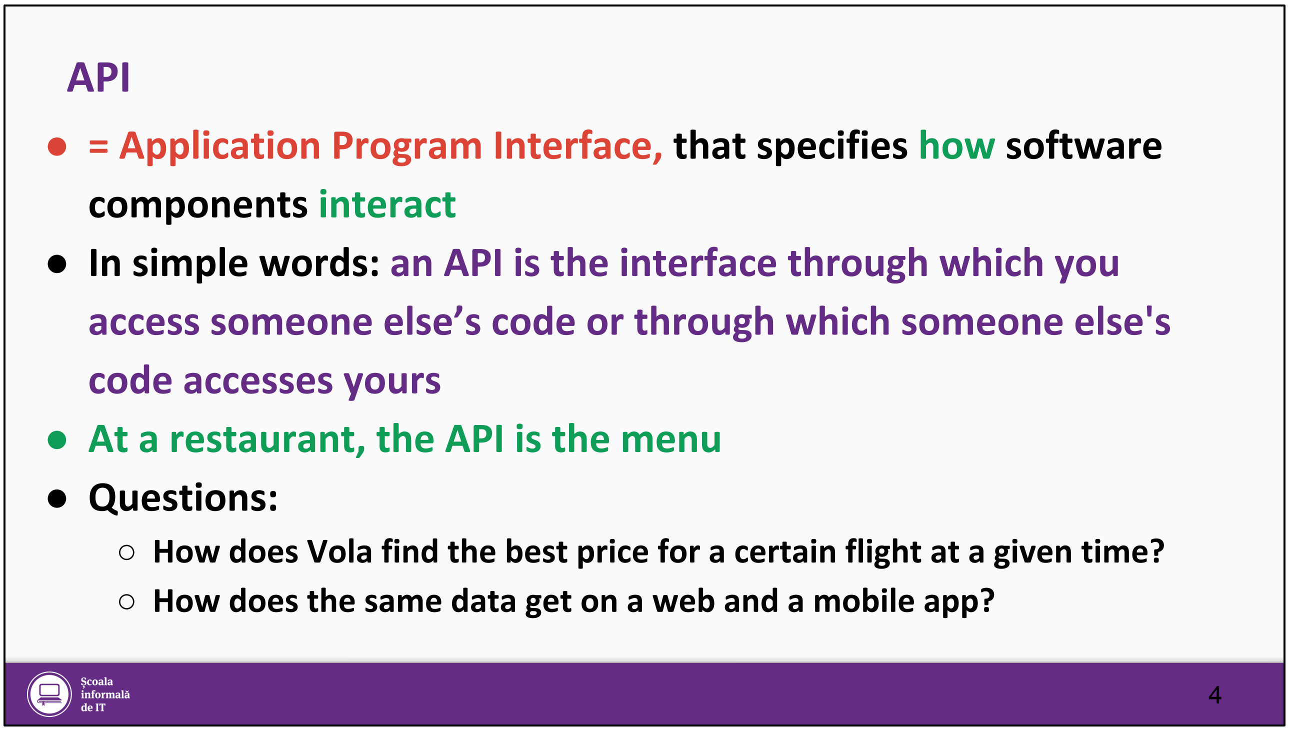
Școala
informală
de IT

JavaScript AJAX

Agenda

- Recap
- What is AJAX
- Sync vs Async programming model
- XMLHttpRequest
- Code samples using jQuery AJAX
- Let's code!

Before we begin...



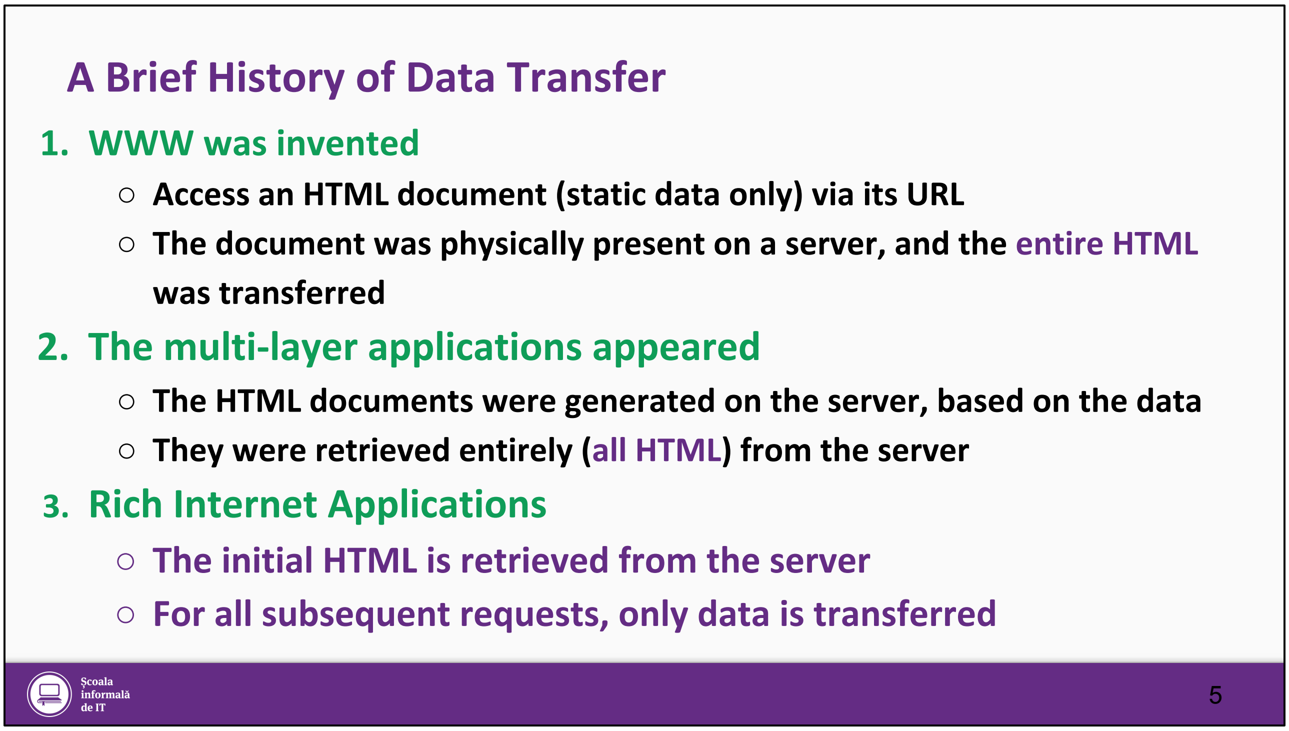
API

- = Application Program Interface, that specifies how software components interact
- In simple words: an API is the interface through which you access someone else’s code or through which someone else's code accesses yours
- At a restaurant, the API is the menu
- Questions:
  - How does Vola find the best price for a certain flight at a given time?
  - How does the same data get on a web and a mobile app?

Generically speaking, **an API or an Application Program Interface is a contract that specifies how software components interact**.

A real-world analogy would be a restaurant, where you as a client use the menu to order some food. You are the client, the waiter along with the chef are the back-end, and the API defines the **contract** between you and them.

