<codoacodo/>

Agencia de
Aprendizaje
a lo largo
de la vida

Desarrollo Fullstack

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 24

Asincronismo en Javascript

- ¿Qué es?
- Call Stack
- Event Loop
- Promesas
- Async/Await

Clase 25

Solicitando info desde Javascript

- AJAX
- Fetch
- Template Strings
- Local Storage
- Repaso DOM

Clase 26

Node JS

- Servidor Web con Node
- Rutas básicas
- Método GET
- Enviar Datos
- Enviar Archivos

JAVASCRIPT

Pedir datos desde el cliente

Para el ejemplo de hoy vamos a trabajar con una API que nos brinda información sobre los personajes de la serie Rick y Morty.

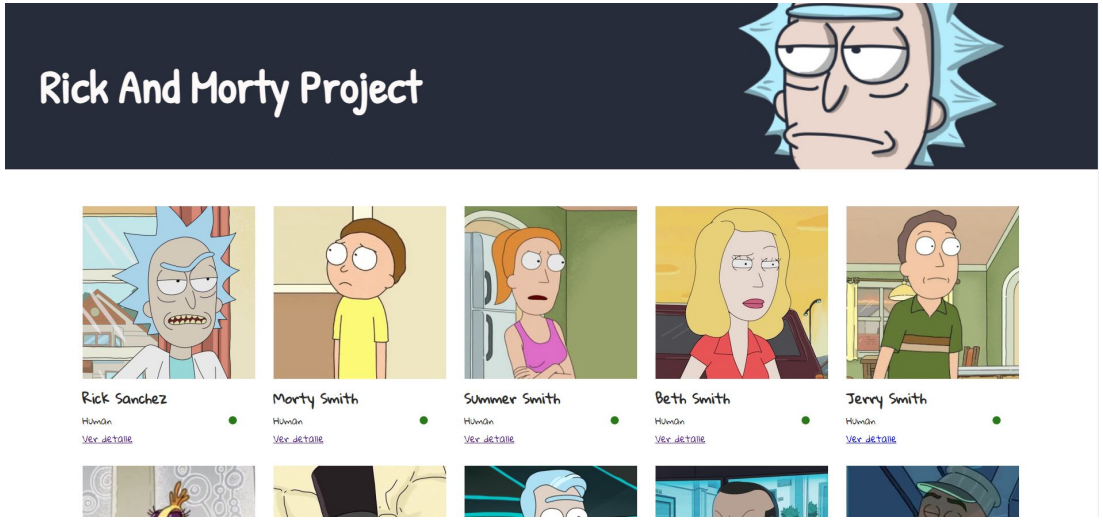
Podemos consultar su documentación en:

https://rickandmortyapi.com/documentation

Tomaremos como base la carpeta con archivos que se encuentra en el drive compartido.

Estructura

Nos encontramos con un archivo index.html, la idea es tomar los personajes de rick y morty y **pintarlos en la pantalla**, como en este ejemplo:



***Los estilos ya están creados, solo vamos a respetar algunas clases para poder aprovecharlos.**

Estructura

Luego, **al presionar ver detalle** nos va a redirigir a una página para ver ese personaje de forma detallada:

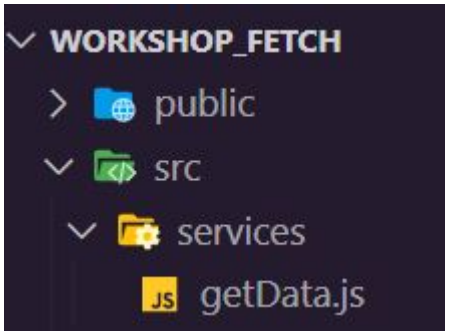
¡VAMOS AL DESAFÍO!

Primer paso, vamos a buscar la información.

Carpeta /src

Dentro de esta carpeta irá nuestra lógica javascript.

- Lo primero es crear una carpeta **services** y dentro un archivo llamado getData.js



Dentro de este archivo necesitamos crear 2 funciones.

- getCharacter(**id**) se ocupará de pedirle a la API que nos envíe el personaje solicitado.

  El parámetro id es el número de personaje que queremos.

- getCharacters(**page**) nos buscará todos los personajes según la página solicitada.

  El parámetro page es el número de página que queremos.

