

# 5. Website Building

Adrian Adiaconitei

***LINKAcademy***

# Objective

- ✓ Javascript/jQuery AJAX
  - ✓ Callback
  - ✓ PROMISE
  - ✓ ASYNC/AWAIT
- ✓ SVG

# SINCRON



Ap1.html

# ASINCRON



Ap2.html

# Callback

- ✓ Funcția de tip callback este o funcție apelată în interiorul altei funcții( care și-a terminat executia) pentru a executa o altă acțiune

```
const button = document.getElementById('button');  
function callback(){  
    console.log("Hello world");  
}
```

```
button.addEventListener('click',callback);  
ap3.html
```



# Callback

Funcția de tip callback \$.ajax()

- ✓ Success
- ✓ Error
- ✓ Complete

Ap4.html nu se mai folosește de la **jQuery 3.0**

Promises jQuery >1.6

- ✓ done
- ✓ fail
- ✓ Always

Ap5.html

- ✓ <https://api.jquery.com/jQuery.ajax/#jqXHR>

# Callback hell

- ✓ Apelam sincron mai multe funcții de tip callback, ceea ce va duce la un cod greu de administrat

- ✓ ap6.html

```
asyncJavaScript = function(err, callback) {  
  callback(function(err, callback) {  
    callback(function(err, callback) {  
      callback(function(err, callback) {  
        callback(function(err, callback) {  
          callback(function(err, callback) {  
            callback(function(err, callback) {  
              console.error('CALLBACK HELL');  
            });  
          });  
        });  
      });  
    });  
  });  
});
```

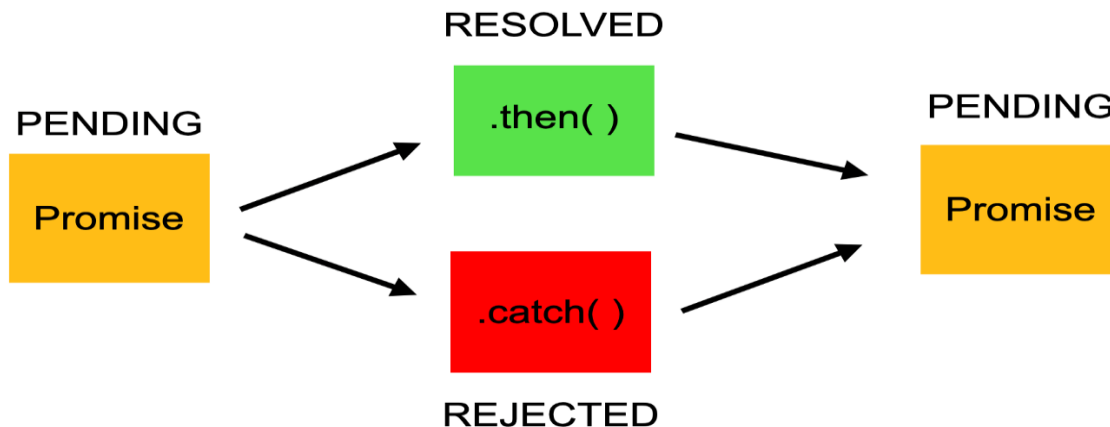
# Promises

O promisiune: un obiect JavaScript care leagă un cod care poate dura ceva timp și un cod care trebuie să aștepte rezultatul ECMAScript 2015(ES6)

- **pending**: stare inițială, în așteptare, nici îndeplinită, nici respinsă
- **fulfill**: operațiunea a fost finalizată cu succes
- **rejected**: operațiunea a eșuat

[https://www.w3schools.com/js/js\\_promise.asp](https://www.w3schools.com/js/js_promise.asp)

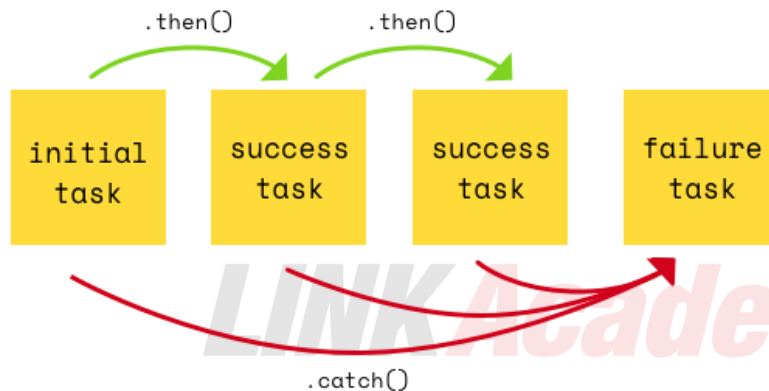
Ap7.html



# Promises

Dacă este nevoie de a executa două sau mai multe operații asincrone, astfel încât fiecare operație ulterioară să înceapă atunci când operația anterioară reușește, de asemenea, operația ulterioară ia rezultatul operației anterioare. Acest lucru se realizează prin crearea **Promise Chaining**.

