CICLO 4a

[FORMACIÓN POR CICLOS]

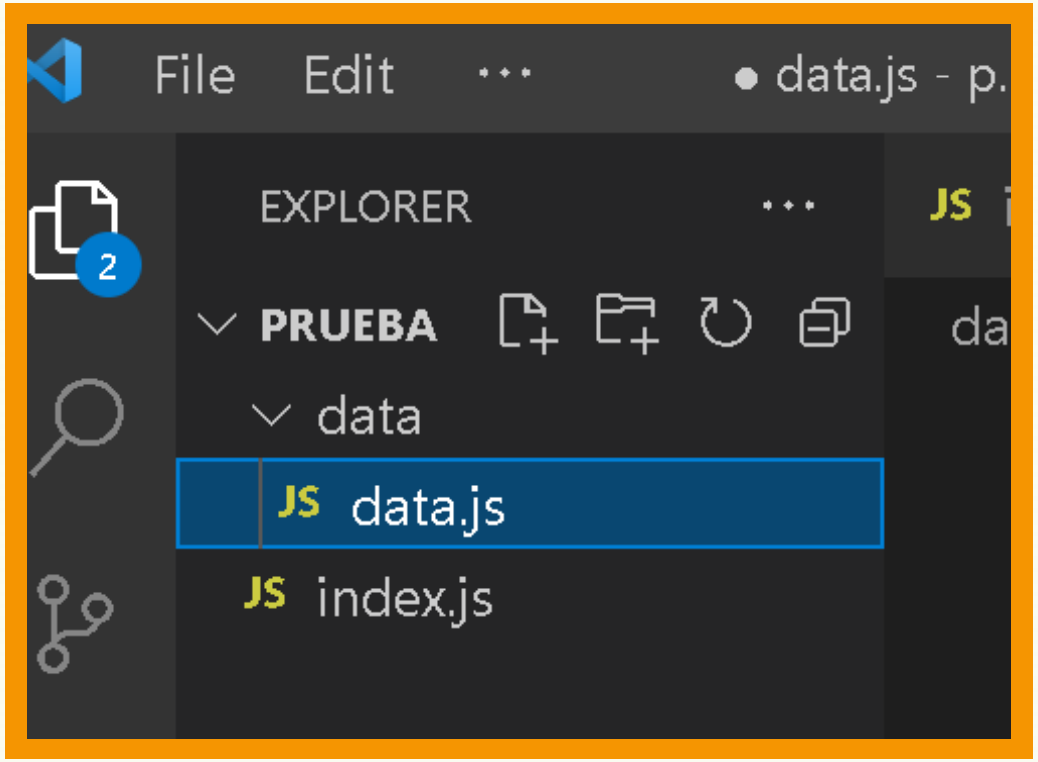
Desarrollo de APLICACIONES WEB

JavaScript
Import, export y funciones comunes de arreglos

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Para iniciar

Las importaciones y exportaciones las trabajaremos a partir de un ejemplo. Para iniciar, en esta semana descarga del curso el archivo data, copia y pega en un nuevo archivo .js:




```
//Archivo data.js
const heroes = [
    {
        id: 1,
        name: 'Batman',
        owner: 'DC'
    },
    {
        id: 2,
        name: 'Spiderman',
        owner: 'Marvel'
    },
    {
        id: 3,
        name: 'Superman',
        owner: 'DC'
    },
    {
        id: 4,
        name: 'Flash',
        owner: 'DC'
    },
    {
        id: 5,
        name: 'Wolverine',
        owner: 'Marvel'
    },
];
```

```
export const heroes = [
    {   id: 1,
        name: 'Batman',
        owner: 'DC'
    },
    {   id: 2,
        name: 'Spiderman',
        owner: 'Marvel'
    },
    {   id: 3,
        name: 'Superman',
        owner: 'DC'
    },
    {   id: 4,
        name: 'Flash',
        owner: 'DC'
    },
    {   id: 5,
        name: 'Wolverine',
        owner: 'Marvel'
    },
];
```

Exportar

Si lo que pretendemos es importar archivos o data JS, lo primero es exportar primero. Esto es muy sencillo: simplemente sobre el fichero de nuestra data vamos a usar la palabra reservada **export.**

Importar

Para trabajar con datos de otro archivo hay varias maneras. En este caso utilizamos en un archivo denominado index.js y arrancamos con la palabra reservada **import.**
En las importaciones no siempre terminamos con “;”, pero usarla es una buena práctica.

```
// Primera forma
import {data} from '/datos/data'
console.log(data);
```

Uso de data desde una función de flecha

```
//Find
const getHeroeById = (id)=>{
      return heroes.find(( heroe )=> heroe.id == id);
    }
    console.log(getHeroeById(2));

//Filter
const getHeroeByOwner = ( owner )=> heroes.filter(( heroe ) =>
heroe.owner == owner);

    console.log(getHeroeByOwner('DC'));
```

Se usará en una función flecha una predeterminada llamada **find**, que pretende mostrarnos cómo encontrar información en un archivo externo. Sin embargo, también podemos usar la función predeterminada filter, dependiendo del caso.