When we talk about APIs, we usually talk about the contract between the front-end and the back-end code. In this context, an API is a wrapper over some services offered by the back-end (like reading, adding, updating and deleting data). All these services are accessible using the HTTP request methods and their URL (endpoint).



A Brief History of Data Transfer

1. **WWW was invented**
   - **Access an HTML document (static data only) via its URL**
   - **The document was physically present on a server, and the entire HTML was transferred**
2. **The multi-layer applications appeared**
   - **The HTML documents were generated on the server, based on the data**
   - **They were retrieved entirely (all HTML) from the server**
3. **Rich Internet Applications**
   - **The initial HTML is retrieved from the server**
   - **For all subsequent requests, only data is transferred**

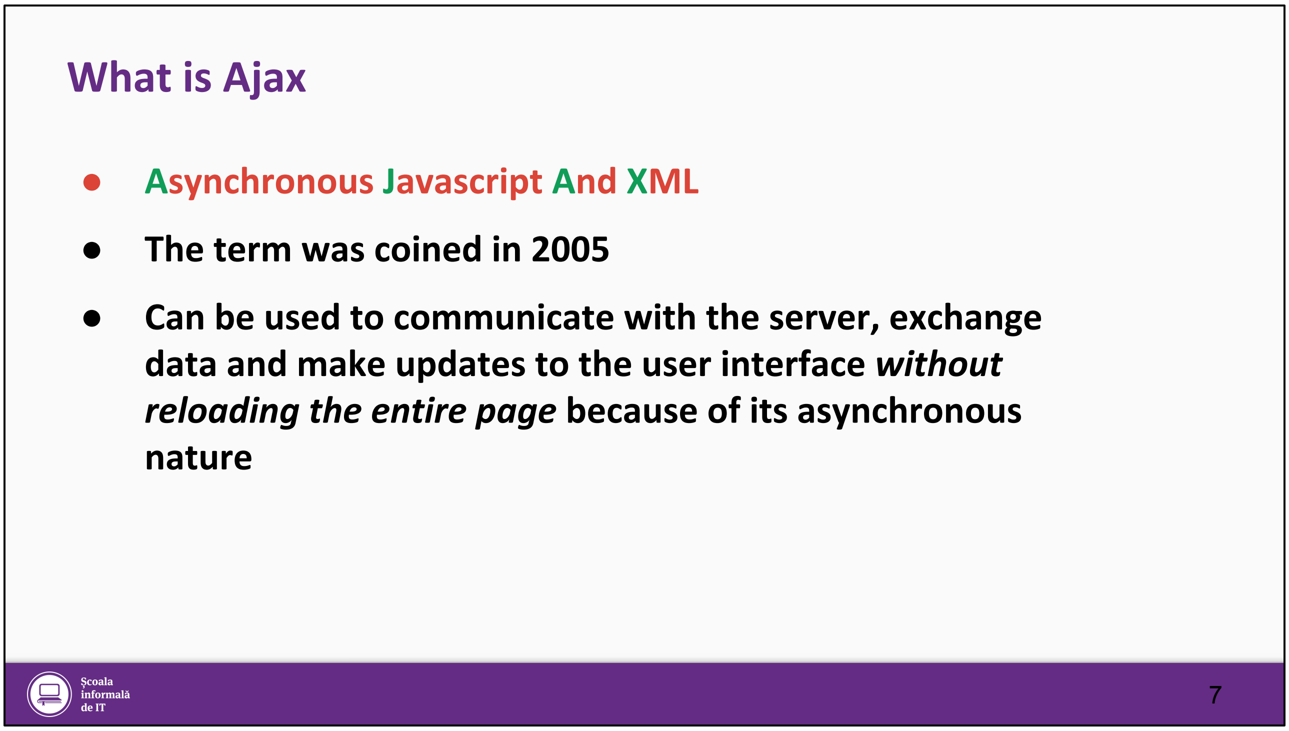
Școala informală de IT

Remember what the World Wide Web is? **A global hyperlinked information system**, invented in the late 1980s. Back then (and some time afterwards) all information was written in HTML documents that were accessible over the Internet using URLs (and HTTP as the communication protocol). When accessing a URL, people were accessing a document that was physically present on a server, and the entire HTML was transferred over the network.

Along with the evolution of the Internet, of software programs and of the web browsers, the multi-layer applications appeared. They were not just static anymore: in an accountancy program, not all suppliers have their own HTML page with their data. Rather than that, the HTML page for a supplier was generated on the server, based on an HTML template and the data for that specific supplier. When accessing a supplier page, the entire HTML document was retrieved from the server.

Over the time, the web browsers evolved faster than the networks, and the people working in the IT industry noticed that the network is the one that is actually slowing down web applications, and that a lot of the resources were called for for too many times. For instance, the header of a web app doesn't necessarily change based on the page the user is on. Thus, a new type of applications started to be developed: Rich Internet Application. In this context, ONLY the initial HTML is retrieved from the server. This includes the HTML to be displayed first, when the page is open, along with the HTML templates for all sections that will be displayed on the page. This makes it possible for all subsequent requests to only transfer data. This is the current state of the apps.

AJAX



What is Ajax

- Asynchronous Javascript And XML
- The term was coined in 2005
- Can be used to communicate with the server, exchange data and make updates to the user interface *without reloading the entire page* because of its asynchronous nature

