AJAX

AJAX is a developer's dream, because you can:

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

What is AJAX?

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

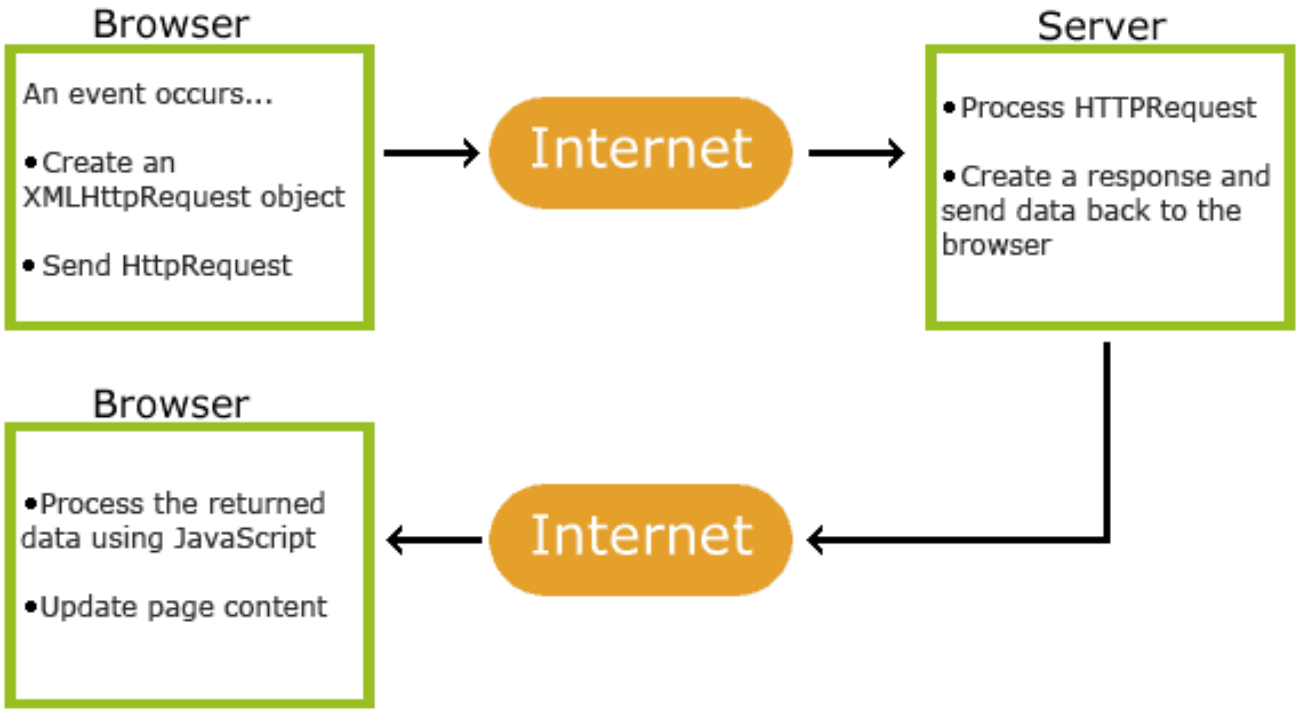
AJAX is not a programming language.

AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or **JSON** text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to **update parts of a web page, without reloading the whole page.**



Example (XMLHttpRequest):

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    displayProjects(JSON.parse(this.responseText));
  }
};
xmlhttp.open('GET', 'api/projects');
xmlhttp.send();
```

Cofinanciado por:

jQuery AJAX

jQuery.ajax()

jQuery Ajax is one of the simplest methods to make an HTTP call.

Examples:

Save some data to the server and notify the user once it's complete.

```
$.ajax({
  method: "POST",
  url: "some.php",
  data: { name: "John", location: "Boston" }
})
  .done(function( msg ) {
    alert( "Data Saved: " + msg );
  });
```

Docs: https://api.jquery.com/jQuery.ajax/

Shorthand Methods

Docs: https://api.jquery.com/category/ajax/shorthand-methods/

jQuery.get()

The $.get method is used to execute GET requests. It takes two parameters: the endpoint and a callback function.

This is a shorthand Ajax function, which is equivalent to:

```
$.ajax({
  url: url,
  data: data,
  success: success,
  dataType: dataType
```

Cofinanciado por:

```
});
```

Get the test.php page contents, which has been returned in json format (<?php echo json_encode(array("name"=>"John","time"=>"2pm")); ?>), and add it to the page.

