REACT

Compartiré mis apuntes de la clase con lo que habia aprendido anteriormente (Cualquier feedback, es bienvenido)

**Componentes vs Elementos**

En React hay varias formas de crear un elemento:

* **Con clases:** Actualmente no se usa

**class** **Componenete** **extends** **React**.**Components** {

**render**() {

**return** (

//Codigo

)

}

}

* **Con React.createElement:** Se sigue usando. Es opcional

Sintaxis: React.createElement(elemento, atributos, texto/contenido)  
En el *elemento* se colocaran el nombre de las etiquetas HTML. Ejemplo, h1, h2, p, form,…

En los *atributos* se colocaran los atributos de las etiquetas, es decir, id, class, placeholder,…

En texto o contenido se coloca contenido que va dentro de la etiqueta. Es decir, <h1>**este contenido**</h1>

Veamos con un ejemplo:

**const** ejemplo1 = React.createElement('h1', {'id': 'title'}, 'Oli React')

**const** ejemplo2 = React.createElement(

'p',

{

'id': 'paragraph-elemental',

'class': 'paragraph'

},

'Oli React'

)

En el ejemplo 1, lo que hice fue colocar el ejemplo de Juan que mostró en el video.

En el ejemplo 2, lo que hice fue colocar un ejemplo parecido, solo que le di una lectura más legible para los casos que tengamos que colocar más atributos y no se nos complique la lectura al tener esto:

**const** ejemplo2 = React.createElement('p', {'id': 'paragraph-elemental','class': 'paragraph' }, 'Oli React');

Ambas versiones del ejemplo 2 son validas, solo es cuestión de que la persona lea mejor el código.

* **Con Funciones:** Se usa actualmente y es más cómodo que usar React.createElement()

**function** **Componente** = () => {

**return**(

//Codigo

)

}

ReactDOM.render(qué\_elemento, dónde) se encarga de renderizar el elemento y colocarlo en el dom. Se pasa por por parametro el elemento a colocar en el DOM y en dónde se quiere colocar

Esto es un componente:

//Componente

**const** Componente = () => {

**return** (

//Codigo

)

}

<Componente />

Este es un elemento:

//Elemento

<h1>Dorime</h1>

Nota: Los componentes son una version traducida de los elementos en HTML a Javascript (JSX).

**Props vs Atributos**

Cuando estamos trabajando con React, para definir el atributo class, no usamos class sino className. React te puede aceptar class como atributo, pero luego te saldrá advertencias y, de paso, es una mala práctica

Las propiedades las podemos recibir de los parametros de los componentes. Ejemplo:

<App nombre="Dorime">

**const** App = (props) = {

**return** (

<p>{props.nombre}</p>

)

}

Tambien podemos recibir children que vienen entre el contenido del Componente.

<App>

<p>Dorime, Ameno</p>

</App>

**const** App = (props) = {

**return** (

{props.children}

)

}

React.state (react hooks)

const [estado, estadoFuncion] = use.ReactState(‘value’)

Este posee dos valores, el primero es el estado el cual es asignado por el valor de la función ‘ ReactState(“value”) ‘. Después esta la función ‘estadoFuncion’ el cual nos permite cambiar el valor del estado. Se podría decir que vuelve a utilizar la funcoin ReactState(“value”) nuevamente, asignándole un nuevo valor al estado cada vez que sea llamado.

LOCAL STORAGE

Hubo un momento en el que me confundí de donde salía tanta variable XD, pero analizándolo volví al camino. Es fácil perderse con tanto código

**12.-Persistencia de datos con Local Storage**

Nos permite tener persistencia de datos directamente en el front-end, sin necesidad del back-end.

Todos los navegadores web (o los mejorcitos) tienen esta herramienta, que nos permite guardar la info que nosotros queramos y mantenerla activa durante mucho tiempo.

* No importa si los usuarios cierran o recargan la ventana, cierran el navegador o apagan la computadora o vuelven dentro de cierto tiempo, local storage va a seguir guardando toda la info en el navegador.
* No solo se puede guardar, también se puede recibir.

Para hacer uso de esta herramienta vamos a crear un item, un nuevo elemento a guardar dentro de local storage.

NOTA: Solo se puede guardar texto, no objetos, números, ni otro tipo de datos.

localStorage.setItem('ejemploTodos', ejemplo); //(lo que queremos guardar, lo que vamos a guardar)

//Para guardar los todos usaremos

**const** ejemplo = JSON.stringify() //Nos permite transformar en texto un objeto o array de JS

JSON.parse(ejemplo) //Transforma en JS la variable con el array hecho string

localStorage.getItem('ejemploTodos'); //Nos lo va a mostrar

JSON.parse(localStorage.getItem('ejemploTodos'));