**Conceptos Rápidos de react.**

**Términos y teoría de react:**

**React no es un marco de trabajo como lo es angular, react es una librería de JavaScript, que es utilizada para crear interfaces de usuario en frontend, la cual utiliza la modelo vista controlador.**

Redux: revisa el estado del código de tu aplicación con react.

**Comandos para el desarrollo de una aplicación en react:**

**create-react-app <NombreApp>:** con este comando podemos crear una nueva aplicación en react.

**Nota:** Create React App es un inicializador de proyectos de angular creado por Facebook esto, nos brinda una plantilla para crear una aplicación en react.

**npm start:** Corre nuestro proyecto en react, lo cual nos permite escuchar los cambios que realizamos en él en tiempo real.

**npm run build:** genera archivos de nuestro proyecto optimizados para producción.

**Carpetas y archivos de nuestra app en react.**

**node\_modules:** con las dependencias npm del proyecto

**public:** esta es la raíz de nuestro servidor donde se podrá encontrar el index.html, el archivo principal y el favicon.ico que sería el icono de la aplicación.

**src:** aquí es donde vamos a trabajar principalmente para nuestro proyecto, donde vamos a colocar los archivos de nuestros componentes React.

**README.md:** que es el readme de Create React App, con cantidad de información sobre el proyecto y las apps que se crean a partir de él. Este se escribe con un lenguaje o estructura de lenguaje llamado markdown.

**package.json:** que contiene información del proyecto, así como enumera las dependencias de npm, tanto para desarrollo como para producción. Si conoces npm no necesitarás más explicaciones.

.**gitignore:** que es el típico archivo para decirle a git que ignore ciertas cosas del proyecto a la hora de controlar el versionado del código. En este archivo se van a escribir las carpetas que se van a ignorar al momento de hacer el push a nuestro repo.

**Prueba practica de manejo de react**

**Archivos del proyecto**

Tag **<noscript>:** Esta etiqueta que, está ubicada en nuestro índex.HTML principal, realiza una validación en caso de que usuario desactive en JS en su navegador. Este será el mensaje que visualizará nuestro cliente.



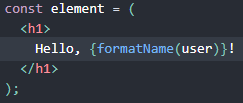
**Presentación de JSX:**

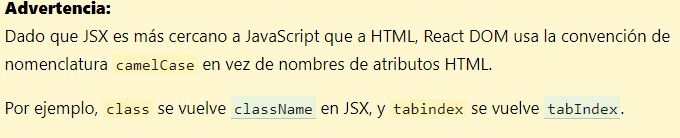


Esta curiosa sintaxis de etiquetas no es ni un string ni HTML.

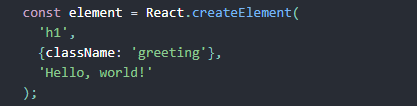
Se llama JSX, y es una extensión de la sintaxis de JavaScript. Recomendamos usarlo con React para describir cómo debería ser la interfaz de usuario. JSX puede recordarte a un lenguaje de plantillas, pero viene con todo el poder de JavaScript.

Cuando el elemento JSX contiene más de una línea de código es necesario en volverlo entre paréntesis para evitar errores, también el código se vuelve mas entendible.





Con JXS podemos crear elementos que poseen una estructura a un común objeto en JavaScript. Ejemplo:



De esta forma podemos crear elementos reutilizables que podrán ser utilizados en cualquier parte de nuestra aplicación, Tenga en cuenta que esta forma utiliza la importación a la biblioteca de react y su método createElement, para hacer la creación del mismo.

Este elemento lo vamos llamar cuando necesitemos hacer el renderizado.

En temas de seguridad el JSX se comporta de la siguiente forma, “Por defecto, React DOM [escapa](https://stackoverflow.com/questions/7381974/which-characters-need-to-be-escaped-on-html) cualquier valor insertado en JSX antes de renderizarlo. De este modo, se asegura de que nunca se pueda insertar nada que no esté explícitamente escrito en tú aplicación. Todo es convertido en un string antes de ser renderizado. Esto ayuda a prevenir vulnerabilidades [XSS (cross-site-scripting)](https://es.wikipedia.org/wiki/Cross-site_scripting).”

**Renderizado de elementos:**

Los elementos son los bloques más pequeños de las aplicaciones de React.

Un componente está constituido por más de un elemento.

Para renderizar un elemento en el ReactDom, se apunta a nuestro nodo raíz, si es una aplicación creada desde cero en react, de la siguiente forma. Ejemplo:



Es de esta forma que hacemos la inserción de un elemento, clase o función que retorne un elemento JSX a nuestro ReactDom.