```
$.get( "test.php", function( data ) {
  $( "body" )
    .append( "Name: " + data.name ) // John
    .append( "Time: " + data.time ); //  2pm
}, "json" );
```

jQuery.post()

The **$.post** method is another way to post data to the server. It take three parameters: the url, the data you want to post, and a callback function.

This is a shorthand Ajax function, which is equivalent to:

```
$.ajax({
  type: "POST",
  url: url,
  data: data,
  success: success,
  dataType: dataType
});
```

Alert the results from requesting test.php with an additional payload of data (HTML or XML, depending on what was returned).

```
$.post( "test.php", { name: "John", time: "2pm" })
  .done(function( data ) {
    alert( "Data Loaded: " + data );
  });
```

Send form data using Ajax requests

```
$.post('api/projects', $('#my-form').serialize());
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

jQuery.getJSON()

This is a shorthand Ajax function, which is equivalent to:

```
$.ajax({
  dataType: "json",
  url: url,
  data: data,
  success: success
});
```

Examples:

```
$.getJSON('api/projects', function(data){
    displayProjects(data)
});
```

Loads the four most recent pictures of Mount Rainier from the Flickr JSONP API.

```
(function() {
  var flickerAPI =
"https://api.flickr.com/services/feeds/photos_public.gne?j
soncallback=?";
  $.getJSON( flickerAPI, {
    tags: "mount rainier",
    tagmode: "any",
    format: "json"
  })
    .done(function( data ) {
      $.each( data.items, function( i, item ) {
        $( "<img>" ).attr( "src", item.media.m ).appendTo(
"#images" );
        if ( i === 3 ) {
          return false;
        }
      });
    });
})();
```

Fontes e mais recursos

https://api.jquery.com/category/ajax/

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Fetch API

Fetch is a new **native JavaScript API**, supported by most browsers today. **Fetch allows you to make network requests similar to XMLHttpRequest**. According to Google Developers Documentation Fetch makes it easier to make asynchronous requests and handle responses better than with the older XMLHttpRequest. **It is an improvement over the XMLHttpRequest API**. The main difference between Fetch and XMLHttpRequest is that the Fetch API uses Promises, hence avoiding callback hell**.**

Fetch Interfaces

The Fetch API has following interfaces

- **fetch()**: The fetch() method used to fetch a resource.
- **Headers**: Represents response/request headers, allowing you to query them and take different actions depending on the results.
- **Request**: Represents a resource request.
- **Response**: Represents the response to a request.

Cofinanciado por:

A basic fetch request is really simple to set up. Have a look at the following code:

```
fetch('http://example.com/movies.json')
  .then(function(response) {
    return response.json();
  })
  .then(function(myJson) {
    console.log(JSON.stringify(myJson));
  });
```

Request options

See `fetch()` for the full options available, and more details.

```
// Example POST method implementation:

postData(`http://example.com/answer`, {answer: 42})
  .then(data => console.log(JSON.stringify(data))) // JSON-string from
`response.json()` call
  .catch(error => console.error(error));

function postData(url = ``, data = {}) {
  // Default options are marked with *
    return fetch(url, {
        method: "POST", // *GET, POST, PUT, DELETE, etc.
        mode: "cors", // no-cors, cors, *same-origin
        cache: "no-cache", // *default, no-cache, reload, force-cache,
only-if-cached
        credentials: "same-origin", // include, *same-origin, omit
        headers: {
            "Content-Type": "application/json; charset=utf-8",
            // "Content-Type": "application/x-www-form-urlencoded",
        },
        redirect: "follow", // manual, *follow, error
        referrer: "no-referrer", // no-referrer, *client
        body: JSON.stringify(data), // body data type must match "Content-
Type" header
    })
    .then(response => response.json()); // parses response to JSON
}
```

Cofinanciado por:

MADEIRA 14-20 · REGIÃO AUTÓNOMA DA MADEIRA · PORTUGAL 2020 · UNIÃO EUROPEIA Fundo Social Europeu

Uploading JSON data

Use `fetch()` to POST JSON-encoded data.

```
var url = 'https://example.com/profile';
var data = {username: 'example'};

fetch(url, {
  method: 'POST', // or 'PUT'
  body: JSON.stringify(data), // data can be `string` or {object}!
  headers:{
    'Content-Type': 'application/json'
  }
}).then(res => res.json())
.then(response => console.log('Success:', JSON.stringify(response)))
.catch(error => console.error('Error:', error));
```

```
fetch('api/projects').then(response => {
  let data = response.json();
});

// method
fetch("api/projects", {
  method: "POST",
  body: formData
});

// headers
fetch("api/projects", {
  method: "POST",
  body: formData,
  headers: {
    "Content-Type": "application/json"
  }
});
```

Fontes e mais recursos

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

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

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu