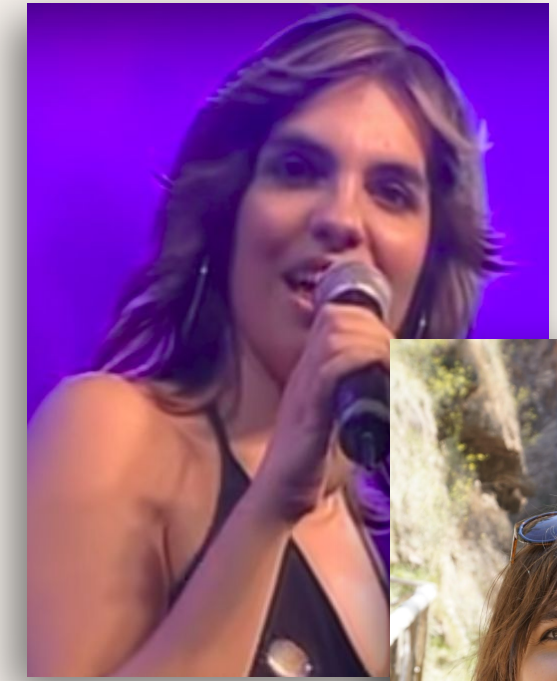
PRIMITIVE OBSESSION

PART I

HOLA!

Soy Aida Albarrán

@aidaispro
aidalbla@gmail.com

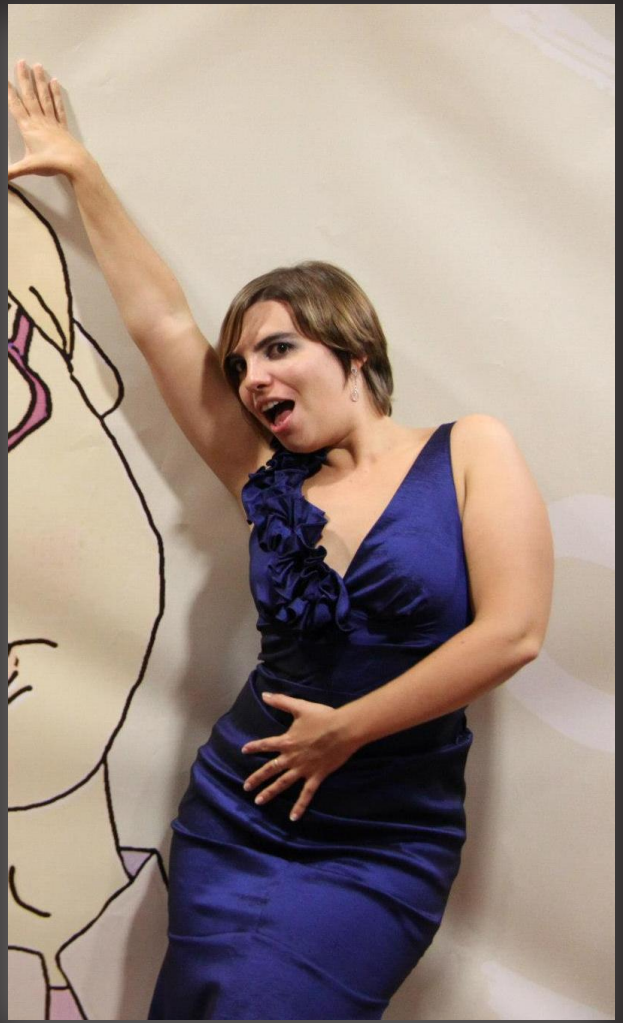


¿DE QUÉ VAMOS A HABLAR HOY?

1. Javascript. No sabes Javascript, John Snow.
2. Clean Code. El arte de programar.
3. Testing. Para refactorizar, crear tests debes.
4. Asincronía. Esa gran “amiga”.

