**REPASO DE LA CLASE 1**

**LIBRERIAS:** Reack  
 ReackDom

**METODOS:** .createElement() + argumentos

.render().() + argumentos

**LINKS CDN**

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

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

**FORMAS DE DECLARAR VARIABLES**

var: variable estandar. Funciona en varios ambitos o bloques

var x = 1; No se usa

let: Funciona solo dentro del bloque de codigo

let x = 1;

const: Declara una constantes la cual no va a poder modificarse

const x = 1;

x = 2;

**DECLARAR FUNCIONES**

function myFunction(){}

const myFunction = function(){}

const myFunction = () => {} Mas se usa

**METODOS DE JAVASCRIPT QUE VAMOS A USAR CON REACT**

.map() \*

.filter() \*

.slice()

.push()

.pop()

.sort()

.reduce()

**.map()** Itera sobre cada elemento del array original, aplica la función callback a cada uno y crea un nuevo array con con los resultados de la transformación

Sintaxis:

array.map(funcion(elemento, indice, array) {

    // Lógica de transformación })

const numeros = [1, 2, 3, 4, 5];

const multiplicados = numeros.map((numero) => {

  return numero \* 2;

});

console.log(multiplicados); // Resultado: [2, 4, 6, 8, 10]

**.filter()** Se utiliza para filtrar elementos de un array según ciertos criterios y crear un nuevo array con los elementos que cumplen con esos criterios

Sintaxis:

array.filter(funcion(elemento, indice, array) {

    // Lógica de filtrado })

const numeros = [1, 2, 3, 4, 5];

const numerosPares = numeros.filter((numero) => {

  return numero % 2 === 0;

});

**.slice()** Se utiliza para extraer una porción de una cadena o matriz existente

Sintaxis: cadena.slice(inicio, fin)

const cadena = "Hola mundo";

const subcadena = cadena.slice(0, 4);

console.log(subcadena); // Resultado: "Hola"

const numeros = [1, 2, 3, 4, 5];

const subarray = numeros.slice(1, 4);

console.log(subarray); // Resultado: [2, 3, 4]

**.push()** Se utiliza para agregar uno o más elementos al final de un array existent

Modifica el array original

Sintaxis: array.push(elemento1, elemento2, ..., elementoN)

const numeros = [1, 2, 3];

numeros.push(4);

console.log(numeros); // Resultado: [1, 2, 3, 4]

**.pop()** Se utiliza para eliminar el último elemento de un array y devolver ese elemento eliminado.

Puedes usarlo para reducir la longitud de un array y obtener el elemento eliminado.

Sintaxis: array.pop()

const numeros = [1, 2, 3, 4, 5];

const ultimoElemento = numeros.pop();

console.log(numeros); // Resultado: [1, 2, 3, 4]

console.log(ultimoElemento); // Resultado: 5

**.sort()** Se utiliza para ordenar los elementos de un array en su lugar

uedes usarlo para ordenar alfabéticamente o numéricamente los elementos de un array.

Sintaxis: array.sort()

const frutas = ["manzana", "banana", "naranja", "kiwi"];

frutas.sort();

console.log(frutas); // Resultado orden alfabético: ["banana", "kiwi", "manzana", "naranja"]

const numeros = [5, 2, 10, 1, 8];

numeros.sort(function(a, b) {

  return b - a;

});

console.log(numeros); // Resultado Descendente: [ 10, 8, 5, 2, 1 ]

**.reduce()** Se utiliza para reducir los elementos de un array a un único valor

Puedes usarlo para realizar operaciones de suma, resta, concatenación u otras operaciones de reducción en un array.