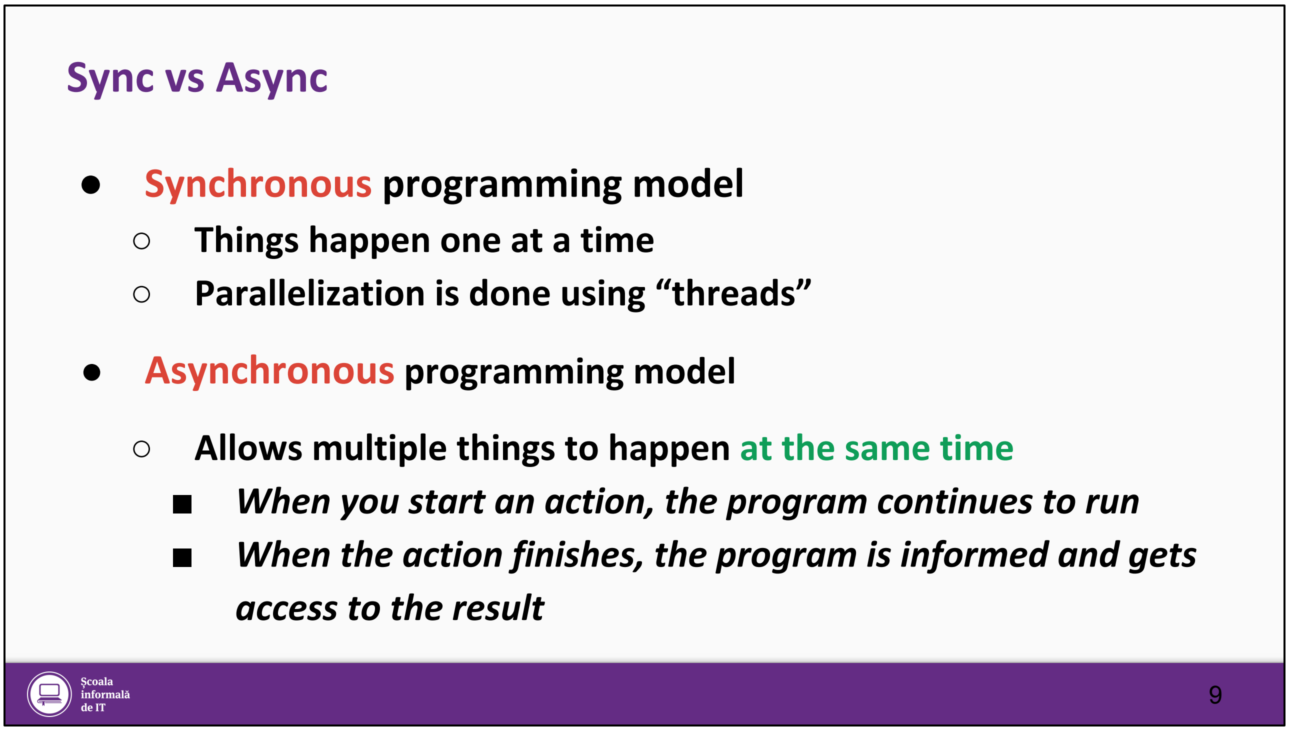
Școala informală de IT

Ajax is an acronym for Asynchronous Javascript And XML. The technology was not new (in 2005, when the term was coined), it rather put several technologies together: HTML or XHTML, Cascading Style Sheets, JavaScript, The Document Object Model, XML, XSLT, and most importantly the XMLHttpRequest object.

The most important feature it has comes from its asynchronous nature: AJAX can be used to communicate with the server, to exchange data and to make updates to the user interface without reloading the entire page.

We'll talk more about what sync and async means next.

Sync vs. Async


Sync vs Async

- Synchronous programming model
  - Things happen one at a time
  - Parallelization is done using “threads”
- Asynchronous programming model
  - Allows multiple things to happen at the same time
    - *When you start an action, the program continues to run*
    - *When the action finishes, the program is informed and gets access to the result*

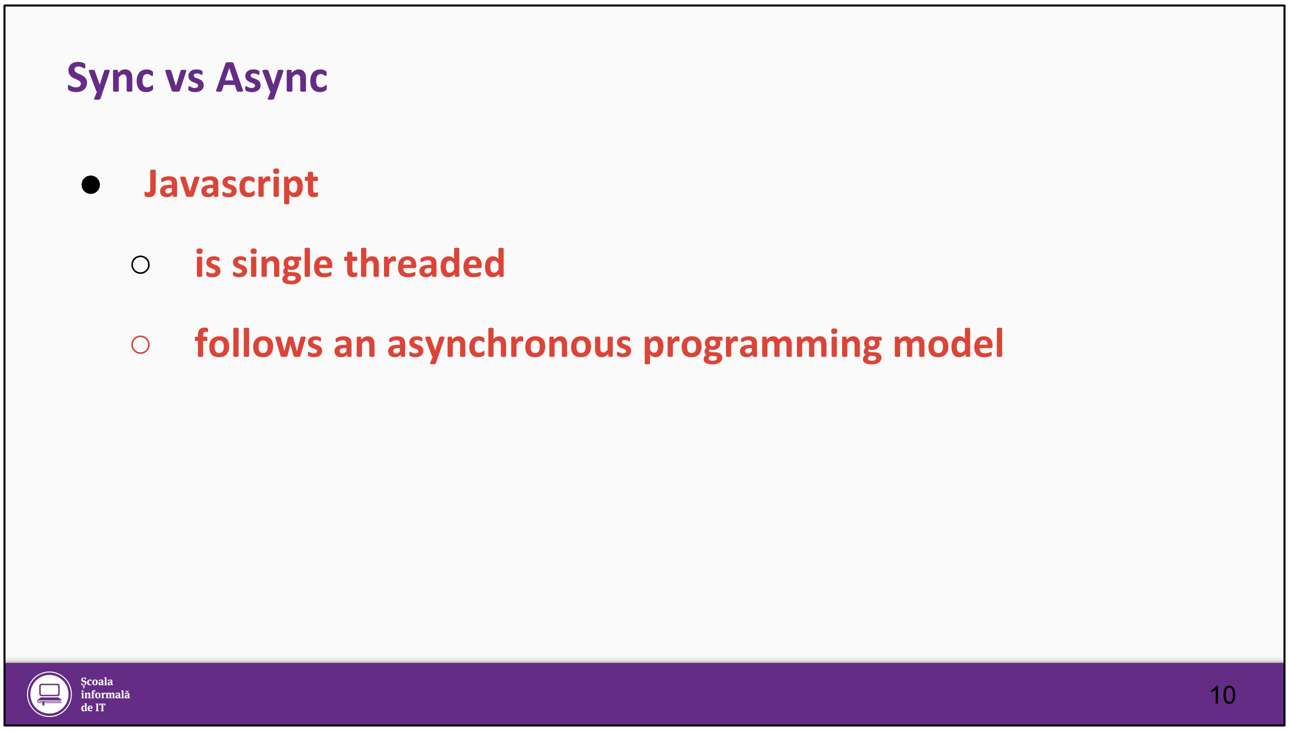
Școala informală de IT

9

Software programs as we know them until now run the code from top to bottom, line by line, and they stop when all code is run. However, in order to be really useful to the user, programs interact with things outside of the processor: for instance, they might need to communicate over a network to get some data, or they might request data from the hard disk. These operations are slower than the code that is run in memory. This means that our programs (web applications, in our case) are freezed until these slow operations are finished. Or at least that’s what we expect to happen, but there are some possible improvements to this:

- In the synchronous programming model, where things happen one at a time (as described above), some operations may be parallelized. That’s done using the so-called “threads”. In simple words, that means that each processor core executes a part of the operations, and the results are merged in the end.
- In an asynchronous model, multiple things are allowed to happen at the same time: when you start an action, the program continues to run. When the action finishes, the program is informed and gets access to the result.

Javascript is single threaded and follows an asynchronous programming model.