COMIENZOS

ADALAB
powered by ECare



KAIROSDS

1 JAVASCRIPT

2 CLEAN CODE

3 TESTING

4 ASINCRONÍA

NO SABES JAVASCRIPT, JOHN SNOW

Primer encuentro con la realidad

OBJETIVO FUNCIONES

BEFORE:

```
handleResponse(firstList){
  const getAllPokeDatas = firstList.results.map(poke =>
    this.getPokemonByUrl(poke.url));
}
```

OBJETIVO FUNCIONES

AFTER:

```
handleResponse(firstList){
  firstList.results.forEach(poke =>
     this.getPokemonByUrl(poke.url));
}
```

TIPOS DE FUNCIONES

QUERY

ACTION (o COMMAND)

OBJETIVO FUNCIONES

BEFORE:

```
handleResponse(firstList){
  firstList.results.forEach(poke => this.getPokemonByUrl(poke.url));
}
```

AFTER:

```
handleResponse(firstList){
  firstList.results.forEach(poke => this.setPokemonByUrl(poke.url));
}
```

COMMAND QUERY SEPARATION (CQS)

EL ARTE DE SEPARAR

PRIMITIVE OBSESSION / AIDA ALBARRÁN @aidaispro

EDUCACIÓN PARA LA CIUDADANÍA EN JAVASCRIPT

Maléfica

SON CIUDADANOS DE PRIMERA CLASE PORQUE PUEDEN SER...

DEVUELTAS
por otras funciones

ALMACENADAS
en variables

usadas como
PARÁMETROS

FUNCIONES COMO CIUDADANOS DE PRIMERA

```
function assignClass(className) {
 return function(element) {    //Se devuelve esta función
      document.getElementbyId(element).classList.add(className);
  }
}
const hideElement = assignClass('hide'); // Que se almacena en
esta variable
hideElement('myId');   //Ejecuto la función almacenada
```

FUNCIONES COMO CIUDADANOS DE PRIMERA

```
function askForImage(callback) {
 const request = new XMLHttpRequest();
 request.open('GET',
'https://api.thedogapi.com/v1/images/search?size=full' );
 request.addEventListener('load', function() {
   const response = JSON.parse(request.responseText)[0].url;
   callback(response);
 });
 request.send();
}

function showPicture() {
 askForImage(function(image) {
    document.body.innerHTML = '<img src="'+ image + '">';
 })
}
showPicture()
```

FUNCIONES COMO CIUDADANOS DE PRIMERA CLASE

SIRVEN IGUAL PARA UN ROTO QUE PARA UN “DESCOSÍO”

1 JAVASCRIPT

2 CLEAN CODE

3 TESTING

4 ASINCRONÍA

EL ARTE DE PROGRAMAR

Código limpio y sin malos olores

PRIMITIVE OBSESSION / AIDA ALBARRÁN @aidaispro

> **Any fool can write code that a computer can understand. Good programmers write code that humans can understand.**
>
> —Martin Fowler.

PRINCIPIOS S.O.L.I.D.

SINGLE RESPONSIBILITY

DEPENDENCY INVERSION

SOY UNA ARTISTA

VISTA

CONTROLADOR

LÓGICA DE NEGOCIO / MODELO

PERSISTENCIA DE DATOS

```
export interface TweetInfo {
 tweetid: TweetIdClass
 screen_name: string
 followers: number
 retweets: number
 favs: number
 date: string
 link: string
 text: string
}
```

PRIMITIVE OBSESSION SMELL

CODE SMELLS QUE NO HUELEN MUY BIEN

```
export interface TweetInfo {
 tweetid: TweetIdClass
 screen_name: string
 followers: number
 retweets: number
 favs: number
 date: string
 link: string
 text: string
}
```

VALUE OBJECT

Objetos que lo valen. Tanto es así, que se diferencian unos de otros por su valor, pero conservan la misma estructura y además son inmutables.

VALUE OBJECT APLICADO

```
interface PersonProps {
    name: string;
    age: number;
}
```

VALUE OBJECT APLICADO

```
import { valueObject, type } from "valueobject.ts";

class Person extends valueObject({
    name: type.string,
    age: type.number,
}) {
    constructor(args: PersonProps) {
        super(args);
        if (!this.validateAge()) throw new Error('INVALID_AGE');
    }
    validateAge(): boolean {
        if (this.age < 0) return false;
        return true;
    }
}
```

VALUE OBJECT APLICADO

```
function printPerson(person: Person) {
    console.log('Se ha creado:', person);
}

const personData = {
    name: "Aida",
    age: 30
};
const person = new Person(personData);

printPerson(person);
```

```
Se ha creado: Person { name: 'Aida', age: 30 }
```

