**Front End 3**

**¿Qué es React? ⇓**

React es una librería con la cual crearemos proyectos desde el Front-End

**¿Cuáles son las ventajas de utilizar React en un proyecto? ⇓**

Su implementación permite que la carga de la aplicación sea más rápida y performance, actualizaciones de estado desde el navegador, menos carga desde el Back end, sin recargas de página, las secciones son componentes reutilizables

**¿Qué son las props? ⇓**

Son las propiedades de React, se usan para mejorar la funcionalidad de los componentes

**¿Cuál es la diferencia entre una librería y un framework? ⇓**

En la library los desarrolladores están a cargo del flujo del programa y es el mismo que decide dónde y cuándo llamar a la función de la library, en cambio el framework está a cargo del flujo del programa, nos indica lo que debemos hacer y llama al código que lo necesite.

**¿Qué es el Estado = STATE ? ⇓**

El State React es un objeto que representa el estado actual del componente, si el State cambia , el componente también cambiará.

**¿Cuándo fue creada React, y por quién? ⇓**

 React fue creada por Facebook en 2011.

**¿Qué son los componentes? ⇓**

Son independientes, centradas  y reutilizable pedazos de códigos.

**¿Qué entiendes por “Componentes Reutilizables”? ⇓**

Entendemos que son componentes que se pueden utilizar más de una vez sin la necesidad de escribir el código nuevamente.

**¿Por qué React no ha trascendido de librería a Framework? ⇓**

Por que usa otras librerías y herramientas que se puede considerar un framework.

**¿Qué se entiende como renderizar? ⇓**

Renderizar es la recopilación de los componentes, que constituye en estilizar el Dom.

**¿Qué se conoce o entiende como stack de desarrollo? ⇓**

Un conjunto de servicios de software que se utilizan para desarrollo de aplicaciones.

**¿Qué diferencia hay entre la extensión .js y la extensión .jsx? ⇓**

La extensión js no reconoce algunos comandos de Emmet la cual jsx si.

React y Babel

<script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>

    <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>

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

<script type="text/babel">

    /\* Tu código React va aquí

        No olvides que el elemento padre debe llamarse igual que el elemento que espera el render \*/

    ReactDOM.render(div\_el, document.getElementById("root"));

    </script>

**¿Qué es JSX?**

JSX son las siglas de JavaScript XML, y es una extensión de la sintaxis de JavaScript, es decir, que nos permite escribir Javascript de otra forma diferente a la manera estándar.

JSX no es un requisito para usar React, pero es muy conveniente usar esta sintaxis porque se puede escribir HTML directamente dentro de código JavaScript.

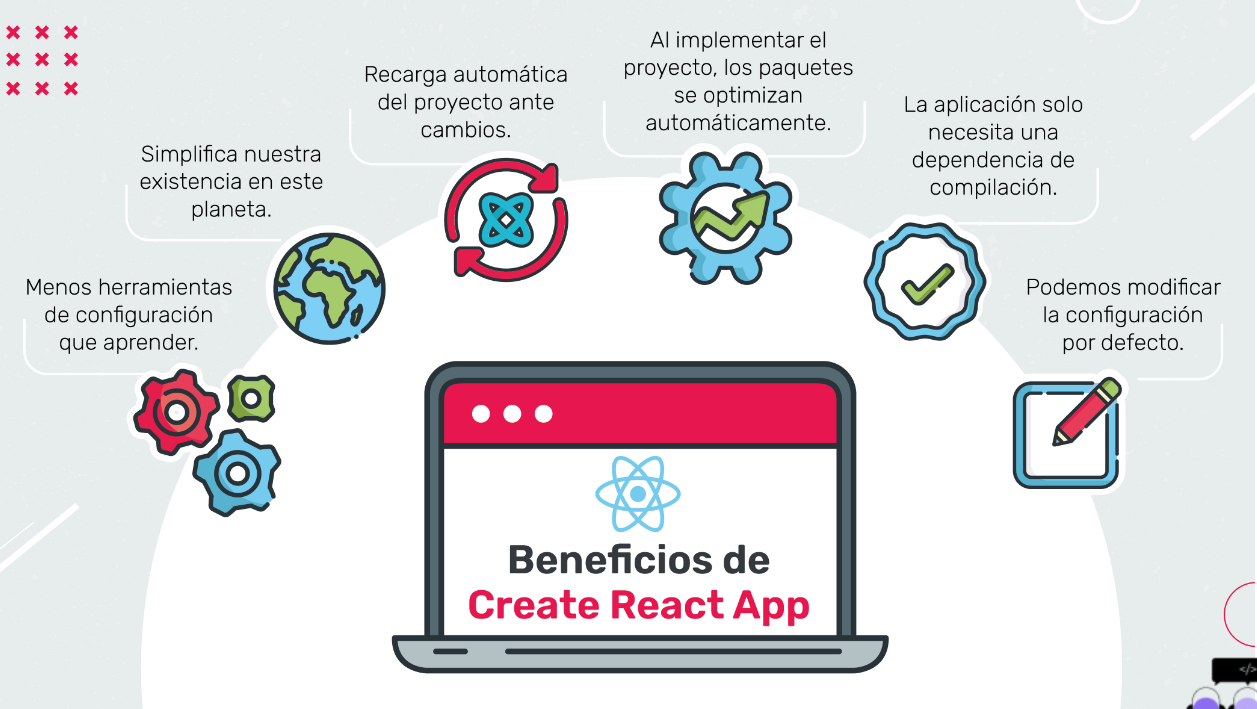
En el corazón de React está la idea de que en realidad la lógica de renderizado (o representación visual) está intrínsecamente acoplada con la lógica encargada de cómo manejar los eventos, cómo cambiar el estado de la interfaz con el tiempo y cómo preparar los datos para su visualización.