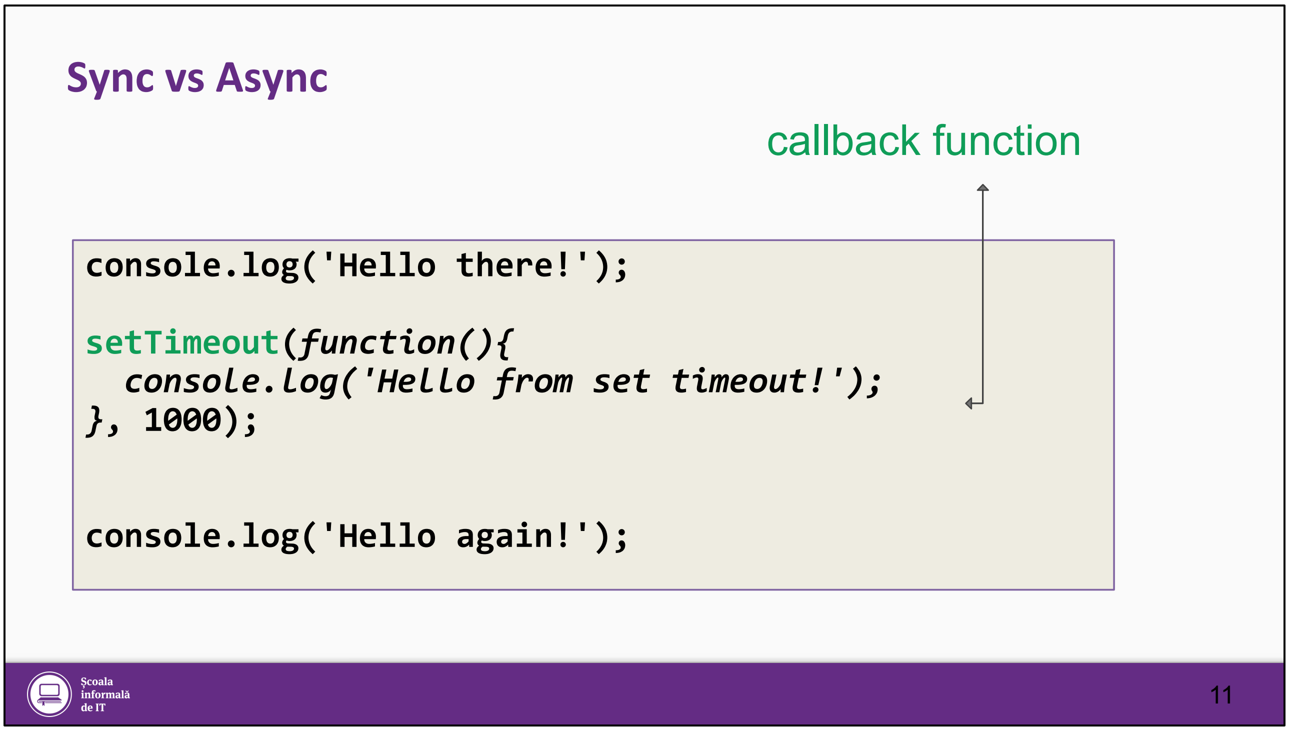


Sync vs Async

- Javascript
  - is single threaded
  - follows an asynchronous programming model

Şcoala informală de IT

Javascript is single threaded and follows an asynchronous programming model.


Sync vs Async

callback function

```
console.log('Hello there!');

setTimeout(function(){
  console.log('Hello from set timeout!');
}, 1000);


console.log('Hello again!');
```

Școala informală de IT

Let's look at some code samples to better understand this.

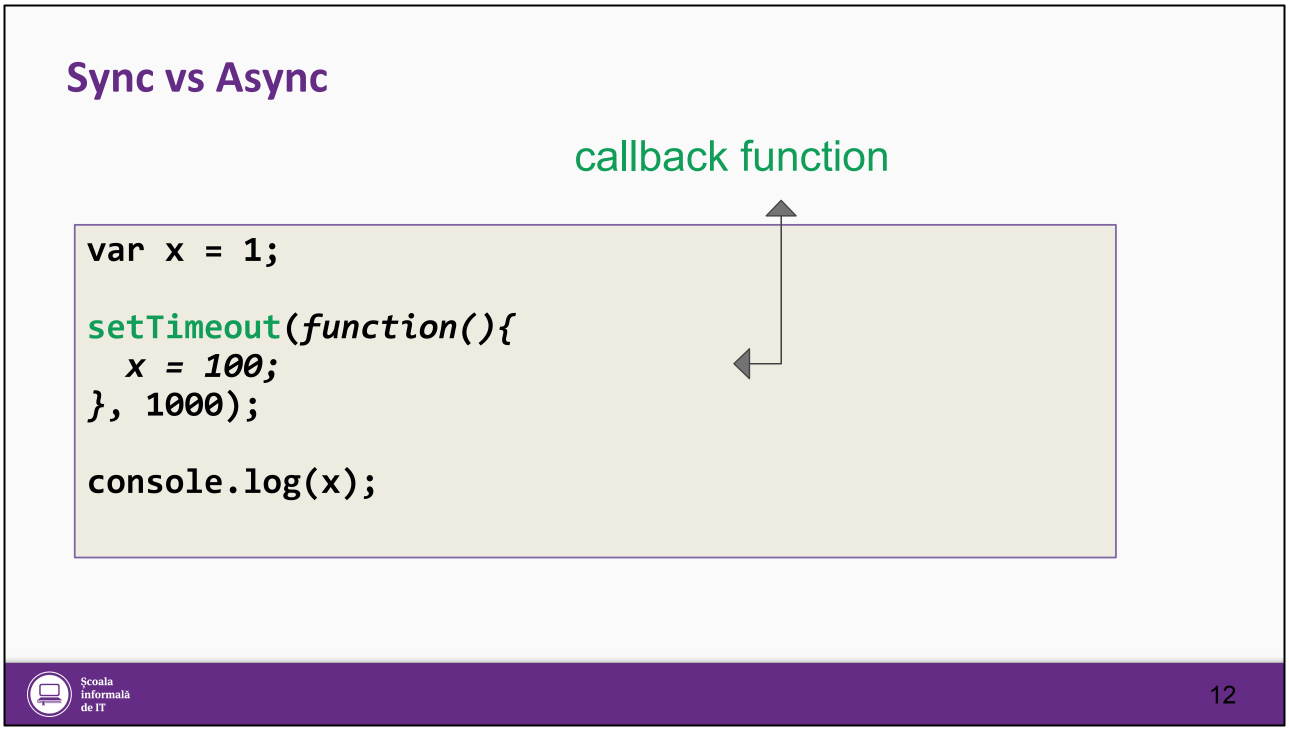
setTimeout sets a timer which executes a function or specified piece of code once after the timer expires (MDN).

In the code above, the text will be displayed in the console in the following order:

- Hello there!
- Hello again!
- Hello from set timeout! (after 1 second)

In async programming, a function that performs a slow action (like a network call) or an intentional delay take an argument called a **callback function**. The action is started, and when it finishes, the callback function is called with the result.

To make it easier to remember what a callback function is, think about the following analogy: you need to wash your car, but you don't have the time to sit there and wait for it to be washed, so you ask the person who's washing it to call you when it's ready and you'll come get it. In other words, you start the slow action - you ask someone to wash your car, and when the action is finished - your car is washed, the person *calls you back* to tell you that it's clean.



