**REACT**

**Usando la librería de React desde el CDN:**

<!-- Cargat React -->

<script crossorigin src="https://unpkg.com/react@16/umd/react.production.min.js"></script>

<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>

<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>

Babel permite utilizar características de JavaScript. Es un compilador de este lenguaje que ayuda a que el código funcione en los diferentes navegadores.

**Crear React App**

*En la terminal:*

npx create-react-app myApp

cd myApp

npm start

**Spread operator …**

….

Para crear copias.

Ayuda a no modificar el objeto original.

Promesas:

/\*/ Trabaja con Fetch (peticiones Http)

* Las promesas pueden encadenarse. Conocidas como promesa en cadena

promesa

.then()

.then()…

* .catch( console.warn ) 🡪 devolver el error de la petición

Async:

* Devuelve una promesa y utilizamos menos código cuando generamos una nueva promesa.

*const* getImagen = *async*() => {

*return* "Hola jostin";

}

getImagen().then( console.log )

* Puede estar independiente del await.

*const* getImagen = *async*() => {

    try {

*const* apiKey = 'OchQvdWT1TUY5upq3PjLNKG13lQqechx';

*const* response =  *await* fetch(`https://api.giphy.com/v1/gifs/random?api\_key=${ apiKey }`);

*const* { data } = *await* response.json();

*const* { url } = data.images.original;

*const* img = document.createElement('img');

        img.src = url;

        document.body.append(img);

    } catch (error) {

        console.log(error);

    }

}

getImagen();

**QUE SON LOS COMPONENTES?**

* Pequeña pieza de código encapsulada re-utilizable que puede tener estado o no.

**PropTypes:**

* Indicar que se debe insertar el tipo que se especifica en la variable, en este caso la variable saludo solo va aceptar datos del tipo String.
* isRequired 🡪 para obligar a escribir algo como parámetro

*//El saludo debe ser del tipo string*

PrimeraApp.propTypes = {

    saludo: PropTypes.string.isRequired

}

Mensaje de error, aunque no te afecta mucho en general, no es obligatorio mandar tal tipo.

ReactDOM.render( <PrimeraApp saludo={123} /> , divRoot );

**DefaultProps:**

*//propiedades por defecto*

*//otro método, en lugar de colocarlo en la linea 4 al costado del parámetro*

PrimeraApp.defaultProps = {

    subtitulo : 'Soy un subtitulo'

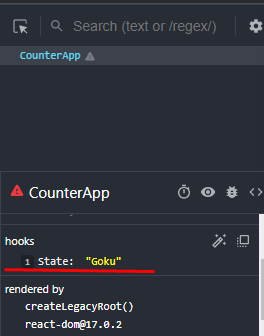
}

* se mostrará en la página si es que no le paso el dato en el js principal, en la etiqueta del componente.

Crear un componente con PropTypes : rafcp

Hooks:

* es una función
* *//useState toma 2 argumentos, valor y funcion*
* *//valor, setValor*
* *const* state = useState('Goku');



**Introducción a las pruebas**

**Tipos:**

1. UNITARIAS: Enfocadas en pequeñas funcionalidades.
2. INTEGRACIÓN: Enfocadas en como reaccionan varias piezas en conjunto.

**Características:**

1. Fáciles de escribir
2. Fáciles de leer
3. Confiables
4. Rápidas
5. Principalmente unitarias

**AAA:**

**Arrange-Arreglar: preparamos el estado inicial**

1. Inicializamos variables
2. Importaciones necesarias

**Act-Actuar: aplicamos acciones o estimulos**

1. **Llamar métodos**
2. **Simular clicks**
3. **Realizar acciones sobre el paso anterior**

**Assert-Afirmar: observar el comportamiento resultante**

1. **Son los resultados esperados.**
2. **Ej: que algo cambie, algo incremente o bien que nada suceda.**

**Testing with JEST: (javascript testing framework)**

* **Verificar si se cumplen condiciones de un valor con el valor esperado**

**expect( p1 ).toBe(p2);**

import '@testing-library/jest-dom';

import { getSaludo } from '../base/02-template-string';

describe('Pruebas en el 02-template-string.test.js', () => {

    test('Debe de retornar Hola Jostin', () => {

*const* nombre = 'Jostin';

*const* saludo = getSaludo(nombre);

        expect( saludo ).toBe('Hola '+ nombre);

    });

*//getSaludo debe retornar Hola Carlos si no hay dato en el nombre.*

    test('Debe retornar Hola Carlos', () => {

*const* saludo = getSaludo();

        console.log(saludo);

        expect(saludo).toBe('Hola Carlos');

    });

});

* **Verficar igualdad en objetos, comparar las propiedades del objeto.**

**Expect(p1).toEqual(p2);**

import { getUser } from "../base/05-funciones";

describe('Pruebas en 05-Funciones', () => {

    test('getUser debe retornar un objeto', () => {

*const* userTest = {

            uid : 'ABC123',

            username: 'El\_Papi1502'

        }

*const* user = getUser();

        expect( user ).toEqual( userTest );

    });

});