**JavaScript:**

*const* nombre = 'sebi';

*const* apellido = 'Epstein';

//const nombreCompleto = nombre + ' ' + apellido;

*const* nombreCompleto = ` ${nombre} ${apellido}`;

**Clonar un objeto:**

*const* persona = {

    nombre: 'tony',

    apellido: 'Stark',

    edad: 45,

    direccion: {

        ciudad: 'tucuman',

        calle: '25 de mayo',

        numero: 710,

    },

};

*const* persona2 = { ...persona }; // Clono el objeto de persona en persona2

persona2.nombre = 'Pedrito' // Cambio el nombre de persona2 SOLAMENTE

console.log( persona );

console.log( persona2 );

**Ese es el operador SPREAD**

**Arreglo en JS:**

//const arreglo = new Array();

*const* arreglo = [1,2,3,4];

//arreglo.push(1);

*let* arreglo2 = [...arreglo, 5];

console.log( `este es el arreglo 1: ${arreglo}` );

console.log( `este es el arreglo 1: ${arreglo2}` );

**Función MAP()**

*const* arreglo2 = [1,2,3,4];

*const* arreglo3 = arreglo2.map(*function*(*numero*){

 return numero \* 2;

});

**Funciones**

// FUNCION CON NORMAL

*const* saludar = *function* (*nombre*) {

    return `Hola ${ nombre }`;

}

// FUNCION FLECHA

*const* saludar2 = (*nombre*) *=>* {

    return `Hola ${ nombre }`;

}

// FUNCION FLECHA LIGERA

*const* saludar3 = (*nombre*) *=>* `Hola ${ nombre }`;

// FUNCION FLECHA CON RETORNO DE OBJETO

*const* getUser = () *=>* ({

       uid: 'ABC12345',

        username: 'usuario1',

    });

console.log( saludar("sebi"));

console.log( saludar2("sebi"));

console.log( saludar3("sebi"));

console.log( getUser());

**Desestructuración**

*const* persona = {

    nombre: 'tony',

    edad: 45,

    clave: 'ironman',

};

*const* { nombre, edad, clave } = persona;

console.log( nombre );

console.log( edad );

console.log( clave );

*const* persona = {

    nombre: 'tony',

    edad: 45,

    clave: 'ironman',

    rango: 'soldado',

};

*const* retornaPersona = ({ *nombre*, *edad*, *rango* = 'capitan' }) *=>* {

    console.log(edad, nombre, rango);

}

retornaPersona( persona );

*const* persona = {

    nombre: 'tony',

    edad: 45,

    clave: 'ironman',

    rango: 'soldado',

};

*const* Prueba = ({ *clave*, *nombre*, *edad*, *rango* = 'capitan' }) *=>* {

    //console.log(edad, nombre, rango);

    return {

        nombreClave: clave,

        anios: edad,

        latIng:{

            lat: 14.2552,

            lng: -12.4425

        }

    }

}

*const* {nombreClave, anios, latIng:{lat, lng}} = Prueba( persona );

console.log(nombreClave, anios);

console.log(lat, lng);

# **Filter Y Find**

import { heroes } from './data/heroes';

*const* getHeroeById = (*id*) *=>* {

    return heroes.find( (*heroes*) *=>* id === heroes.id);

}

//console.log( getHeroeById(4) );

*const* getHeroeByOwner = ( *owner* ) *=>* heroes.filter( (*heroes*) *=>* owner === heroes.owner);

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

# **Promesa:**

*const* promesa = new *Promise*( (*resolve*, *reject*) *=>* {

    setTimeout(() *=>* {

*const* p1 = getHeroeById(2);

       // console.log(heroe);

        //getHeroeByOwner();

        resolve( p1 );

    }, 2000);

});

Promesa con Id del héroe

*const* getHeroeByIdAsync = ( *id* ) *=>* {

    return new *Promise*( (*resolve*, *reject*) *=>* {

        setTimeout(() *=>* {

*const* p1 = getHeroeById(22);

           // console.log(heroe);

            //getHeroeByOwner();

            if( p1 ){

                resolve( p1 );

            } else {

                reject('No se puede encontrar el heroe');

            }

        }, 2000);

    });

}

getHeroeByIdAsync(4)

    .then( *heroe* *=>* console.log('Heroe', heroe))

    .catch( *err* *=>* console.warn( err ))

**Forma más fácil de then y catch**

getHeroeByIdAsync(4)

    .then( console.log /\*heroe => console.log('Heroe', heroe)\*/)

    .catch( console.warn /\*err => console.warn( err )\*/);

# **Fetch**

*const* apiKey = 'xOEnsoKrVNzdRsi94IkBuB9PabY4Sv0S'

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

peticion

.then( *resp* *=>* resp.json() )

.then( ({*data*} ) *=>* {

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

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

    img.src= url;

    document.body.append( img );

})

.catch( console.warn);

# **Async Await**

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

  try {

*const* apiKey = "xOEnsoKrVNzdRsi94IkBuB9PabY4Sv0S";

*const* resp = await fetch(

      `https://api.giphy.com/v1/gifs/random?api\_key=${apiKey}`

    );

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

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

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

    img.src = url;

    document.body.append(img);

  } catch (error) {

      // MANEJO DEL ERRROR

      console.error( error )

  }

};

getImagen();

# **Operación Ternaria:**

*const* mensaje = (activo) ? 'Activo' : 'Inactivo'; // ESPERA EL IF Y ELSE

*const* mensaje = activo && 'Activo' ; //SOLO ESPERA ENTRAR AL IF

**REACT**

# **Para crear una REACT-APP desde la terminal:**

npx create-react-app NombreDeLaApp

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamentenode\_modules**

* Tiene las librerías y los paquetes que hace que la aplicación REACT funcione
* No se la modifican manualmente.
* Para modificar es con comando para instalar algo.