Por eso, React separa los aspectos (concerns) en unidades débilmente acopladas llamadas "componentes" que contienen tanto el marcado (HTML) como la lógica (Javascript), en lugar de separarlos artificialmente en archivos .js y .html.

SEMANA 2:

¿Qué es Create React App?

Create React App es un creador oficial de aplicaciones de React desarrollado por Facebook. CRA configura, mediante un solo comando, un ambiente de desarrollo de forma que podamos usar las últimas características de Javascript, permitiendo una gran y simple experiencia de desarrollo, y optimizando nuestra aplicación para producción. Asimismo, genera proyectos con compilación y linting preestablecidos. También viene con un servidor de desarrollo y soporte de aplicación web progresiva (PWA) de primera clase.



PASOS PARA CREAR REACT APP



LOS COMPONENTES

Son piezas funcionales y fundamentales de la aplicación, ya que nos van a permitir separar las distintas partes que conforman la estructura de un sitio web en pequeñas piezas independientes y reutilizables. Estas están pensadas para trabajar de forma aislada, pero haciendo parte de un “todo”.

Técnicamente, los componentes son funciones nativas de JavaScript que se ejecutan cada vez que sea necesario. Como cualquier función, pueden estar atadas, o no, a una serie de argumentos que nos permiten generar que este bloque de código pueda ser reutilizable, estableciendo que genere partes de la interfaz de usuario con información realmente dinámica.

PROPS

Las props son los datos internos de un componente. Representan información que es enviada al momento en el que un componente es utilizado. Estas permitirán que la información interna del componente sea variable para que podamos tener estructuras HTML realmente dinámicas y 100% reutilizables.

FRAGMENT

Cuando se trabaja con componentes en React, es necesario retornarlos dentro de una etiqueta que los envuelva. Esto produce que en un HTML se creen etiquetas vacías innecesarias y lo resuelve Fragment. Una posible solución es agrupar todo en una <div>, pero generaría etiquetas extra una y otra vez. Para ello React nos da la posibilidad de usar Fragment y así utilizar un envoltorio que finalmente no generará un nodo extra en nuestro DOM.

CHILDREN

Seguramente van a existir casos en los que tengamos un componente y queramos tan dinámico su uso que no va a ser suficiente con el simple uso de las props. Es aquí en donde cobran un particular protagonismo los children. A través de ellos vamos a tener la capacidad de enviar (como si fuera una prop) cualquier tipo de estructura HTML.

¿CUANDO LO UTILIZAMOS?

En caso de no saber qué contenido puede llegar a haber dentro de un componente padre, es común que los componentes actúen como cajas o contenedores genéricos de otros componentes. En estos casos es recomendable usar la prop special children para pasar elementos hijos directamente en el resultado. Esto permite que otros componentes pasen hijos arbitrarios anidando el JSX.

COMO

Cualquier elemento que pasemos dentro de la etiqueta JSX <Padre> se pasará a Padre como prop.children. Como resultado obtendremos al Hijo envuelto en el contenedor Padre.