Sync vs Async

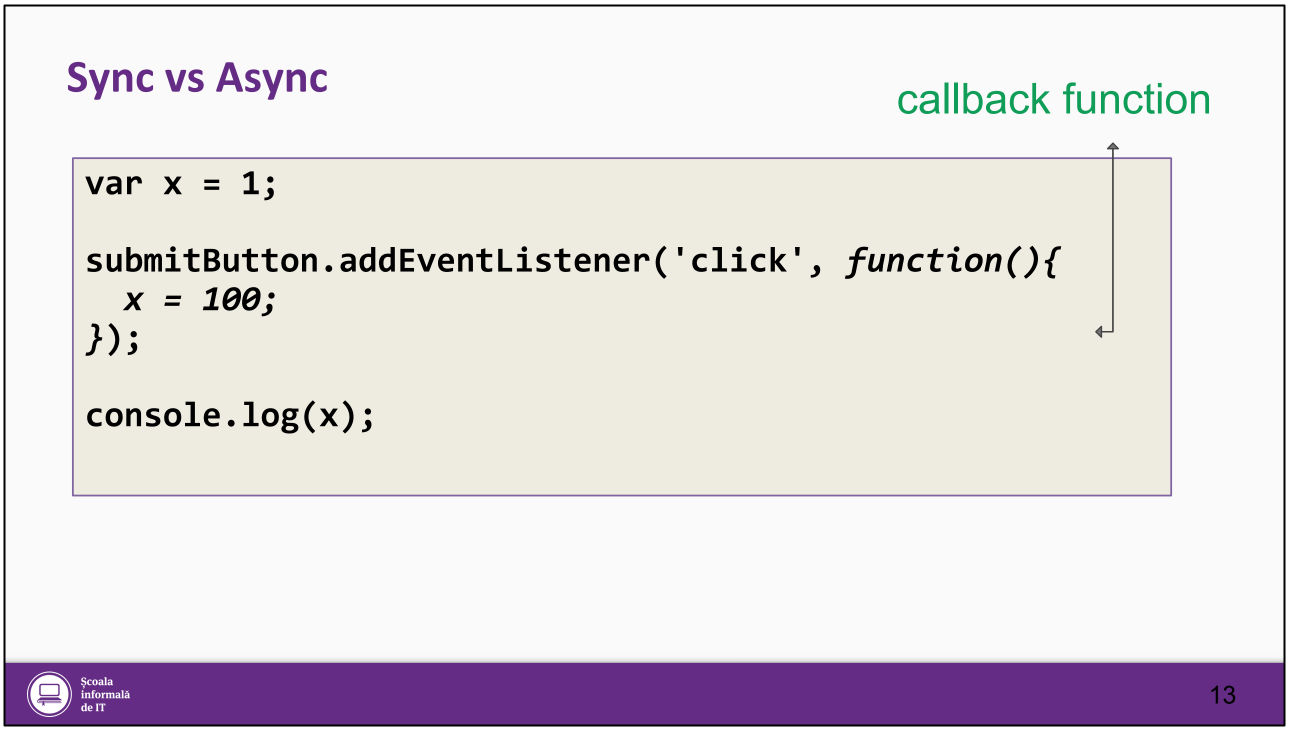
callback function

```
var x = 1;

setTimeout(function(){
  x = 100;
}, 1000);

console.log(x);
```

What is logged in the console?



Sync vs Async

callback function

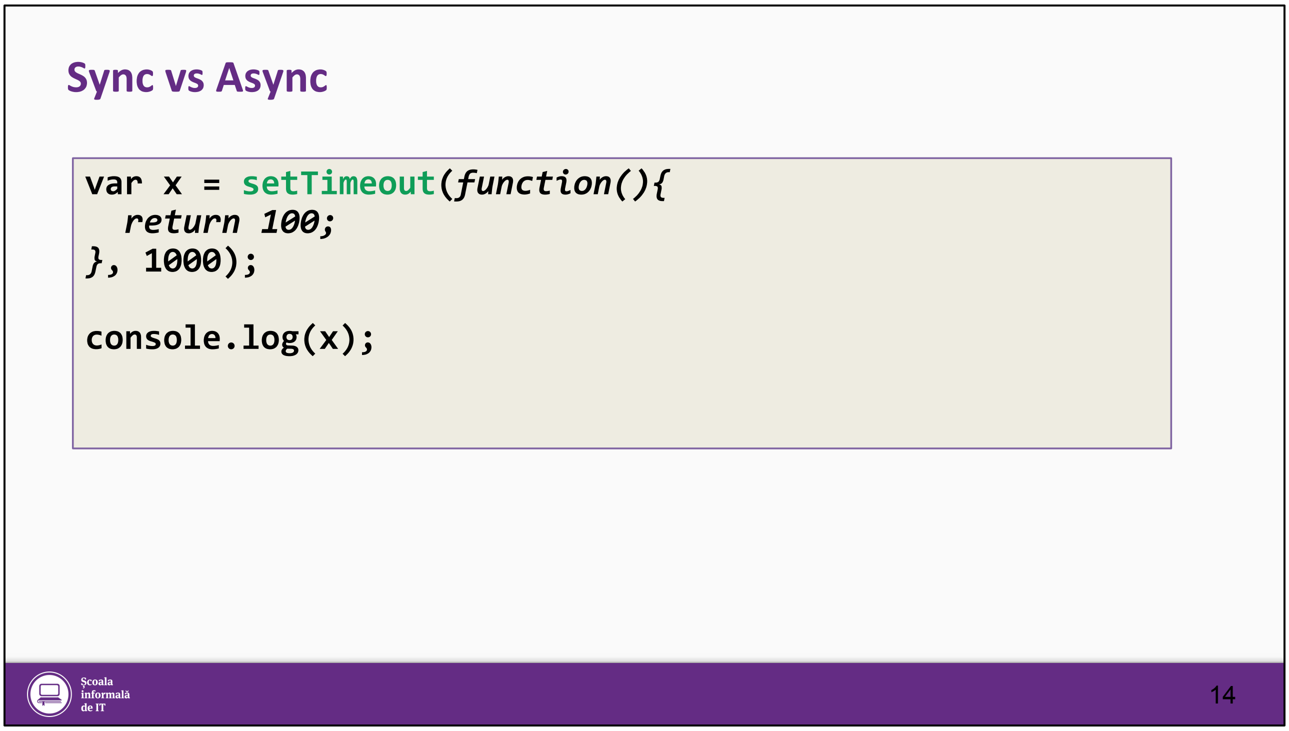
```
var x = 1;

submitButton.addEventListener('click', function(){
  x = 100;
});

console.log(x);
```

Şcoala informală de IT

What is logged in the console?



Sync vs Async

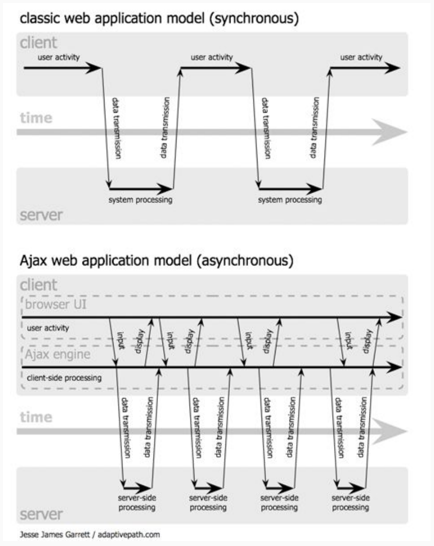
```
var x = setTimeout(function(){
  return 100;
}, 1000);

console.log(x);
```

Școala informală de IT

What is logged in the console?