Ap7\_1.html Ap7\_2.html  
Ap8.html





# Async/await

```
async function f() {  
  return 1;  
}  
  
f().then(alert);
```

```
async function f() {  
  Return Promise.resolve(1);  
}  
  
f().then(alert);
```

# Async/await

**Async/await** – este **Syntactic sugar** pentru **PROMISES**

## Syntactic sugar

- ✓ funcționalitate proiectată pentru rescrierea altei funcționalități făcând codul mai ușor de înțeles
- ✓ NU ne permite să facem ceva ce nu puteam face și înainte folosind alte metode( PROMISES)

# Async/await

- ✓ Există două moduri principale de a gestiona codul asincron: `then/catch` (ES6 - 2015) și **`async/await`** (ES8-2017)
- ✓ Marele avantaj al `async / await` este că face ca codul asincron să pară sincron. (fă cererea, așteaptă să se termine și apoi dă-mi rezultatul)
- ✓ **`Async`** face ca o funcție să returneze o Promisiune
- ✓ **`Await`** face ca o funcție să aștepte pentru o Promisiune
- ✓ **`Await`** poate fi utilizat numai în cadrul unei funcții **`async`**. <ES2022
- ✓ **`Await`** oprește execuția unei funcții asincrone până când Promisiunea este rezolvată. Ap9.html. Ap9\_1.html

# Async/await

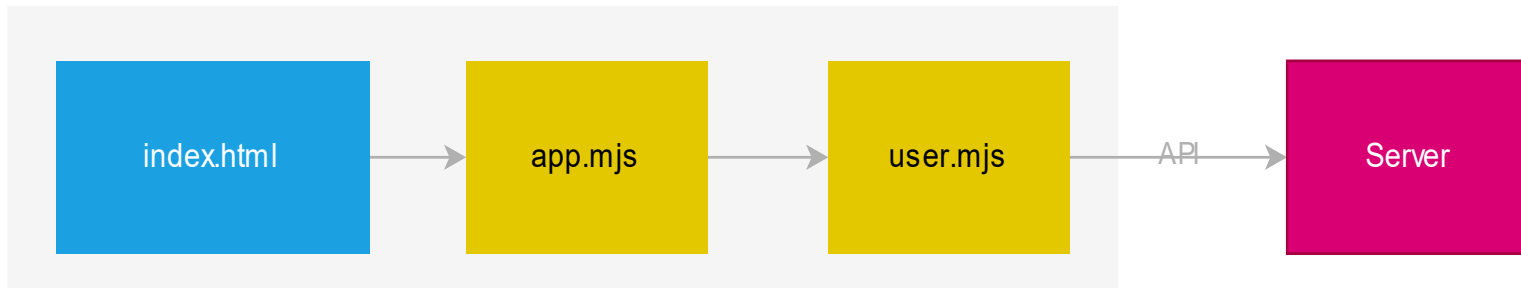
Ap10.html : Provocare Rescriem aplicatia ap2.html folosind  
async/await

Ap11.html : Provocare Rescriem aplicatia ap7\_1.html folosind  
async/await

# Async/await

Aplicație: Să afișam informațiile primite din acest API folosind async/await. (top-level await : ES2022):