Buscamos los datos

```
// Declaramos una URL base que es la de la API
const baseURL = 'https://rickandmortyapi.com/api';

// Creamos la función asíncrona para ir a buscar un único personaje
const getCharater = async (id) => {
    const res = await fetch(`${baseURL}/character/${id}`);
    const data = await res.json(); // sacamos el body de la respuesta
    return data;
}

// Creamos la función asíncrona para ir a buscar todos los personajes
const getCharacters = async (page) => {
    const res = await fetch(`${baseURL}/character/?page=${page}`);
    const data = await res.json(); // sacamos el body de la respuesta
    return data;
}

export { getCharater, getCharacters };
```

No olvidemos **exportar** nuestras funciones, de esta manera podemos llamarlas desde otro archivo y tener nuestra lógica separada.

Ya tenemos las funciones que traen la info, ahora sigamos con home.

index.html

Tenemos un HTML prearmado que invoca un archivo javascript llamado “home.js”.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Proyecto - Ricky y Morty</title>
    <link rel="stylesheet" href="./public/css/index.css">
    <link rel="stylesheet" href="./public/css/loader.css">
</head>
<body>
    <header>
        <h1 class="main-title">Rick And Morty Project</h1>
    </header>
    <main id="characters">
        <div id="lds-ring" class="lds-ring"><div></div><div></div><div></div><div></div></div>

    </main>
    <footer>
        <p>Made with ❤️ by Codo a Codo - Curso de Fullstack NodeJS</p>
    </footer>
    <script src="./src/home.js" type="module"></script>
</body>
</html>
```

Tenemos un main donde colocaremos nuestros personajes mediante el uso del DOM.

El atributo **“module”** de la etiqueta <script> nos permite trabajar con módulos, separando nuestro código para **mayor organización**.

Ahora llenemos nuestro index de información.

Buscamos los datos - src/home.js

```
// Aquí necesitamos traer todos los personajes, por
// eso importamos nuestra función getCharacters
import { getCharacters } from "./services/getData.js";


// Tomamos el main container de nuestro Home y el loader
const container = document.querySelector('#characters');
const loader = document.getElementById('lds-ring');
```

- Traemos la función a utilizar.
- Accedemos al DOM y tomamos la etiqueta main#characters.
- A este elemento le agregaremos todos los personajes.

Además capturamos el elemento con el ID "lds-ring", es un ícono animado que dará el efecto de carga cuando la información demore demasiado.

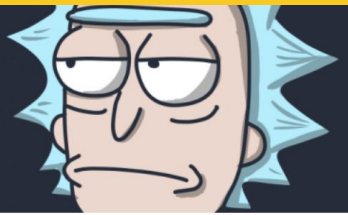
```
const charactersList = async (page = 1) => { // por defecto le pasamos el page 1 por si no lo recibe
    // mostramos el loader antes de llamar a la API
    loader.style.display = 'grid';
    // pedimos los personajes
    const { results } = await getCharacters(page);
    // ocultamos el loader una vez que ya tenemos la respuesta
    loader.style.display = 'none';
    results.forEach(character => { // por cada personaje creamos un article con sus datos
        const article = document.createElement('article');
        article.setAttribute('class', 'character');
        article.innerHTML = `
                <img src="${character.image}" alt="">
                <h2>${character.name}</h2>
                <div>
                    <p>${character.species}</p>
                    <p class="${character.status.toLowerCase()}"></p>
                </div>
                <a href="/#/${character.id}">Ver detalle // Al hacer click redirige a /#/idDelPersonaje
                </a>
        `;
       container.appendChild(article);
    });
}
```

Usamos los datos recibidos y creamos un elemento HTML nuevo para inyectarlo al DOM

Nos queda llamar a nuestra función para que la magia suceda.

```
charactersList();
```

Rick And Morty Project



Rick Sanchez

Human

Ver detalle

Morty Smith

Human

Ver detalle

Summer Smith

Human

Ver detalle

Beth Smith

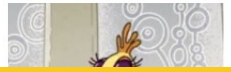
Human

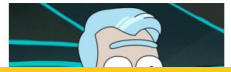
Ver detalle

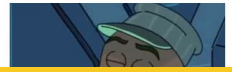
Jerry Smith

Human

Ver detalle







Sigamos con la página para cada personaje, pero... primero un detalle.

Reconociendo cambios en la URL

Para poder saber que personaje es el que tenemos que mostrar el detalle vamos a realizar un pequeño truco:

Cuando la URL cambie al presionar el enlace *"ver detalle"* de un personaje, el navegador reconoce el evento y guarda el ID de ese personaje en el localStorage para ser tomado por el archivo de details.js