Sync vs Async in Ajax context



In the classic web application model - the synchronous one, each user activity that needed to trigger a request to the server was causing the page to freeze. When the response came back (this usually meant a reload of the page, because it brought HTML back), the page was ready to be used again. A new user activity triggered again a freeze and a reload of the app and so on.

In the Ajax web application model - the asynchronous one, the page in the browser never freezes: the users can always perform their activities, i.e., trigger requests to the back-end. When the responses are back from the server, the browser is notified, and it’s possible to access the retrieved data using callback functions, and afterwards to display it.

Next, we’ll talk about the most important components of AJAX.

XMLHttpRequest

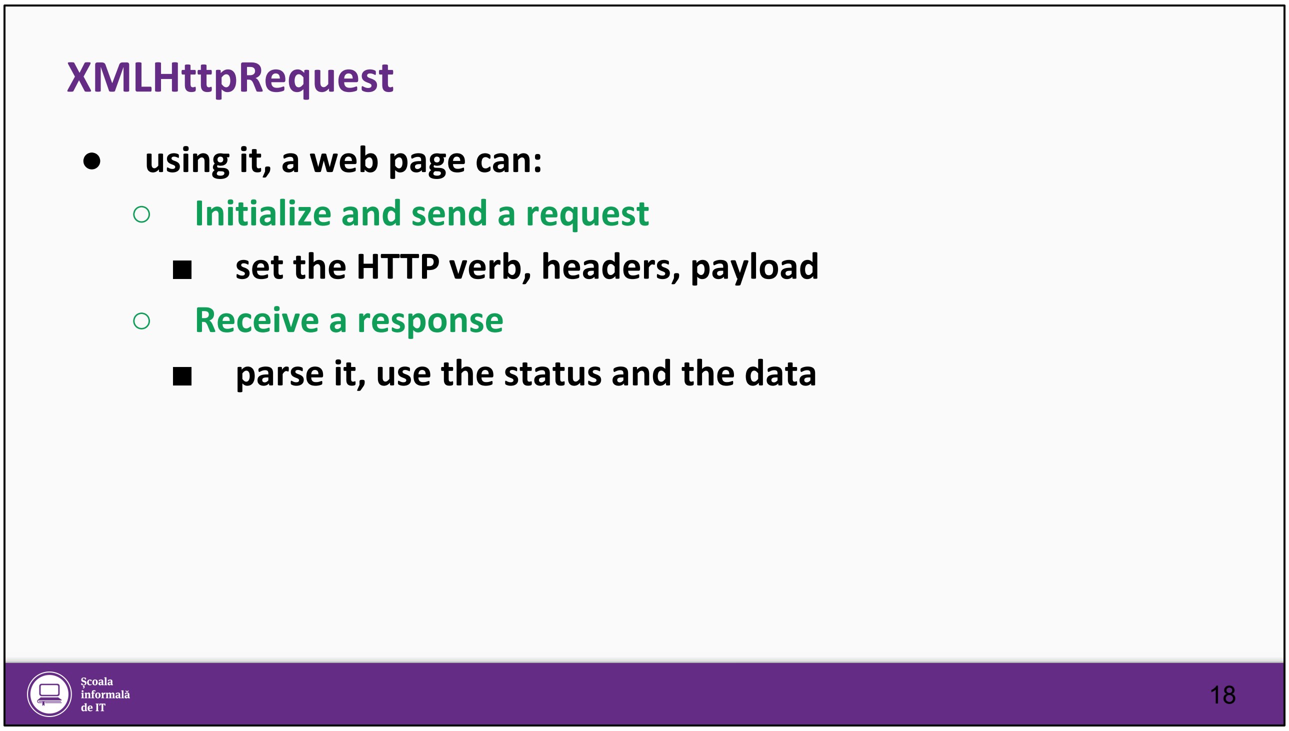
XMLHttpRequest

- = provides the client with functionality for transferring data between a client and a server
- supports a variety of
  - formats: JSON, XML, HTML, text
    - see http://json.org/example.html
  - protocols: HTTP, FTP

Școala informală de IT

XMLHttpRequest is an object that provides the client (the browser) with functionality for transferring data to and from the server.

It supports various formats of data: JSON, XML, HTML, text - we’ll talk about them more in a minute. Also, it can be used over HTTP and FTP protocols.

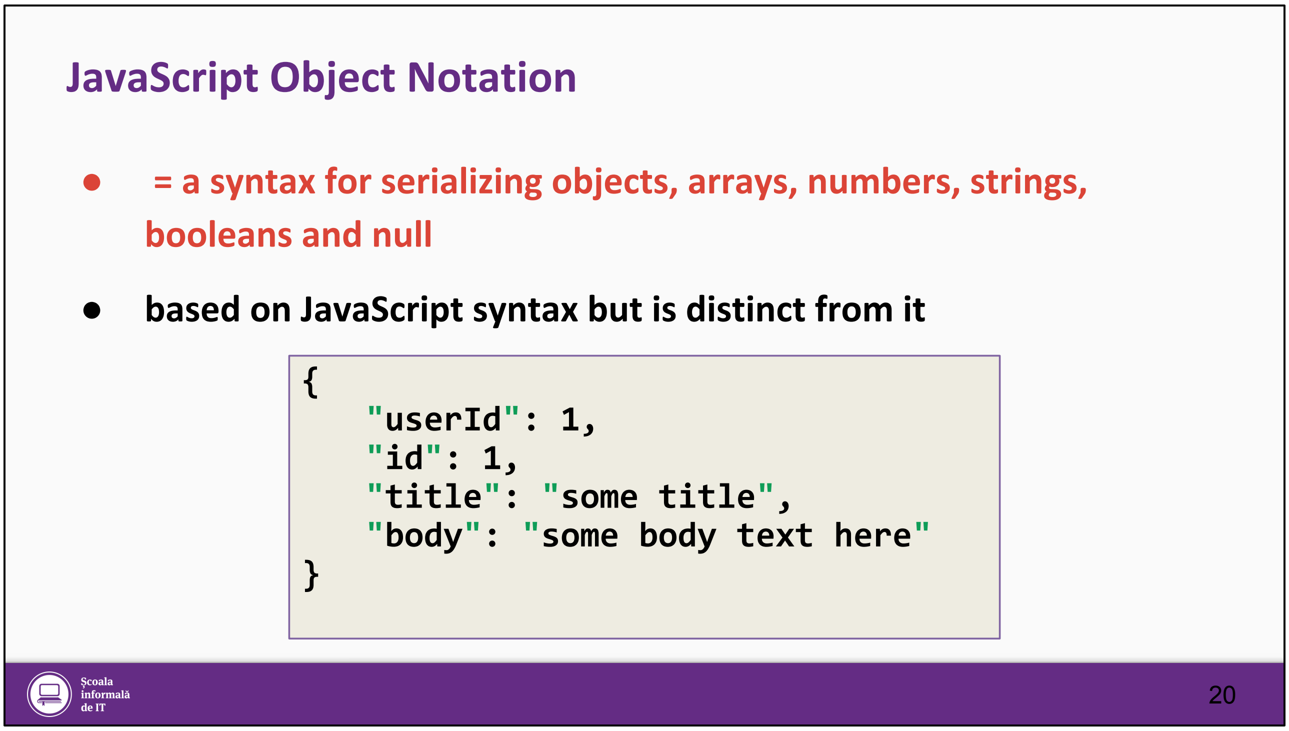


XMLHttpRequest

- using it, a web page can:
  - Initialize and send a request
    - set the HTTP verb, headers, payload
  - Receive a response
    - parse it, use the status and the data

Using XMLHttpRequest object, a web page can send a request to the server, and receive a response and parse it in order to use the retrieved data.

