `
      <h2>${user.name.first} ${user.name.last}</h2>
      <p><strong>Email:</strong> ${user.email}</p>
      <p><strong>Location:</strong> ${user.location.city},
                                ${user.location.country}</p>
      
    `;
  }
});

// 8. Fire/Send the request.
xhr.send();

```

Sample output

Fetching Random User Data

We're using the api <https://randomuser.me/api/> for this example.

Also we are using 'old school' JavaScript XMLHttpRequest object here. This way, you build an XHR-request this in older examples, however

Fetch Random User Data

Ricardo Viana

Email: ricardo.viana@example.com

Location: Mogi Guaçu, Brazil



60_XHR_example.html

Ajax/http and a webserver

- NOTE: This does NOT work on the `file://` system. You *will* need a **webserver**.
- XHR-calls are always being send to a server
 - GET
 - POST
- You can use a `localhost://`-server,
 - or something like `npx serve` from the command line,
 - the current folder is now the root of a webserver / site.
- Lots of examples on StackOverflow, W3Schools, TutsPlus, etc.

<http://net.tutsplus.com/articles/news/how-to-make-ajax-requests-with-raw-javascript/>

Final Script

This is a relatively simple script that will allow you to asynchronously request pages by using a "load(URL, CALLBACK)" function.

```
view plain copy to clipboard print ?
1. // Our simplified "load" function accepts a URL and CALLBACK parameter.
2. load('test.txt', function(xhr) {
3.     document.getElementById('container').innerHTML = xhr.responseText;
4. });
5.
6. function load(url, callback) {
7.     var xhr;
8.
9.     if(typeof XMLHttpRequest != 'undefined') xhr = new XMLHttpRequest();
10.    else {
11.        var versions = ["MSXML2.XmlHttp.5.0",
12.                        "MSXML2.XmlHttp.4.0",
13.                        "MSXML2.XmlHttp.3.0",
14.                        "MSXML2.XmlHttp.2.0",
15.                        "Microsoft.XmlHttp"]
16.
17.        for(var i = 0, len = versions.length; i < len; i++) {
18.            try {
19.                xhr = new ActiveXObject(versions[i]);
20.                break;
21.            }
22.            catch(e){}
23.        } // end for
24.    }
25.
26.    xhr.onreadystatechange = ensureReadiness;
27.
28.    function ensureReadiness() {
29.        if(xhr.readyState < 4) {
30.            return;
31.        }
32.
33.        if(xhr.status != 200) {
34.            return;
35.        }
36.
37.        // all is well
38.        if(xhr.readyState == 4) {
39.            callback(xhr);
40.        }
41.    }
42.
43.    xhr.open('GET', url, true);
44.    xhr.send('');
45.
46. }
```

<http://www.w3schools.com/ajax/ajax-examples.asp>

w3schools.com

HOME HTML CSS JAVASCRIPT JQUERY XML ASP.NET PHP SQL MORE... REF

AJAX Basic

AJAX HOME
AJAX Intro
AJAX Example

AJAX

XMLHttpRequest

XHR Create Object
XHR Request
XHR Response
XHR readyState

AJAX Advanced

AJAX ASP/PHP
AJAX Database
AJAX XML File

AJAX Examples

AJAX Examples

AJAX Examples

« Previous



Try it Yourself - Examples

A simple AJAX example

Create a simple XMLHttpRequest, and retrieve data from a TXT file.

Load an XML file with AJAX

Create an XMLHttpRequest to retrieve data from an XML file.

Retrieve header information with AJAX

Retrieve header information of a resource (file).

Retrieve specific header information with AJAX

Retrieve specific header information of a resource (file).

Retrieve the content of an ASP file

How a web page can communicate with a web server while a user type characters in an input field.

Retrieve content from a database

How a web page can fetch information from a database with AJAX.

Retrieve the content of an XML file

Create an XMLHttpRequest to retrieve data from an XML file and display the data in an HTML table.

An AJAX example with a callback function

Create an XMLHttpRequest with a callback function, and retrieve data from a TXT file.

« Previous

More modern approach – using `fetch()`

- Modern Browsers (Edge, Chrome, Firefox): use `fetch()`.
- Simplifies the process of making HTTP requests in JavaScript.

HTML Events

- HTML Events
- HTML Event Objects
- HTML Event Properties
- HTML Event Methods

Web APIs

- API Canvas
- API Console
- API Fetch**
- API Fullscreen
- API Geolocation
- API History
- API MediaQueryList
- API Storage
- API Validation

JavaScript Fetch API

[< Previous](#)


Examples