Exportaciones por defecto

Es posible también hacer exportaciones por defecto si tengo claro que todo el archivo será exportado y en este caso no es necesario indicar el nombre. En el archivo que importo puedo indicar cualquier nombre porque allí lo que importa es la ruta:

```
//index.js
import  superHeroes  from
'./datos/data';
```

Opción 1

```
export default [
    {   id: 1,
        name: 'Batman',
        owner: 'DC'
    },
    {   id: 2,
        name: 'Spiderman',
        owner: 'Marvel'
    },
    {   id: 3,
        name: 'Superman',
        owner: 'DC'
    },
    {   id: 4,
        name: 'Flash',
        owner: 'DC'
    },
    {   id: 5,
        name: 'Wolverine',
        owner: 'Marvel'
    },
];
```

Opción 2

```
const heroes = [
    {   id: 1,
        name: 'Batman',
        owner: 'DC'
    },
    {   id: 2,
        name: 'Spiderman',
        owner: 'Marvel'
    },
    {   id: 3,
        name: 'Superman',
        owner: 'DC'
    },
    {   id: 4,
        name: 'Flash',
        owner: 'DC'
    },
    {   id: 5,
        name: 'Wolverine',
        owner: 'Marvel'
    },
];
export default heroes;
```

Exportación individual

En una exportación por defecto es posible realizar una exportación individual. En mi archivo index puedo indicar entre llaves la exportación individual para que esta quede agregada:

```
import  superHeroes, {owners}  from
'./datos/data';
console.log( owners );
```

```
const heroes = [
    {   id: 1,
        name: 'Batman',
        owner: 'DC'
    },
    {   id: 2,
        name: 'Spiderman',
        owner: 'Marvel'
    },
    {   id: 3,
        name: 'Superman',
        owner: 'DC'
    },
    {   id: 4,
        name: 'Flash',
        owner: 'DC'
    },
    {   id: 5,
        name: 'Wolverine',
        owner: 'Marvel'
    },
];
Export const owners = [‘DC’,’Marvel’];
export default heroes;
```

Exportación individual 2

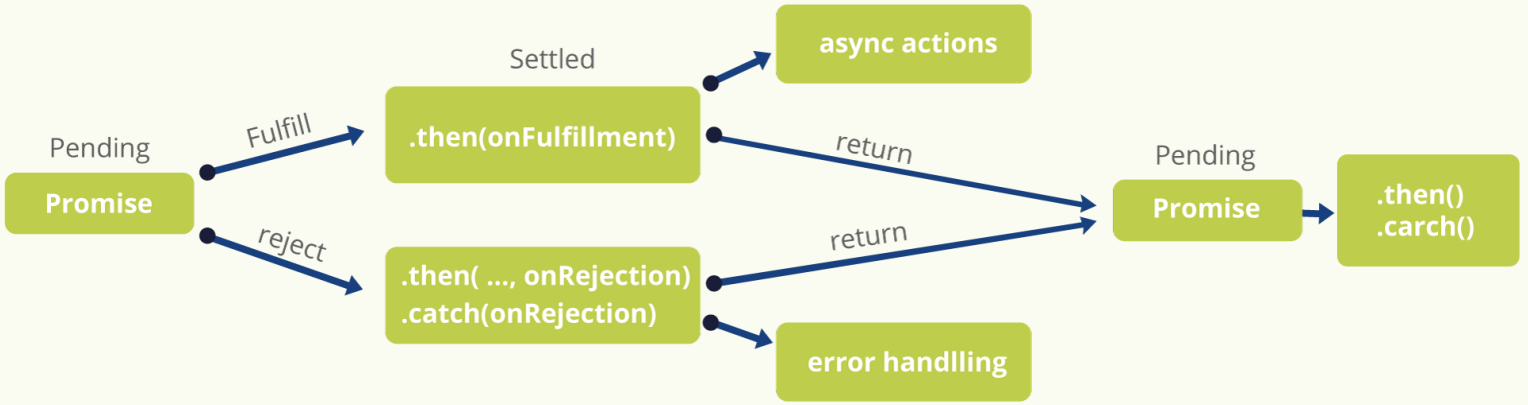
Hay muchas maneras de hacer importaciones y exportaciones. Otra de las usadas para este caso es:

```
import  superHeroes, {owners}  from
'./datos/data';
console.log( owners );
```

```
const heroes = [
    {   id: 1,
        name: 'Batman',
        owner: 'DC'
    },
    {   id: 2,
        name: 'Spiderman',
        owner: 'Marvel'
    },
    {   id: 3,
        name: 'Superman',
        owner: 'DC'
    },
    {   id: 4,
        name: 'Flash',
        owner: 'DC'
    },
    {   id: 5,
        name: 'Wolverine',
        owner: 'Marvel'
    },
];
const owners =['DC','Marvel'];
export {
heroes as default, owners};
```