JSON



JavaScript Object Notation

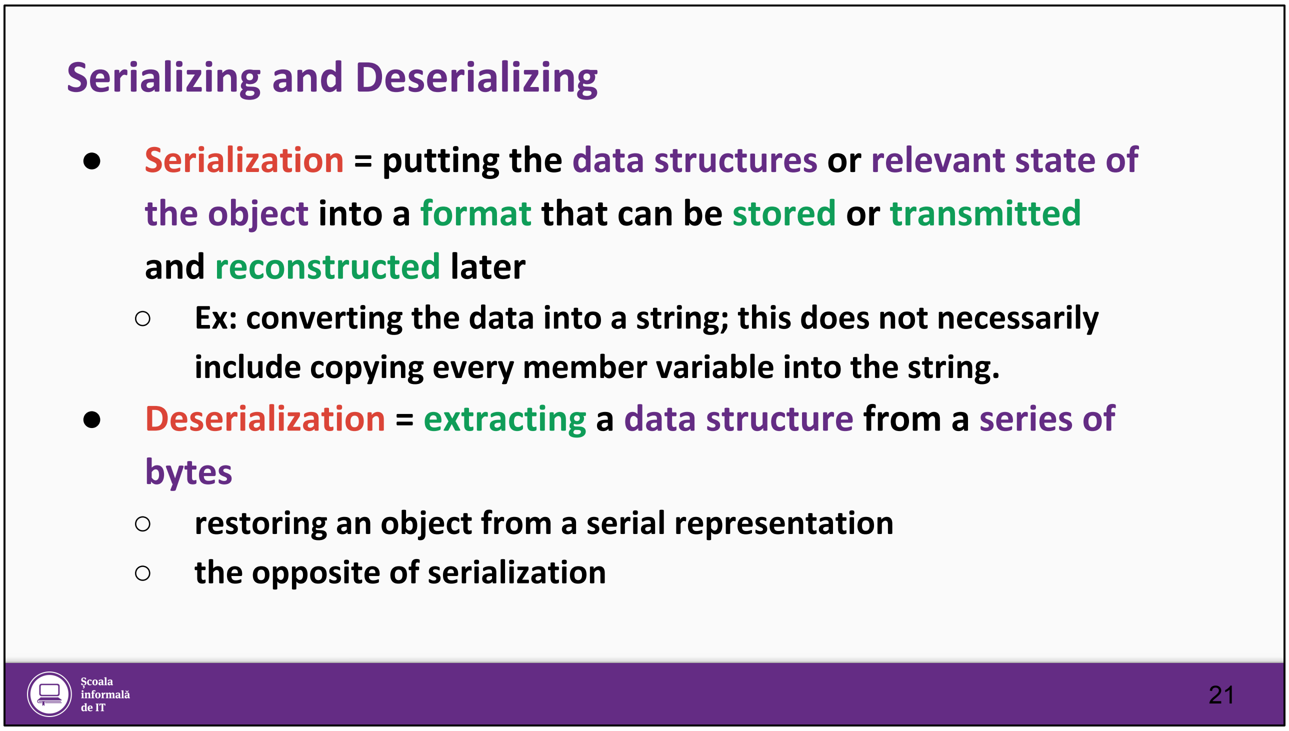
- = a syntax for serializing objects, arrays, numbers, strings, booleans and null
- based on JavaScript syntax but is distinct from it

```
{
    "userId": 1,
    "id": 1,
    "title": "some title",
    "body": "some body text here"
}
```

Although the X in AJAX stands for XML, in the past few years the most used format for sending data was JSON, because it's based on the JavaScript syntax (so it's easier to use) and it's more lightweight than XML.

In JSON, each property in an object needs to be a string, and needs to be wrapped in double quotation marks.

Let's look at the definition though. What is serialization?



Serializing and Deserializing

- Serialization = putting the data structures or relevant state of the object into a format that can be stored or transmitted and reconstructed later
    - Ex: converting the data into a string; this does not necessarily include copying every member variable into the string.
- Deserialization = extracting a data structure from a series of bytes
    - restoring an object from a serial representation
    - the opposite of serialization

Școala informală de IT

Think about the way information flows over the internet. There are basically several packages of **bytes** that are first decomposed and then recomposed at the destination. This means that we cannot send complex data structures (like objects) over the Internet.

In order to transfer the data, however, it's possible to convert the data structures that we're using into a format that can be sent over the network. The most common format is a string. This process of converting the data structures into strings is called **serialization**. We're serializing our objects so that they can be sent over the Internet and then reconstructed at destination.

The "reconstruction" process is called **deserialization**: extracting a data structure from a series of bytes.

JSON Methods

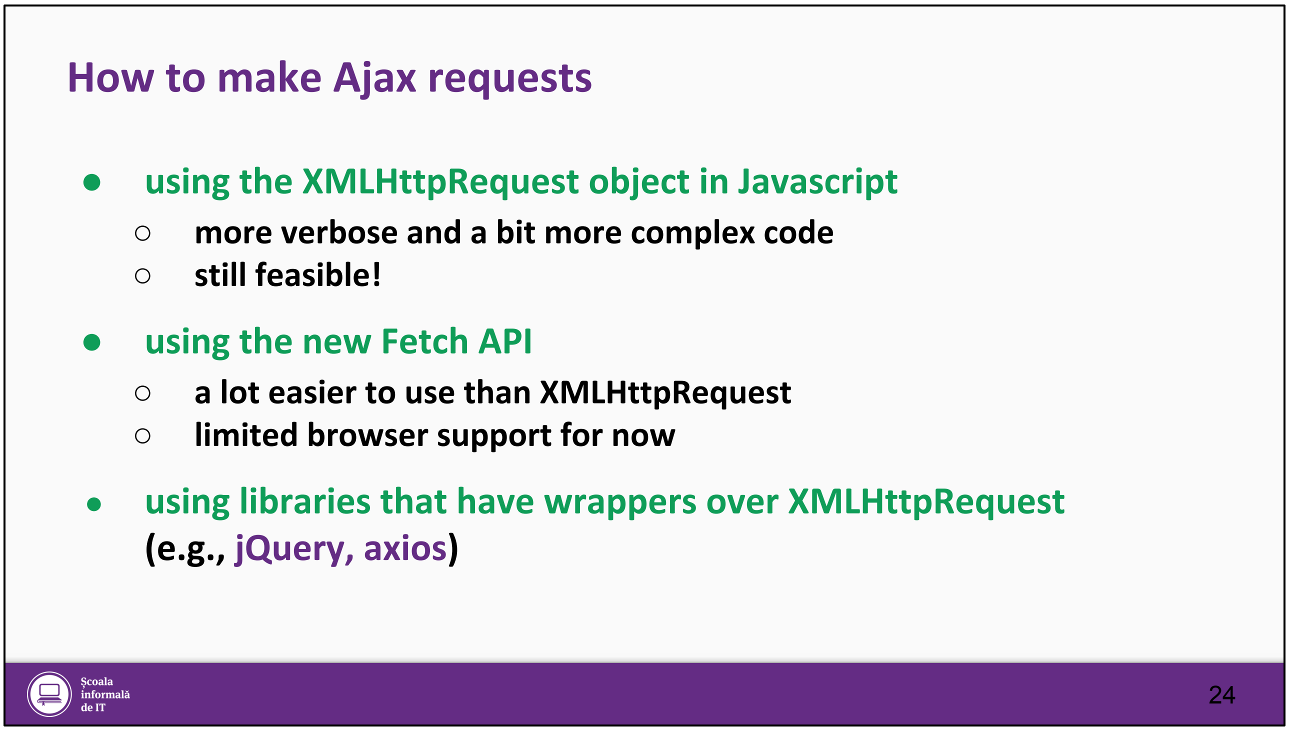
- JSON.parse() (MDN) - deserialization
  - Parse a string as JSON, optionally transform the produced value and its properties and return the value

```
JSON.parse('{}');                    // {}
```