```
fetch(file)
  .then(x => x.text())
  .then(y => myDisplay(y));
```


[Try it Yourself »](#)


Fetch is based on `async` and `await`. The example might be easier to understand like this:


<https://randomuser.me/>


 Home


 User Photos


 Documentation

 Change Log

 Stats & Graphs

 Donate

 Copyright Notice

 Photoshop Extension



```

fetch('https://randomuser.me/api/')
  .then(response => {
    if (!response.ok) {
      throw new Error('Error in Network response');
    }
    // Parse the JSON from the response.
    // It is passed to the next .then() in the chain.
    return response.json();
  })
  .then(data => {
    const user = data.results[0]; // only fetch 1 user.

    // Display the user data.
    const userDataDiv = document.getElementById('userData');
    userDataDiv.innerHTML = `
      <h2>${user.name.first} ${user.name.last}</h2>
      <p><strong>Email:</strong> ${user.email}</p>
      <p><strong>Location:</strong> ${user.location.city},
                                     ${user.location.country}</p>
      
    `;
  })
  .catch(error => {
    console.error('Fetch error:', error);
  });

```

Using `async` and `await`

- **Instead** of using the Promise-notation with `.then()` and `.catch()`, we can also use `async / await` notation
- Functionality is the same!
- `Async / await` is just **syntactic sugar** over Promises
 - More in line with other languages (C#, Java)
- Some like this notation more.
 - **Personal preference**

```
// Mark the function as `async`
async function fetchRandomUser() {
  try {
    const response = await fetch('https://randomuser.me/api/');

    if (!response.ok) {
      throw new Error('Error. ');
    }

    // Use the keyword `await` here.
    const data = await response.json();
    const user = data.results[0];

    // 4. Display the user data
    const userDataDiv = document.getElementById('userData');
    userDataDiv.innerHTML = `...`;
  } catch (error) {
    console.error('Fetch error:', error);
  }
}
```

62_async_await.html

Sample output – identical

Fetching Random User Data using `fetch()` and `async/await`

We're again using the api `https://randomuser.me/api/` for this example.

This script is using the more modern `fetch()` again, but this time not in a Promise notation, but with `async` and `await`. Some like this notation more, but there are no functional differences.

Fetch Random User Data



Anand Dhamdhame

Email: `anand.dhamdhame@example.com`

Location: Ujjain, India



Server sided – PHP, Java, .NET, etc.

- Make sure server returns correct MIME Type!
- You need to configure and format the return results yourself!
 - Based on your server sided platform
- For instance: PHP:

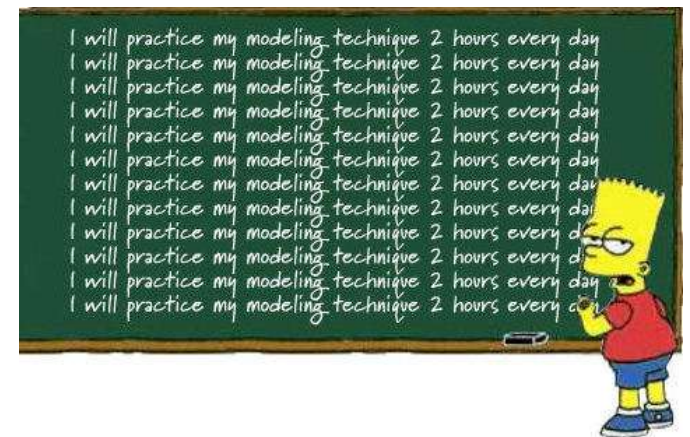
```
1 <?php
2     header('Content-type: application/json')
3 >?
4 <?php
5     // Zorg hier voor het ophalen & formatteren
6     // van gegevens uit de database. Let op
7     // de correcte JSON-notatie met aanhalingstekens,
8     // dubbele punt en komma's.
9 >?
10 { "naam": "peter", "leeftijd" : 43, "woonplaats": "Dieren" }
```

ASP.NET

- Use for instance ASP.NET Web API
- Use Newtonsoft.json DLL

Workshop #40

- Create your own JavaScript-http call to the randomuser.me api
 - Or pick an API you know and like
- Load external data into your page
 - Results can be text, HTML or JSON, depending on the API
- Display all data in a loop on screen.



Sample output

JavaScript- fetching Random User Data

Fetch Random Users

1 - Jovana Rodić

Email: jovana.rodic@example.com

Location: Zaječar, Serbia



2 - Elizabeth Anderson

Email: elizabeth.anderson@example.com

Location: Belmont, Canada

