

1-Fundamentos de las herramientas de desarrollo

Veremos herramientas utiles para el desarrollo con **REACT** como **NPM**, **BABEL**, **WEBPACK** aunque tambien se utilizaran

Entorno de desarrollo de REACT JS

Se puede implementar **REACT** utilizando la libreria sin compilar pero lo mas recomendado es utilizar **JSX** el cual es un lenguaje compilado. Para la implementacion de **REACT** se utilizara el modulo *create-react-app*.

Explicacion de NPM, Webpack y babel

NPM: Es un gestor de modulos de JS.

BABEL: Es un traductor de codigo JS con estandares nuevos, lo que permite que el proyecto sea utilizable en cualquier navegador, aunque este no soporte los ultimos estandares del **ecmascript**.

WEBPACK: Permite ulizar modulos para reutilizar codigos, es muy util utilizarlo junto con **BABEL**.

Chrome DevTools

Para abrir las **DevTools** se utiliza *ctrl+shift+i*.

Las **DevTools** posee una *consola* la cual permite una interaccion entre el programa y el navegador.

Creando Stack con REACT

Para crear una app de **REACT** utilizamos el comando **CREATE-REACT-APP nombre app**, este comando puede ser instalado con **NPM**.

Comandos utiles

- **NPM START:** Ejecuta la aplicacion en un servidor local.
- **NPM RUN:**

Estructura del proyecto

Dentro de la carpeta **public** se encuentra el **INDEX.html** en el cual se puede encontrar un *div* de clase *root* que en donde se generara nuestra aplicacion. El archivo **manifest.json** se encuentra las cosas necesarias para realizar una **PWA**.

Dentro de la carpeta **src** se encuentran todos los archivos de la aplicacion , en el **Index.js** se encuentra importa el modulo *serviceWorker* junto con el archivo **serviceWorker.js** se utiliza para generar una **PWA**.

2-Introduccion REACT CORE

Esta seccion permite comprender el entorno de **REACT**, y todo su entorno como **JSX** y las **expresiones** las cuales permiten generar codigo *html* de forma de reutilizar codigo. Junto con la aplicacion de codigo de **CSS**

para mejor las paginas web.

Los temas vistos en esta seccion son:

1. Introduccion a JSX
2. Componentes

Introduccion a JSX

JSX es una combinacion entre **JS** y **HTML** la cual permite indexar directamente **HTML** en codigo **JS**.

Creacion de elementos e insercion en el DOM

Utilizar **REACT** debo importar el modulo **React** y **ReactDOM**.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

const app = <h1>Hola React</h1>;

ReactDOM.render(app, document.getElementById('root') );
// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

Expresiones

Para insertar una expresion en **JSX**, utilizando llaves, dentro de ellos puedo colocar cualquier expresion valida de **JS**.

```
function crearFrase( name ){
  return 'Bienvenido ' + name
}

let name = 'Lautaro';

const app = <h1> { crearFrase( name ) } </h1>
```

Componentes

Los componentes permiten la reutilizacion de codigo . Existen 2 grandes grupos de componentes los **FUNCIONALES** y **BASADOS EN CLASES**. Los mas recomendables para utilizar son los componentes **FUNCIONALES** ya que necesitan menor cantidad de lineas de codigo gracias a los **HOOKS**.



Funcionales

Son funciones que retornan una unica etiqueta de **HTML**. Para llamar un componente un funcional se debe utilizar la nomenclatura clasica de **HTML**.

```
react-fundamentos > src > JS index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import './index.css';
4  import App from './App';
5  import * as serviceWorker from './serviceWorker';
6
7  function TarjetaFruta(){
8      return(
9          <div>
10             <h1> Lista de Frutas </h1>
11          </div>
12      );
13  }
14
15
16  ReactDOM.render(<TarjetaFruta/>, document.getElementById('root') );
17  // If you want your app to work offline and load faster, you can change
18  // unregister() to register() below. Note this comes with some pitfalls.
19  // Learn more about service workers: https://bit.ly/CRA-PWA
20  serviceWorker.unregister();
```

Props

Las **props** son variables de entrada de los componentes, los componentes que se le pasan a un **props** se deben utilizar llaves aunque para los datos de tipo **String** son opcionales.

```
react-fundamentos > src >  index.js >  TarjetaFruta
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import './index.css';
4  import App from './App';
5  import * as serviceWorker from './serviceWorker';
6
7  function TarjetaFruta(props){
8    return(
9      <div>
10         <h1> Lista de Frutas </h1>
11         <ul>
12           <li> { props.name } </li>
13         </ul>
14       </div>
15     );
16   }
17
18
19   ReactDOM.render(<TarjetaFruta name='sandia' />, document.getElementById('root') );
20   // If you want your app to work offline and load faster, you can change
21   // unregister() to register() below. Note this comes with some pitfalls.
22   // Learn more about service workers: https://bit.ly/CRA-PWA
23   serviceWorker.unregister();
```