- JSON.stringify() (MDN) - serialization
  - Return a (JSON formatted) string corresponding to the specified value
  - Optionally including only certain properties or replacing property values in a user-defined manner

```
JSON.stringify({ x: 5 });         // '{"x":5}'
```

Coding Ajax Requests

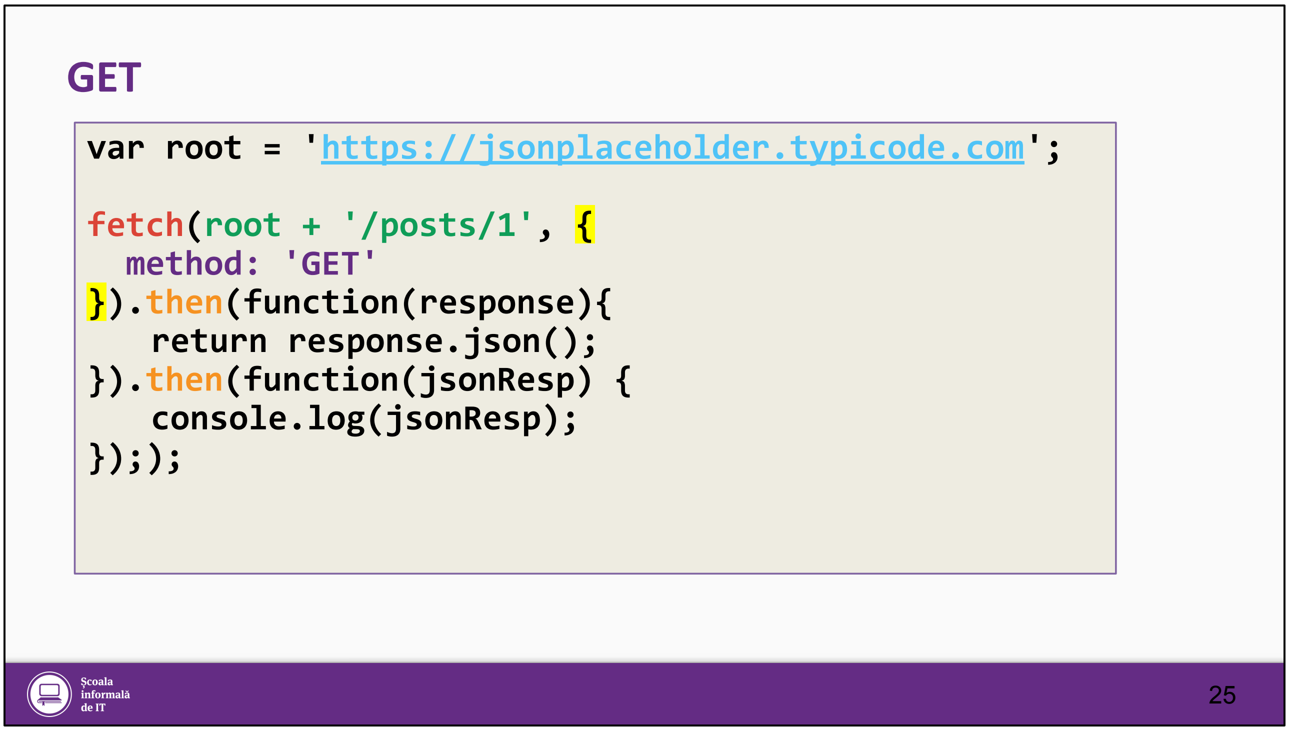


How to make Ajax requests

- using the XMLHttpRequest object in Javascript
  - more verbose and a bit more complex code
  - still feasible!
- using the new Fetch API
  - a lot easier to use than XMLHttpRequest
  - limited browser support for now
- using libraries that have wrappers over XMLHttpRequest (e.g., jQuery, axios)

There are three ways of making Ajax requests: using the XMLHttpRequest object, using the new Fetch API and using libraries that abstract the complexity away from us.

We'll have examples of Fetch API.



GET

```
var root = 'https://jsonplaceholder.typicode.com';

fetch(root + '/posts/1', {
  method: 'GET'
}).then(function(response){
    return response.json();
}).then(function(jsonResp) {
    console.log(jsonResp);
}););
```

Using Fetch API, the code that will be executed when the response is retrieved is the one in the “then” callback. We’ll talk more about “then” in the following sessions.

POST

```
fetch(
  root + '/posts', {
  method: 'POST',
  body: JSON.stringify({
    title: 'foo',
    body: 'bar',
    userId: 1
  })
}).then(function(response){
    return response.json();
}).then(function(jsonResp) {
    console.log(jsonResp);
});
```

PUT

```
fetch(
  root + '/posts/1', {
  method: 'PUT',
  body: JSON.stringify({
    id: 1,
    title: 'foo',
    body: 'bar',
    userId: 1
  })
}).then(function(response){
    return response.json();
}).then(function(jsonResp) {
    console.log(jsonResp); });
```

DELETE

```
fetch(
  root + '/posts/1', {
  method: 'DELETE'
}).then(function(response){
    return response.json();
}).then(function(jsonResp) {
    console.log(jsonResp);
});
```

Şcoala informală de IT

Practice

Let’s practice!

- Display the temperature for Cluj-Napoca for today without refreshing the page
- Display posts on the page. Add functionality to:
    - update post
    - delete post

Resources

https://illustrated.dev/api

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

https://blog.garstasio.com/you-dont-need-jquery/ajax/

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

https://jsonplaceholder.typicode.com/