Esta función puede tomar hasta 4 argumentos:

Acumulador: -------------- El valor acumulado por la suma de los elementos

elemento: ----------------- El elemento actual que se está procesando en el array.

indice: (opcional) -------- El índice del elemento actual en el array.

array: (opcional) --------- El array original en el que se está aplicando la reducción.

valorInicial: (opcional) -- El valor inicial del acumulador. Por defecto se usa el primer elemento del array

Sintaxis:

array.reduce(funcion(acumulador, elemento, indice, array) {

    // Lógica de reducción

}, valorInicial)

const numeros = [1, 2, 3, 4, 5]; // 1 + 2 + 3 + 4 + 5 = 15

const sumaTotal = numeros.reduce(function(acumulador, elemento) {

  return acumulador + elemento;

}, 0);

console.log(sumaTotal); // Resultado: 15

**METODOS DE DOCUMENT**

document.querySelector():

document.querySelectorAll()

document.getElementById() \*

document.getElementsByClassName()

document.getElementsByName()

document.addEventListener()

document.querySelector(): Se utiliza para seleccionar elementos en una página web a traves de un selector.

Devolver solo el primer elemento que encuentra

// Selecciona el primer elemento con la clase "miClase"

var elemento = document.querySelector('.miClase');

// Cambia el color de fondo del elemento seleccionado

elemento.style.backgroundColor = 'red';

document.querySelectorAll(): Se utiliza para seleccionar múltiples elementos en una página web y devuelve una lista de esos elementos.

// Selecciona todos los elementos con la clase "miClase"

var elementos = document.querySelectorAll('.miClase');

// Recorre la lista de elementos y cambia su color de fondo

elementos.forEach(function(elemento) {

  elemento.style.backgroundColor = 'blue';

});

document.getElementById(): Se utiliza para seleccionar un elemento específico en una página web utilizando su identificador único, ID.

// Selecciona el elemento con el ID "miElemento"

var elemento = document.getElementById('miElemento');

// Cambia el contenido de texto del elemento seleccionado

elemento.textContent = '¡Hola, mundo!';

document.getElementsByClassName(): se utiliza para seleccionar elementos en una página web basándose en su clase.

// Selecciona todos los elementos con la clase "miClase"

var elementos = document.getElementsByClassName('miClase');

// Recorre la lista de elementos y cambia su color de fondo

for (var i = 0; i < elementos.length; i++) {

  elementos[i].style.backgroundColor = 'red';

}

document.getElementsByName(): utiliza para seleccionar elementos en una página web basándose en su atributo "name"

    <body>

        <input type="text" name="miNombre" placeholder="Ingresa tu nombre">

      </body>

// Selecciona todos los elementos con el atributo "name" igual a "miNombre"

var elementos = document.getElementsByName('miNombre');

// Recorre la lista de elementos y cambia su contenido de texto

for (var i = 0; i < elementos.length; i++) {

  elementos[i].textContent = '¡Hola!';

}

document.addEventListener(): Se utiliza para agregar un "escuchador" o "listener" a un elemento del documento

Permite detectar cuando ocurre un evento en ese elemento y realizar una acción en respuesta

La funcion, toma dos argumentos principales:

1. Tipo de evento que se desea escuchar
2. Función que se ejecutará cuando ocurra el evento

Puedes utilizar diferentes tipos de eventos:

Eventos de ratón:

'click': se dispara cuando se hace clic en el elemento.

'mouseover': se dispara cuando el puntero del ratón se sitúa sobre el elemento.

'mouseout': se dispara cuando el puntero del ratón sale del elemento.

'mousemove': se dispara cuando se mueve el puntero del ratón sobre el elemento.

Eventos de teclado:

'keydown': se dispara cuando una tecla se mantiene presionada.

'keyup': se dispara cuando una tecla se suelta.

'keypress': se dispara cuando se presiona una tecla y se mantiene presionada.

Eventos de formulario:

'submit': se dispara cuando se envía un formulario.

'input': se dispara cuando se introduce un valor en un campo de entrada.

'change': se dispara cuando cambia el valor de un campo de entrada.

Eventos de carga:

'load': se dispara cuando se completa la carga de un recurso, como una imagen o un archivo externo.

'DOMContentLoaded': se dispara cuando se completa la carga inicial del documento HTML.