VALUE OBJECT APLICADO

```
function printPerson(person: Person) {
    console.log('Se ha creado:', person);
}
const personData = {
    name: "Aida",
    age: -20
};
const person = new Person(personData);

printPerson(person);
```

```
                throw new Error('INVALID_AGE');
                ^

Error: INVALID_AGE
```

VALUE OBJECT APLICADO

```
export interface TweetInfo {
 tweetid: TweetIdClass
 screen_name: string
 followers: number
 retweets: number
 favs: number
 date: string
 link: string
 text: string
}
```

1 JAVASCRIPT

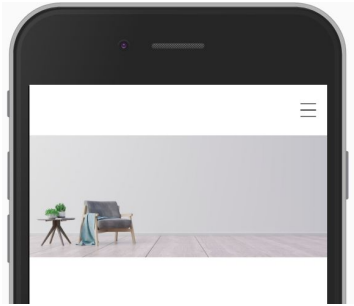
2 CLEAN CODE

3 TESTING

4 ASINCRONÍA

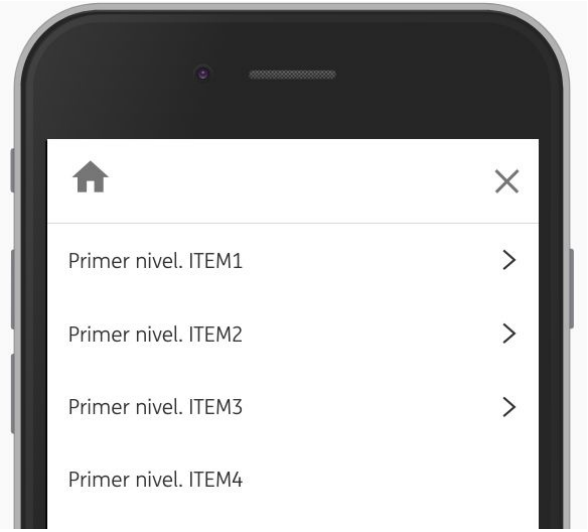
PARA REFACTORIZAR, CREAR TESTS DEBES

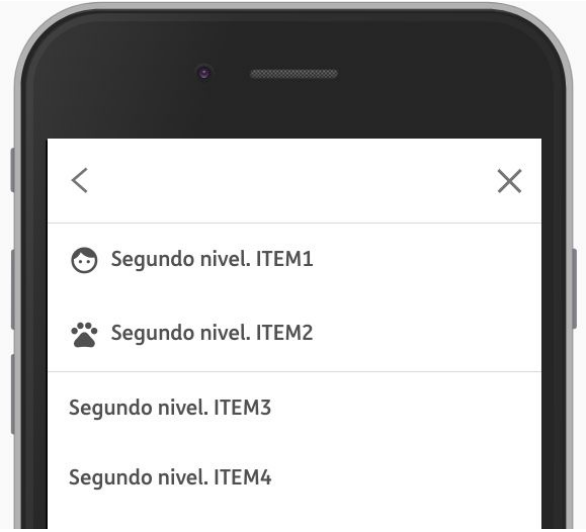
O cómo el testing me ayudó a mejorar mi código.