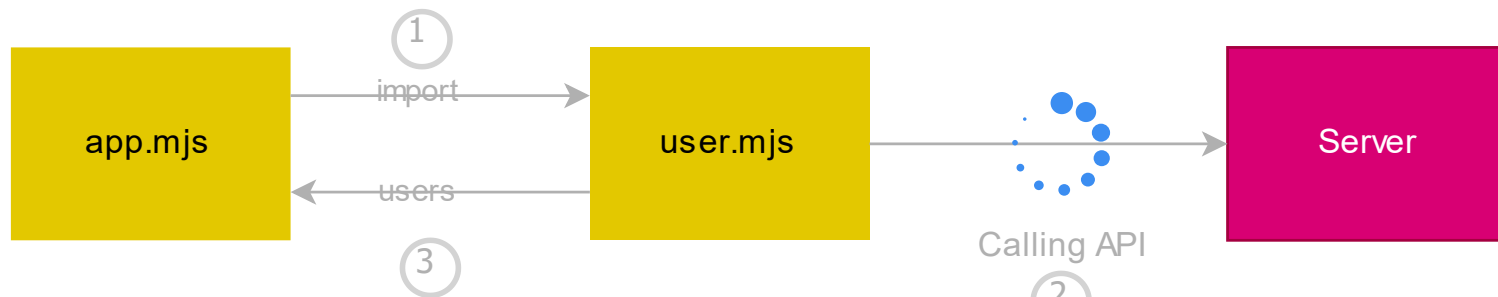
<https://jsonplaceholder.typicode.com/users>



# Async/await

Aplicație: Să afișam informațiile primite din acest API folosind async/await. (top-level await : ES2022):

<https://jsonplaceholder.typicode.com/users>



# JavaScript - cereri asincrone

- ✓ **XMLHttpRequest (XHR) - Vechea metodă**

```
const xhr = new XMLHttpRequest();  
xhr.open("GET", "https://api.example.com/data", true);  
xhr.onload = () => console.log(xhr.responseText);  
xhr.send();
```

- ✓ **Fetch API este recomandat pentru majoritatea cererilor.**

```
fetch("https://api.example.com/data")  
  .then(response => response.json())  
  .then(data => console.log(data))  
  .catch(error => console.error(error));
```

# JavaScript - cereri asincrone

- ✓ **jQuery AJAX este recomandat cand folosim jQuery**

```
$.ajax({  
  url: "https://api.example.com/data", // Endpoint-ul API-ului  
  method: "GET", // Tipul request-ului  
  dataType: "json" // Specificăm că așteptăm JSON ca răspuns  
})  
.done(function(response) {  
  console.log("Success:", response); // Se execută dacă request-ul are succes  
})  
.fail(function(jqXHR, textStatus, errorThrown) {  
  console.error("Error:", textStatus, errorThrown); // Se execută dacă request-ul  
  eșuează  
})  
.always(function() {  
  console.log("Request finished!"); // Se execută indiferent de rezultat (succes sau  
  eroare)  
});
```



# JavaScript - cereri asincrone

## ✓ async/await

```
async function getData() {  
  try {  
    const response = await fetch("https://api.example.com/data");  
    const data = await response.json();  
    console.log(data);  
  } catch (error) {  
    console.error(error);  
  }  
}  
getData();
```

# JavaScript - cereri asincrone

- ✓ **Axios** - O bibliotecă populară bazată pe Promises

```
import axios from "axios";  
axios.get("https://api.example.com/data")  
  .then(response => console.log(response.data))  
  .catch(error => console.error(error));
```

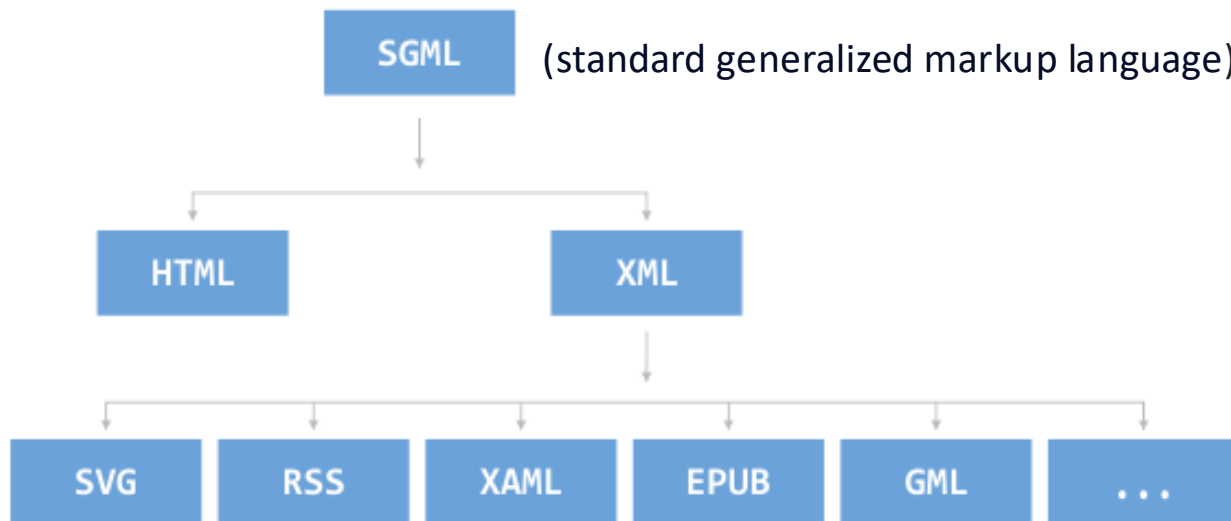
- ✓ **WebSockets**- Pentru comunicare bidirecțională în timp real

```
const socket = new WebSocket("wss://example.com/socket");  
socket.onopen = () => socket.send("Hello Server!");  
socket.onmessage = event => console.log("Message received:", event.data);
```