// Selecciona un elemento de botón

var boton = document.querySelector('#miBoton');

// Agrega un escuchador de evento de clic al botón

boton.addEventListener('click', function() {

  console.log('¡Haz hecho clic en el botón!');

});

**# Babel y JSX**

Babel: Herramienta que convierte el codigo JS moderno en versiones compatibles con navegadores y entornos mas antiguos, permitiendo a los desarrolladores utilizar caracteristicas mas avanzadas sin preocuparce por la compatibilidad

JSX: Extension de JS que se utiliza en aplicaciones web utilizando la biblioteca de react

permite escribir codigo HTML-like dentro del codigo JS.

JSX se compila en codigo JS antes de ser ejecutado por el navegador

**EjemploPractico**

<!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>Document</title>

    <script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script> <!--React-->

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

</head>

<body style="background-color: black; color: white;">

    <div id="root"> <!--Padre de los elemento que vamos a renderizar-->

    </div>

    <!-- <script type="text/javascript">

        <script type="text/babel">

            Codigo JS…….

        </script> -->

    <!--<script type="text/javascript" src="/02\_Clase/Clase\_a.js"></script>-->

    <script type="text/babel" src="/02\_Clase/Clase\_a.js"></script> <!--Cambiamos /javascript por /babel-->

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

</body>

</html>

const lista = [

    "El pianista",

    'Titanic',

    'Avatar',

    'Avengers',

    'Mario Brhos',

]

Recorre la lista y devuelve un <ul> con la lista mapeada de <li>peliculas</li>

    const Peliculas = () => { // Recorre la lista y devuelve un <ul> con la lista mapeada de <li>peliculas</li>

        let peliculas = lista.map( (peli, i) => {

            // a. REACT - (Mapeo de lista)

            //return React.createElement('li', null, i+'. '+peli)

            // b. BABEL - (Mapeo de lista)

            return <li>{i +'. '+peli}</li> // JavaScript <-(antes) return (despues)-> html {JavaScript}

        });

        //return React.createElement('ul', null, lista) // retorna Array de pelicula - El pianistaTitanicAvat..

        //a. REACT - (Retorno de lista)

        //return React.createElement('ul', null, peliculas)

        // b. BABEL - (Retorno de lista)

        return <ul>{peliculas}</ul> // JavaScript <-(antes) return (despues)-> html {JavaScript}

        /\*

            0. El pianista

            1. Titanic

            2. Avatar

            3. Avengers

            4. Mario Brhos

        \*/

    }

**.redenr():** Se utiliza para generar la representación visual o la salida de un objeto o componente, permitiendo que los usuarios vean y interactúen con la información de una manera comprensible y atractiva.

Acepta dos argumentos

. El contenido -> Peliculas()

. El contenedor -> <div id='root'></div>

    ReactDOM.render(

        //Peliculas(), // Invocacion de funciones

        //<Peliculas></Peliculas>, // Etiquetas personalizadas (dobles)

        <Peliculas/>, // Etiqueta personalizada (unica) para ReactDOM: Componente de funcion

<peliculas/>,

        document.getElementById('root') // \*Obtenemos el elementos con el id root (Contenedor)

    )

// JSX: Retorna elementos de JS Y HTML

// Babeljs.io -> Try it out

// Los componentes a diferencia de las funciones se escriben con MAYUSCULA en la primer letra <Peliculas/>,, ya que si se coloca con minuscula lo toma como etiqueta HTML <peliculas/>,, y no JSX

**create.react-app:**

Comando que crea una aplicacion de React con una configuracion predeterminada

Para crear el proyecto es necesario tener:

Node.js - Entorno de ejecucion de JavaScript (Lado Backend)

Verificar instalacion: node -v

Recomendado: version 12+ ( Actualizar)

npm (Node Package manager) - Se encarga de gestionar los paquetes de node

Verificar instalacion: npm -v

Recomendado: version 5.2+

**# COMANDOS DE INSTALACION O ACTUALIZACION DE PAQUETES**

01. npm: Maneja los packetes de node instalados, y si no los tiene los instala

**npm init react-app my-app**

02. npx: Ejecuta los paquetes de Node.js

**npx create-react-app my-app**

03. Yarn: Maneja los packetes de node de manera distinta a npm pero es lo mismo

**yarn create react-app my-app**

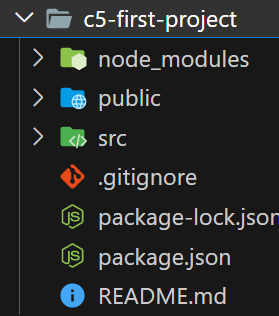
# Nombre del proyecto: Puede ser cualquiera pero SIN mayuscula ni espacios