TESTING Y REFACTOR

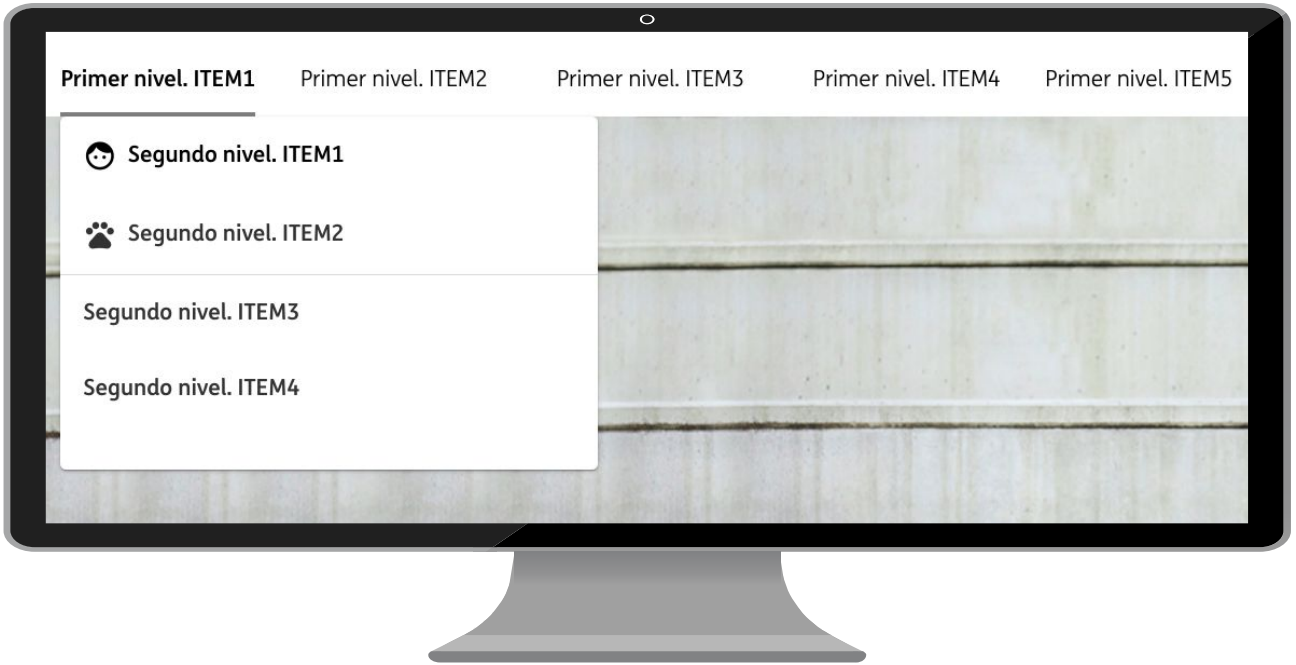
Comportamiento MOBILE





TESTING Y REFACTOR

Comportamiento DESKTOP



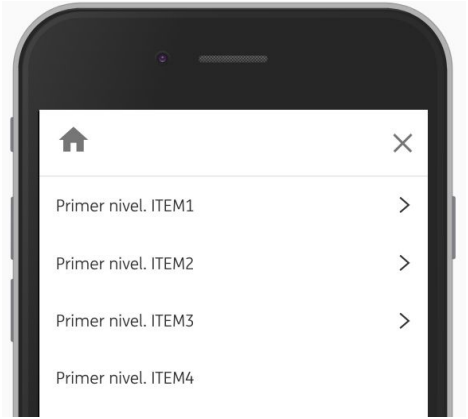
PRIMITIVE OBSESSION / AIDA ALBARRÁN @aidaispro

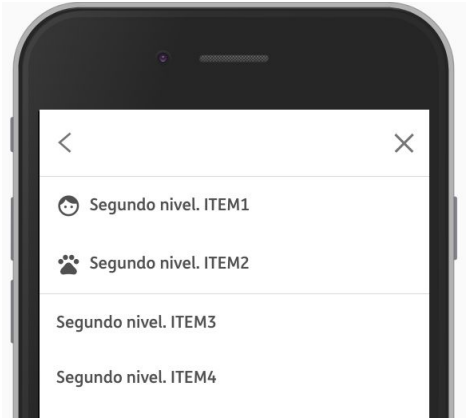
TESTING Y REFACTOR

```
context('Active and not active elements management' , () => {
 function checkIfElementsAreActive listItem, itemContainer, itemSubmenu) {
   expect(listItem.getAttribute('aria-expanded')).to.be.equal('true');
   expect(listItem.querySelector('menu-item').className).to.contain('active');
   expect(itemContainer.className).to.contain('active');
   expect(itemSubmenu.className).to,contain('active');
 }

 function checkIfElementsAreNotActive listItem, itemContainer, itemSubmenu) {
   expect(listItem.getAttribute('aria-expanded')).to.be.equal('false');

expect(listItem.querySelector('menu-item').className).not.to.contain('active');
   expect(itemContainer.className).not.to.contain('active');
   expect(itemSubmenu.className).not.to,contain('active');
 }
```