**Texto

Descripción generada automáticamentePublic**

* Favicon.ico: el icono de REACT
* Manifest.json: se ven dónde están los logos. NADA que ver con REACT
* Robots.txt: configuración por defecto, no es importante. NADA que ver con REACT

**.gitignore**

* Es un archivo que se maneja por la parte de git

**package.json**

* Casi nunca se toca.

**README.md**

* Es un archivo que describe la aplicación.

Texto

Descripción generada automáticamente con confianza baja**src**

* Es en donde mas vamos a pasar el tiempo.
* .test.js donde se realizan las pruebas

# **Hola mundo en REACT**

import React from 'react'

import ReactDOM from 'react-dom'

*const* saludo = <h1>Hola mundo</h1>;

*const* divRoot = document.querySelector('#app');

ReactDOM.render( saludo, divRoot )

console.log(divRoot);

# **Renderizar algo en REACT**

**Index.js**

import React from 'react'

import ReactDOM from 'react-dom/client'

import PrimeraApp from './PrimeraApp';

import './index.css'

*const* divRoot = ReactDOM.createRoot(document.getElementById("app"));

divRoot.render(<*PrimeraApp* />);

**PrimeraApp.js**

import React from 'react'

// componente en una funcion FC

*const* PrimeraApp = () *=>* {

    return <h1>Hola mundo</h1>;

};

export default PrimeraApp;

# **Renderizar mas de una cosa:**

**Opción A: Fragment**

import React, { Fragment } from 'react'

// componente en una funcion FC

*const* PrimeraApp = () *=>* {

    return (

        <*Fragment*>

            <h1>Hola mundo</h1>

            <p> Primer Parrafo</p>

        </*Fragment*>

    );

};

export default PrimeraApp;

**Opción B:**

import React from 'react'

*const* PrimeraApp = () *=>* {

    return (

        <>

            <h1>Hola mundo</h1>

            <p> Primer Parrafo</p>

        </>

    );

};

export default PrimeraApp;

# **Impresión de variables en el HTML**

*const* PrimeraApp = () *=>* {

*const* saludo = 'Hola mundo'

*const* persona = {

        nombre: 'sebi',

        apellido: 'Epstein',

    }

    return (

        <>

            <h1> { saludo } </h1>

            <pre> { JSON.stringify(persona, null, 3) } </pre>

            <p> Primer Parrafo</p>

        </>

    );

};

export default PrimeraApp;

# **Comunicación entre componentes y Props**

**Index.js:**

import React from 'react'

import ReactDOM from 'react-dom/client'

import PrimeraApp from './PrimeraApp';

import './index.css'

*const* divRoot = ReactDOM.createRoot(document.getElementById("app"));

divRoot.render(<*PrimeraApp* saludo="hola soy sebi" />);

**PrimeraApp.js:**

//import React, { Fragment } from 'react'

// componente en una funcion FC

import React from 'react'

*const* PrimeraApp = ( {*saludo* = 'hola perro'} ) *=>* {

    return (

        <>

            <h1> { saludo } </h1>

            {/\*<pre> { JSON.stringify(persona, null, 3) } </pre>\*/}

            <p> Primer Parrafo</p>

        </>

    );

};

export default PrimeraApp;

# **Validacion de Props**

import React from 'react';

import PropTypes from 'prop-types';

// IMPORTO PropTypes PARA PODER USARLO

*const* PrimeraApp = ( { *saludo* } ) *=>* {

    return (

        <>

            <h1> { saludo } </h1>

            <p> Primer Parrafo</p>

        </>

    );

};

PrimeraApp.propTypes = {

    saludo: PropTypes.string.isRequired

    // Obligo a que saludo sea obligatorio y string

}

# **Props por defecto:**

**Opción A:**

*const* PrimeraApp = ( { *saludo*, *subtitulo* = 'Opcion A' } ) *=>* {

    return (

        <>

            <h1> { saludo } </h1>

            <p> { subtitulo }</p>

        </>

    );

};

**Opción B:**

PrimeraApp.defaultProps = {

 subtitulo: 'Soy un subtitulo',

}

# **Rafcp + tabulador**

import React from 'react'

import PropTypes from 'prop-types'

*const* CounterApp = *props* *=>* {

  return (

    <div>CounterApp</div>

  )

}

CounterApp.propTypes = {}

export default CounterApp

# **Evento del mousse: onClick**

import React from 'react'

import PropTypes from 'prop-types'

    // HandleAdd

*const* handleAdd = (*e*) *=>* {

        console.log(e)

    }

*const* CounterApp = ( { *value* } ) *=>* {

    return (

      <>

        <h1>CounterApp</h1>

        <h2> {value} </h2>

        <button onClick={ handleAdd }>+1</button>

      </>

    )

};

# **UseState y Hook**

import React, { useState } from 'react'

import PropTypes from 'prop-types'

*const* CounterApp = ( { *value* } ) *=>* {

*const* [ counter, setCounter ] = useState(0);

    // HandleAdd

*const* handleAdd = (*e*) *=>* {

        setCounter( counter + 1);

        setCounter( (*c*) *=>* c + 1 );

    }

    return (

      <>

        <h1>CounterApp</h1>

        <h2> { counter } </h2>

        <button onClick={ handleAdd }>+1</button>

      </>

    )

};

# **Pruebas**

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

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

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

    test('getSaludo debe de retornar hola sebi', () *=>* {

*const* nombre = 'sebi';

*const* saludo = getSaludo(nombre);

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

     })

});