**NOTA:** al igual que angular, en react para insertar alguna variable, se utiliza la interpolación de streams, pero en vez de dos llaves solo va ser una. Ejemplo:



Los elementos en react son inmutables, una vez se crea el elemento los hijos y atributos de estos no se podrán cambiar.   
a menos de que se vuelva hacer el llamado del elemento para realizar su creación. En caso de que suceda ese caso, react, evalúa que fue actualizado dentro del elemento y renderizado solo aquello con una nueva modificación

En el siguiente link habrá un ejemplo.

<https://codepen.io/pen?&editable=true&editors=0010>

docs:

<https://es.reactjs.org/docs/rendering-elements.html>

**state** and **props (el estado** y **los accesorios.): React posee estos términos que dan control a la forma en que se almacenan y manejan datos.**

**Importación react:** Esta importación es con la cual realizamos las interfaces y trae todos los elementos internos de la biblioteca de react.

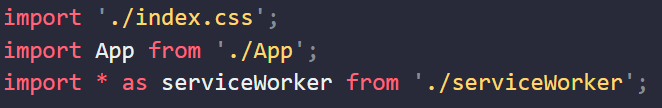


**Importación ReactDom:** con esta importación podemos crear interfaces específicamente para el navegador.



en react también nos permite agregar importación de documentos css, lo que nos permitirá ingresar a las clases escritas dentro de aquel documento importado.

La importación del serviceWorker nos permite ejecutar tu aplicación en modo off online



**Componente (Practica):**

son como elementos HTML personalizados y reutilizables, para crear interfaces de usuario de forma rápida y eficiente, los componentes es la manera de dividir nuestra vista en varias partes lo cual facilitara su manutención de la app, esta división es realizada de acuerdo a nuestras necesidades.

Podemos hacer el llamado de un componente dentro de otro de la siguiente forma:



Como vemos en la imagen el componente HelloWord, y lo llamamos en forma de etiqueta HTML dentro del componente que deseemos.

**Nota:** al momento de crear un nuevo componente debemos saber que la primera letra del nombre de nuestro componente debe ir en mayúscula, como se ve en el ejemplo de arriba.

También podemos crear un nuevo componente en forma de clase. Ejemplo: 

En este caso creamos una clase, y con el método render, se renderiza los elementos que llames y queramos en pantalla.

Los componentes en react tienen la misma filosofía que angular, ya que busca la reutilización del código lo máximo posible

En este primer link estará un fragmento de cogido donde no estará ejemplificado el concepto de modularidad.

<https://codepen.io/pen?editors=0010>

en este segundo link está el mismo ejemplo, pero con el concepto de modularidad aplicado en la resolución del problema.

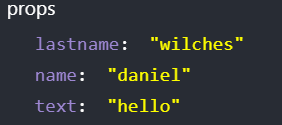
<https://codepen.io/pen?&editable=true&editors=0010>

podemos ver en este ejemplo que se crean más de un elemento con cada paso para solucionar su problema, de estos elementos de react creados se construye el componente en react, y este componente va ser una parte de la vista del usuario.

**Props (propiedades en español):**

Las propiedades o props en inglés, son la agrupación de todos los atributos que enviamos en nuestro elemento de react, estos atributos le podemos dar el nombre que nosotros deseemos, dentro de los props estos se organizan en forma de objeto. Ejemplo:

Este el objeto se llama props y su estructura es de esta forma:



Dentro del código su envió es de esta forma: 

Como vemos se guarda en clave valor, donde clave va ser el nombre de atributo, y el valor será la cadena que valla dentro de comillas.

Para más información sobre componentes y propiedades <https://es.reactjs.org/docs/components-and-props.html>

**State (Estado en español):**

El estado, como hace referencia su nombre, es el cual controla el estado de la aplicación, si el estado es actualizado mediante el método **setState()** el componente se vuelve a renderizar actualizando solo las modificaciones realizadas en el dom.

Ni los componentes padres o hijos pueden saber si un determinado componente tiene o no tiene estado y no les debería importar si se define como una función o una clase.

Por eso es que el estado a menudo se le denomina local o encapsulado. No es accesible desde otro componente excepto de aquel que lo posee y lo asigna.

Un componente puede elegir pasar su estado como props a sus componentes hijos.

En las aplicaciones de React, si un componente tiene o no estado se considera un detalle de implementación del componente que puede cambiar con el tiempo. Puedes usar componentes sin estado dentro de componentes con estado y viceversa.

<https://es.reactjs.org/docs/state-and-lifecycle.html>

Notas: las llaves “{ … }” interpreta código JavaScript