**# INSTALACION en la terminal:**

1. **cd /Desktop/Frontend\_3/02\_Clase/Clase2**
2. **npx create-react-app c5-first-project**
3. **cd c5-first-project**
4. **npm start**

**LocalHost Server:** [**http://localhost:3000/**](http://localhost:3000/)

**# ARCHIVOS Y DIRECTORIO DE CRA**



**node\_modules:** Contiene todas las bibliotecas y paquetes de software de terceros que tu proyecto necesita para funcionar.

**public:** React contiene los archivos estáticos que se utilizarán al construir y ejecutar tu aplicación

**vite.svg**

**src:** Contiene la mayor parte del código fuente de tu aplicación. Aquí se encuentran los componentes de React, Archivos JavaScript, Archivos CSS o estilos, archivos de imagen y otros recursos necesarios tu aplicación

**> assets: Guarda las imágenes**

**> App.css**

**> App.jsx**

**> Index.css**

**> Main.jsx**

**.eslintrc.cjs**

**.gitignore:** Especifica los archivos y directorios que deben ser ignorados y no deben ser rastreados por el control de versiones

**Index.html**

**package-lock.json:**

Este archivo se utiliza para asegurar que todas las personas que trabajen en el proyecto tengan las mismas versiones de las dependencias instaladas, evitando problemas de compatibilidad.

Se actualiza automáticamente cada vez que se agregan, eliminan o actualizan las dependencias del proyecto garantizando que todos trabajen con las mismas versiones de dependencia

Contiene informacion detallada sobre las dependencia que el proyecto necesita

**name:**

**version**

**dependencias**

**arbol de dependencia y como estas se relacionan**

**package.json:**

Contiene la informacion del proyecto como dependencias requeridas, los comandos personalizados y la configuración específica

. Nombre del proyecto

. Version del proyecto

. Dependencia

. Script: Permite definir comandos personalizados en el archivo package.json que te permiten ejecutar. Ej: ejecutar pruebas

. Autor y licencia

. Configuración de webpack

**vite.config.js**

**ACTUALIZAR LIBRERIAS PARA QUE FUNCIONEN CORRECTAMENTE**

**Npm:** Se encarga de comunicar (package-lock.json y package.json) con node\_modules para saber que le hace falta al proyecto para funcionar correctamente

**> node\_modules**

**> package-lock.json**

**> package.json**

1. cd c5-first-project
2. npm install

**LOS ARCHIVOS MAS IMPORTANTES SON:**

****

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

**<div id="root"></div>**

  </body>

</html>

****

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  //<React.StrictMode>

    <App /> // Renderiza lo que se encuentra en el archivo app

  //</React.StrictMode>

);

reportWebVitals();



App.js o App.jsx (Es lo mismo)

Lo que sea que retorne con create.react-app lo va a tomar igual

En este archivo es donde vamos a trabajar

import logo from './logo.svg';

import './App.css';

function App() {

  return (

    <div className="App">

      <header className="App-header">

        <img src={logo} className="App-logo" alt="logo" />

        <p>

          Luciano Greco

        </p>

        <a

          className="App-link"

          href="https://reactjs.org"

          target="\_blank"

          rel="noopener noreferrer"

        >

          Learn React

        </a>

      </header>

    </div>

  );

}

export default App;



**LocalHost Server:** [**http://localhost:3000/**](http://localhost:3000/)



**> .gitignore:** Filtra los archivos que se van a subir git hub y los que no

# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies

/node\_modules **// No se subira a git por que los desarrolladores ya cuentan con estas librerias**

/.pnp

.pnp.js

# testing

/coverage

# production

/build

# misc

.DS\_Store

.env.local

.env.development.local

.env.test.local

.env.production.local

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*