TESTING Y REFACTOR

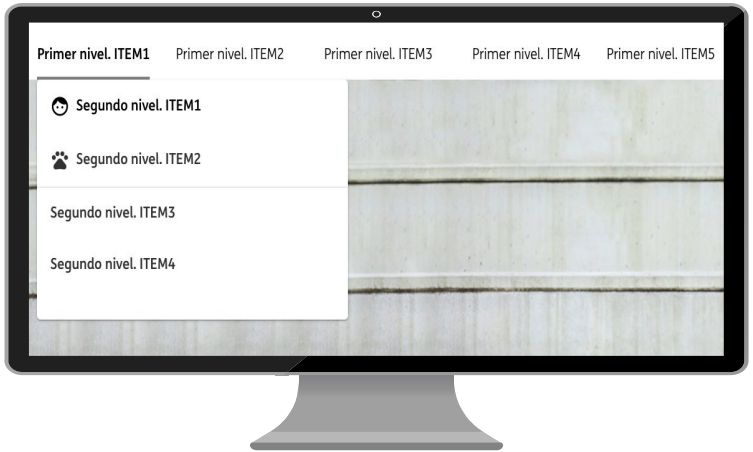
```
it ('should close active menu elements after click ', (done) => {
    flush(() => {
      const currentItem = sut.shadowRoot.querySelectorAll('menu-item')[1];
      const mockEvent = { currentTarget: currentItem.parentNode,};
      sut.toggleMenuItem(mockEvent);
      const listItem = currentItem.parentNode;
      const itemContainer = listItem.parentNode;
      const itemSubmenu = itemContainer.querySelector('submenu').shadowRoot...

      checkIfElementsAreActive(listItem, itemContainer, itemSubmenu);

      sut.closeSubmenu();

      checkIfElementsAreNotActive(listItem, itemContainer, itemSubmenu);
      done();
    });
});
```





TESTING Y REFACTOR

Momento de refactorizar



TDD

Deja que los test “conduzcan” y controlen tus funciones.

1 JAVASCRIPT

2 CLEAN CODE

3 TESTING

4 ASINCRONÍA

ASINCRONÍA

Esa gran amiga a la que todos los desarrolladores de Javascript “amamos”.

```
it ('should close active menu elements after click ', (done) => {
    flush(() => {
      const currentItem = sut.shadowRoot.querySelectorAll('menu-item')[1];
      const mockEvent = { currentTarget: currentItem.parentNode,};
      sut.toggleMenuItem(mockEvent);
      const listItem = currentItem.parentNode;
      const itemContainer = listItem.parentNode;
      const itemSubmenu = itemContainer.querySelector('submenu').shadowRoot...

      checkIfElementsAreActive(listItem, itemContainer, itemSubmenu);

      sut.closeSubmenu();

      checkIfElementsAreNotActive(listItem, itemContainer, itemSubmenu);
      done();
    });
});
```

LENGTH

EN ARRAY: 

```
const myArray = ['Besis', 'de', 'fresi'];
console.log(myArray.length); // 3
```

EN STRING: 

```
const myText = 'Besis';
console.log(myText.length); // 5
```

EN NUMBER Y OBJECT: 

```
const myNumber = 4;
console.log(myNumber.length); // undefined
const myObject = {};
console.log(myObject.length); // undefined
```

LENGTH

EN FUNCTION:

```
function myFunction() {
 // Hago cosas
}
console.log(myFunction.length); // 0
```

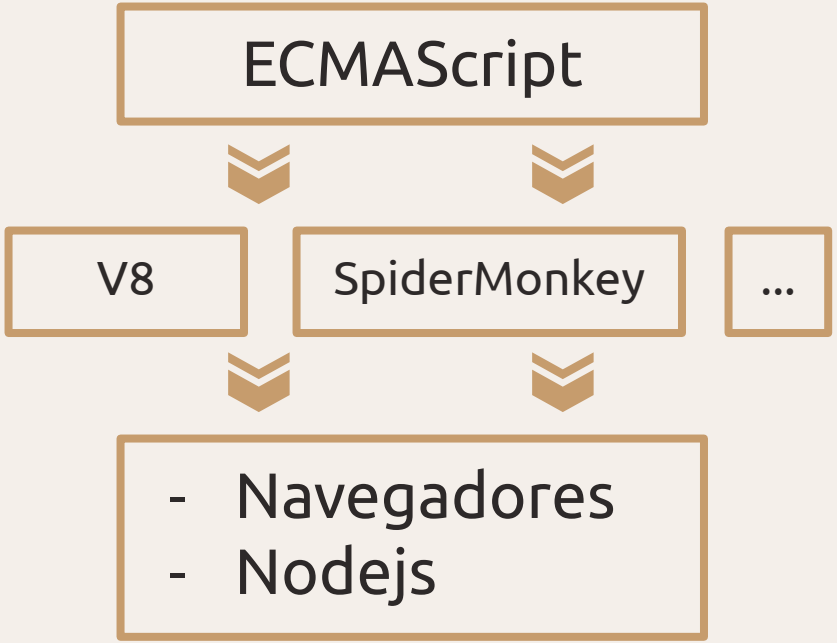
```
function myFunctionWithArguments(name, age, phone)
{
 // Hago cosas
}
console.log(myFunctionWithArguments.length); // 3
```

```
it('should close active menu elements after click', (done) => {
  flush(() => {
    const currentItem = sut.shadowRoot.querySelectorAll('menu-item')[1];
    const mockEvent = { currentTarget: currentItem.parentNode, };
    sut.toggleMenuItem(mockEvent);
    const listItem = currentItem.parentNode;
    const itemContainer = listItem.parentNode;
    const itemSubmenu = itemContainer.querySelector('submenu').shadowRoot...

    checkIfElementsAreActive(listItem, itemContainer, itemSubmenu);

    sut.closeSubmenu();

    checkIfElementsAreNotActive(listItem, itemContainer, itemSubmenu);
    done();
  });
});
```