Promesas

Las promesas en JavaScript
son acciones que se resolverán a futuro (cuando se pueda) y que sabremos si se llevaron a cabo con éxito o no:




```
const promesa = new Promise((resolve, reject)=>{
 setTimeout (()=> {
     resolve();
 }, 2000) //Función js que ejecuta tarea en cierto tiempo
 });
 promesa.then(()=>{
     console.log('Then de la promesa')//Then implica que la
promesa se hizo correctamente
    })
```

Fetch API

La API Fetch proporciona una interfaz para recuperar recursos (incluso a través de la red). Está basada en promesas y se puede usar tanto en el cliente como en el servidor:

```
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="utf-8" />
    <title>
        Repaso JavaScript
    </title>

         <!-- CSS only -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/boo
tstrap.min.css" rel="stylesheet" integrity="sha384-
gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1B
x" crossorigin="anonymous">
    <body>
    <h1>
        Aprendiendo JavaScript rápido
    </h1>
    <p>Hola soy Judy Moreno WEB</p>

    <div class="container text-center w-25 mt-4"></div>
    <form>
        <input class=" form-control" type="text"
placeholder="Pokemon...">
        <button type="submit" class="mt-2 btn btn-
primary">Buscar</button>

    </form>
    <div class='pokemon-container mt-4'></div>

    <script src="main.js" type="text/javascript"> </script>
</body>
</head>
</html>
```

Fetch API

La API Fetch proporciona una interfaz para recuperar recursos (incluso a través de la red). Está basada en promesas y se puede usar tanto en el cliente como en el servidor:

```
const input = document.querySelector("input");
const button = document.querySelector("button");
const pokemonContainer =
document.querySelector(".pokemon-container");
const url =
"https://pokeapi.co/api/v2/pokemon/pikachu/"

function traerPokemon(){
    fetch(url)
    .then((response) =>response.json())
    .then((data)=> {
        crearPokemon(data);
    });
   }


function crearPokemon(pokemon){
    const img = document.createElement("img");
    img.src = pokemon.sprites.front_default;

    const h3 = document.createElement("h3");
    h3.textContent= pokemon.name;

    const div = document.createElement("div");
    div.appendChild(img);
    div.appendChild(h3);

}
traerPokemon();
```

Async

Es una sintaxis especial para trabajar promesas, mucho más sencilla y mas fácil de entender:

Promesa normal

```
const getImagenPromesa = () => {
    const pomesa = new promise((resolve, reject)=> {
        resolve('https://2.bp.blogspot.com/-3d92ta4_JEc/TvmzTTgNBKI/AAAAAAAACJ4/GFC9bCwM5vQ/s1600/imagenes-jpg-712861.jpg')
    })
    return getImagenPromesa;
}

getImagenPromesa().then(console.log);
```

Async

```
const getImagen = async() => {
    return('https://2.bp.blogspot.com/-3d92ta4_JEc/TvmzTTgNBKI/AAAAAAAACJ4/GFC9bCwM5vQ/s1600/imagenes-jpg-712861.jpg')

}

getImagen().then(console.log);
```

Async - Await

Async funciona sin await, pero await no funciona sin el Async. Lo usamos cuando es necesario que espere a que una promesa termine antes de ejecutar la siguiente línea de código:

```
(async () => {
    let response = await fetch('/article/promise-
chaining/user.json');
    let user = await response.json();
    ...
  })();
```