# SVG- Scalable Vector Graphics

SVG este prescurtarea de la *Scalable Vector Graphics*, care este, în esență, un limbaj descriptiv bazat pe XML folosit pentru a reprezenta grafica vectorială.

```
<rect x="10" y="10" width="200" height="100" fill="#4D4D4D"  
stroke="#4F95FF" stroke-width="3" />
```



# SVG- Scalable Vector Graphics

## Avantaje

- ✓ dimensiune mică a fișierului, important pe web;
- ✓ capacitatea de a defini orice dimensiune;
- ✓ absența pixelizării, adică pierderea calității la mărirea graficii.

## Cum folosim SVG:

- ✓ ``
- ✓ `<object type="image/svg+xml" data="img/logo.svg">`  
`</object>`
- ✓ `<svg xmlns="http://www.w3.org/2000/svg" width="560" height="200"`  
`version="1.1" viewBox="0 0 560 200">`  
`<rect id="my-rect" x="15" y="15" width="300" height="100" /></svg>`

# SVG- Scalable Vector Graphics

Hărți SVG:  
Ap12.html  
Ap13.html

<https://www.amcharts.com/svg-maps/>

<https://mapsvg.com/maps>

<https://www.freecodecamp.org/news/how-to-make-clickable-svg-map-html-css/>



JS

# Bootstrap

**Aplicație: CvTemplate** - folosind exemplele din curs să facem o aplicație de tip onepage: index.html

- ✓ Navbar/Scrollspy
- ✓ Contact: Form Validation
  - ✓ Google captcha
- ✓ Carousel –
  - ✓ Optimizare imagini încărcate în funcție de dispozitiv
- ✓ Modal și Tooltip/Popovers
- ✓ Dark mode și Go to top
- ✓ Folosim.scss Optimizăm( CSS, JS, Imagini) folosind: webpack/gulp
- ✓ [Git API Repo](#)



# Fonts

- ✓ <https://fontawesome.com/search?m=free>
- ✓ <https://fonts.google.com/icons>
- ✓ [https://developers.google.com/fonts/docs/material\\_icons](https://developers.google.com/fonts/docs/material_icons)
- ✓ <https://icons.getbootstrap.com/>
- ✓ <https://lineicons.com/icons/?type=free>
- ✓ <https://www.flaticon.com/animated-icons>
- ✓ <https://fontello.com/>
- ✓ <https://material.io/resources/icons/>

Joc1: **Numar puzzle** *Ordonați numerele*

Mutari: 0

6	7	2
4	8	5
3	1	

Reset

Provocare afișați numărul de mutări





# Resurse

<https://api.jquery.com/jquery.ajax/>

<https://baconipsum.com/json-api/>

[https://www.w3schools.com/js/js\\_callback.asp](https://www.w3schools.com/js/js_callback.asp)

[https://www.w3schools.com/js/js\\_async.asp](https://www.w3schools.com/js/js_async.asp)

<https://stackoverflow.com/questions/23714383/what-are-all-the-possible-values-for-http-content-type-header>

<https://www.iana.org/assignments/media-types/media-types.xhtml>

# Evenimente

<https://geekle.us/>

[https://www.meetup.com/find/?source=EVENTS  
&keywords=javascript](https://www.meetup.com/find/?source=EVENTS&keywords=javascript)

<https://www.devafterwork.ro/events>

[https://www.eventbrite.com/d/online/javascript/  
?page=1](https://www.eventbrite.com/d/online/javascript/?page=1)