LENGTH

Una propiedad con más usos que el aceite de oliva virgen extra.

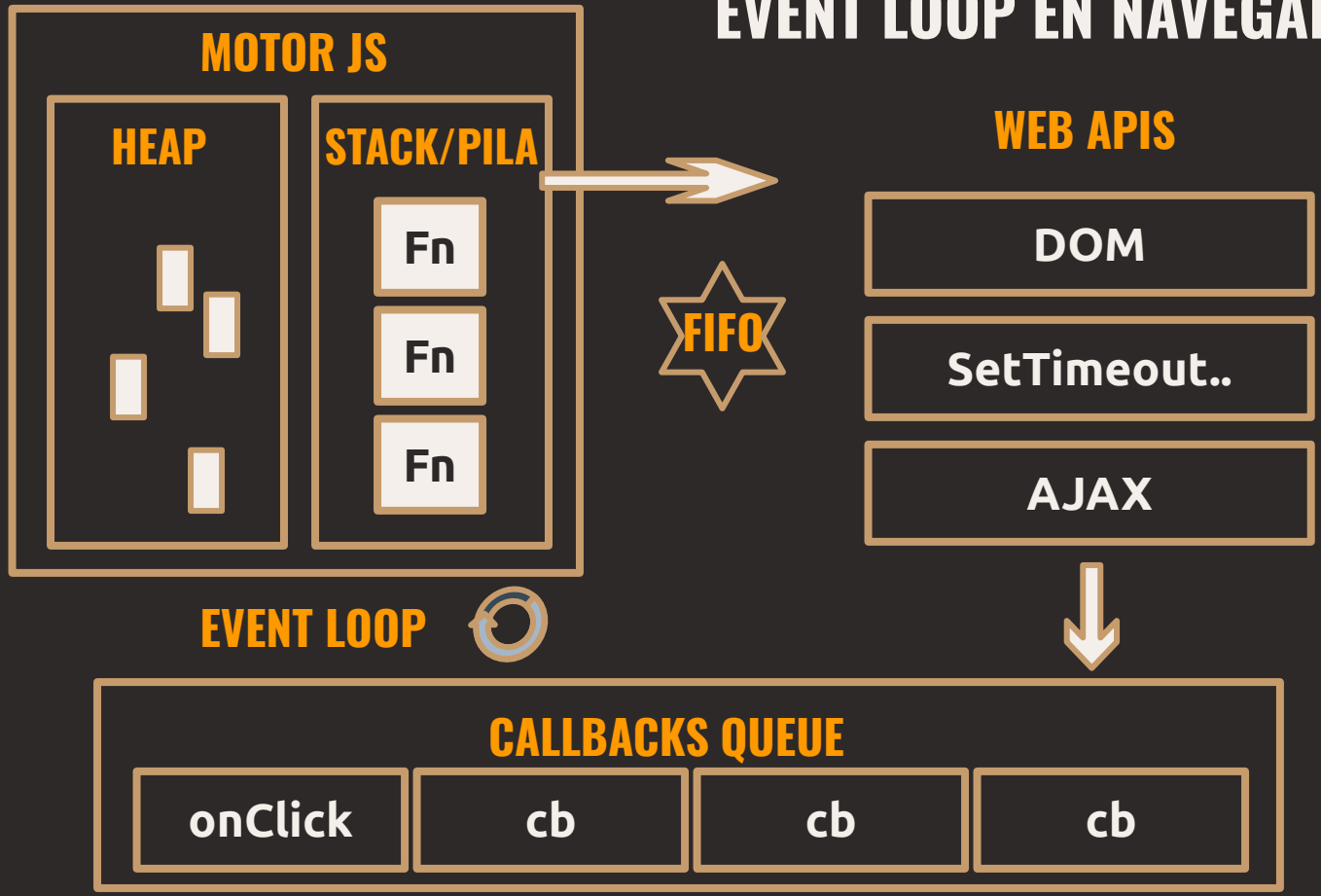
SETTIMEOUT Y SETINTERVAL



MOTOR JS + APIS DEL NAVEGADOR

El extra que te ofrecen los navegadores para hacer tu experiencia con Javascript más placentera....

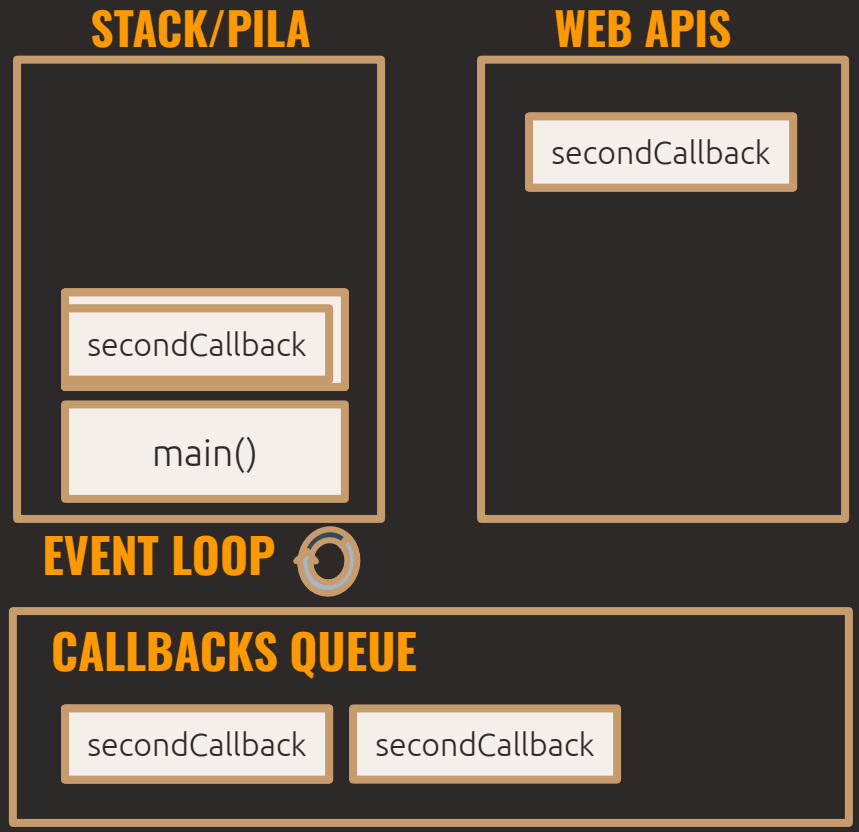
EVENT LOOP EN NAVEGADORES



EVENT LOOP EN NAVEGADORES

```
setTimeout(function firstCallback() {
 console.log('Primer log de SETTIMEOUT');
}, 0);

console.log('Primer log de FUERA');

setTimeout(function secondCallback() {
 console.log('Segundo log de
SETTIMEOUT');
}, 0);

console.log('Segundo log de FUERA');
```

```
//Primer log de FUERA
//Segundo log de FUERA
//Primer log de SETTIMEOUT
//Segundo log de SETTIMEOUT
```



PRIMITIVE OBSESSION / AIDA ALBARRÁN @aidaispro

EVENT LOOP

La magia de posibilitar hacer varias cosas “a la vez” y del mismo modo volvernos locos con la concurrencia...

¡GRACIAS!

¿Alguna pregunta?

@aidaispro
aidalbla@gmail.com