```
// Escuchamos cambios en la URL, atentos a cuando hagan click en un "ver detalle"
window.addEventListener('hashchange', () => {
    // Si el enlace lleva a /#/3, id toma el valor 3 que es el ID del personaje
    const id = location.hash.slice(1).toLocaleLowerCase().split('/')[1] || '/';
    localStorage.setItem('charID', id);
    window.location.replace('/character.html');
});
```

localStorage es como una pequeña base de datos dentro del navegador, gracias a ella podemos persistir datos sencillos.

En este caso guardamos el ID del personaje con el nombre *"charID"* para usarlo en la página de detalle.

character.html

En este caso el HTML usado para cada personaje es muy similar a index, solo cambia un ID en main

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Detalle de Personaje</title>
    <link rel="stylesheet" href="public/css/index.css">
    <link rel="stylesheet" href="public/css/loader.css">
</head>
<body>
    <header>
        <h1 class="main-title">Rick And Morty Project</h1>
    </header>
    <main id="character">
        <div id="lds-ring" class="lds-ring"><div></div><div></div><div></div><div></div></div>

    </main>
    <footer>
        <p>Made with ❤️ by Codo a Codo - Curso de Fullstack NodeJS</p>
    </footer>
    <script src="./src/details.js" type="module"></script>
</body>
</html>
```

En este caso, vamos a crear un archivo llamado **“details.js”** y lo vamos a invocar desde nuestro archivo character.html

Buscamos los datos - src/details.js

```
// Aquí necesitamos traer el personaje solicitado, por eso importamos getCharacter.
import { getCharater } from "./services/getData.js";

// Guardamos en variables los elementos del DOM que vamos a utilizar
const container = document.querySelector('#character');
const loader = document.getElementById('lds-ring');

// Leemos el ID guardado en el localStorage, nos va a servir para traer los datos de este personaje.
const getID = localStorage.getItem('charID');
```

Buscamos los datos - src/details.js

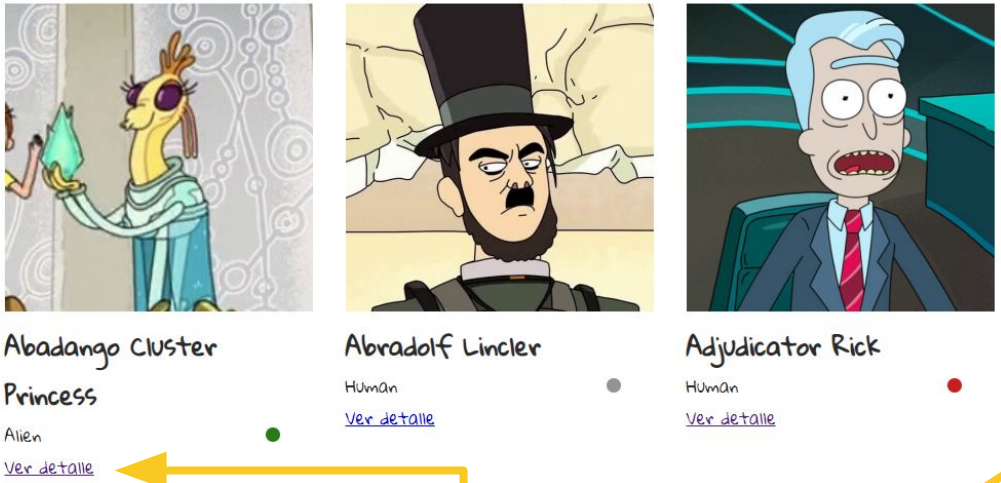
```
const loadCharacter = async (id) => {
    loader.style.display = 'grid';
        const data = await getCharater(id);
    loader.style.display = 'none';

    const article = document.createElement('article');
    article.setAttribute('class', 'character');
    article.innerHTML = `
            <img src="${data.image}" alt="">
            <h2>${data.name}</h2>
            <p class="data"><span>Origen:</span> ${data.origin.name}</p>
            <p class="data"><span>Locación Actual:</span> ${data.location.name}</p>
            <div>
                <p class="data"><span>Especie:</span> ${data.species}</p>
                <p class="${data.status.toLowerCase()}"></p>
            </div>
    `;
   container.appendChild(article);
}

loadCharacter(getID);
```

Similar a home.js, tenemos una **función asíncrona** que recibe el ID del personaje y lo pide a **getCharacter(id)**.

Una vez que recibe la información crea una etiqueta article y le **inyecta el código HTML** para agregarle la información del personaje.

¡Just Magic!

Ahora al hacer click en “ver detalle” nos lleva a la página character.html pero muestra la información del personaje que clickeamos.



Todo esto, con solo 2 archivos HTML que se completan de forma dinámica!

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

Todo en el Aula Virtual.

Gracias