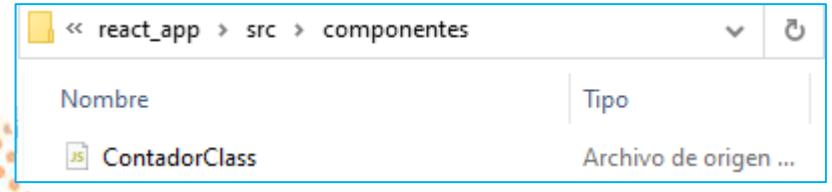
Componentes Basados en Clases

Hasta ahora en las clases hemos creado componentes basados en funciones, sin embargo, no son los únicos componentes que existen, también están los componentes basados en clases y de hecho, estos componentes son los más usados en un principio con React.

Estos componentes son más difíciles de usar, más complejos de entender y ameritan más líneas de código para ser usados, pero aún así, podemos encontrarlos en algunas aplicaciones menos actualizadas y por eso vamos a conocerlos también para que no tengamos sorpresas en caso de toparnos con ellos más adelante en algún desarrollo que tengamos que encarar con React.

Componentes Basados en Clases

Para ver cómo funcionan estos componentes, haremos un caso práctico con un contador, por eso iremos a nuestra carpeta de componentes y crearemos allí un archivo llamado "ContadorClass.js"



Ahora para crear el componente vamos a crear la clase "Contador" extendida de la clase base "Component" de React, dentro de ella va el método "render" y dentro de este en un "return" vamos a codificar todo lo que queremos mostrar en pantalla:

```
1  import React, {Component} from 'react';
2  class Contador extends Component {
3      render(){
4          return(
5
6          );
7      }
8  }
9  export default Contador;
```

Componentes Basados en Clases

En este ejemplo puntual vamos a colocar un encabezado <h2> que diga "Esta es la clase número" dentro de nuestro componente:

```
4           return(
5               <h2>Esta es la clase número</h2>
6           );
```

Luego vamos a nuestro "index.js" para importar el componente y mostrarlo en pantalla:

```
7   import ContadorClass from './componentes/ContadorClass';
```

```
13          {sesion === true ?
14          <>
15            <Aula/>
16            <ContadorClass/>
17            <Boton rojo onClick={() => cambiarEstadoSesion(false)}>Cerrar Sesión</Boton>
18          </>
```

Seguidamente en el valor del estado de inicio de sesión colocamos por defecto "true" para que se muestre en pantalla directamente el contador sin tener que iniciar sesión:

```
9   const App = () =>{
10    const [sesion, cambiarEstadoSesion] = useState(true);
```

Componentes Basados en Clases

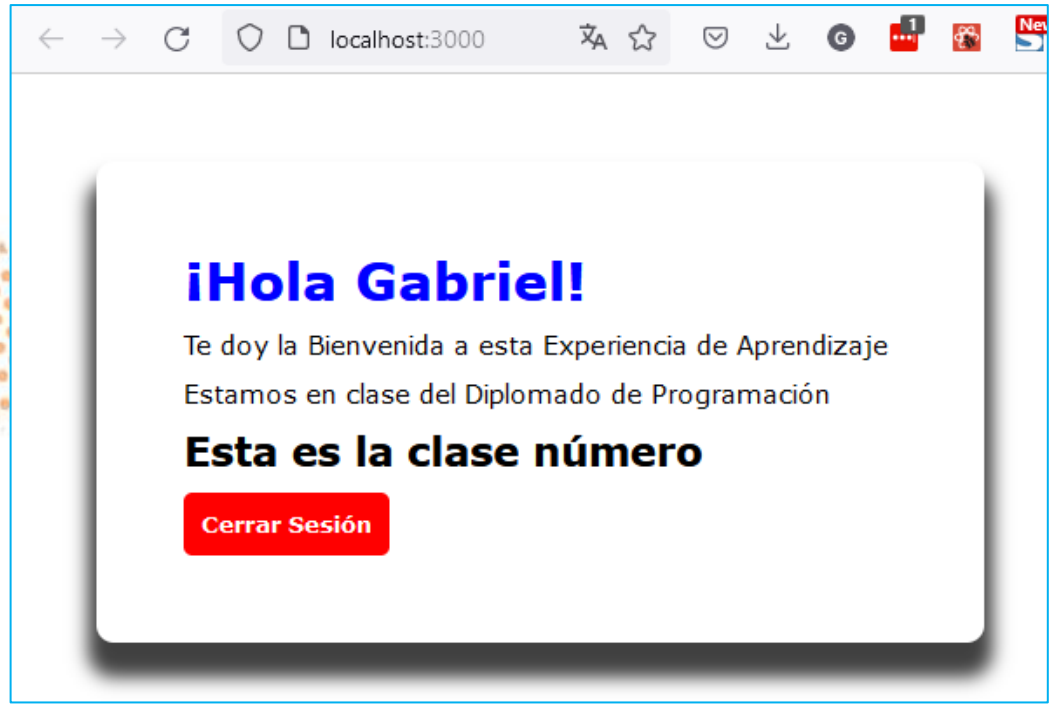
Seguidamente, vamos a ingresar en el componente funcional "Aula.js" y vamos a comentar el título naranja y el listado que muestra a los profesores:

```
9       <>
10        <Titulo nombre="Gabriel" color="blue"/>
11        {/*<Titulo nombre="Alejandro" color="orange"/>*/}
12        <Parrafo>Te doy la Bienvenida a esta Experiencia de Aprendizaje</Parrafo>
13        {clase && <Parrafo>Estamos en clase del {clase}</Parrafo>}
14        {/*<h4>Estos son los profesores:</h4>
15        <ul>
16        {instructores.map((instructor, index) => {
17          return <li key={index}>{instructor}</li>
18        })}
19      </ul>*/}
20      </>
```

De esta forma tendremos nuestra pantalla con menos datos y será más fácil para visualizar los nuevos cambios.

Componentes Basados en Clases

El resultado obtenido hasta el momento en nuestra pantalla es el siguiente:



Componentes Basados en Clases

A continuación crearemos dos (2) métodos que nos ayudarán a incrementar y disminuir el número de clases, pero que por ahora, solo mostrarán una alerta al ser invocados:

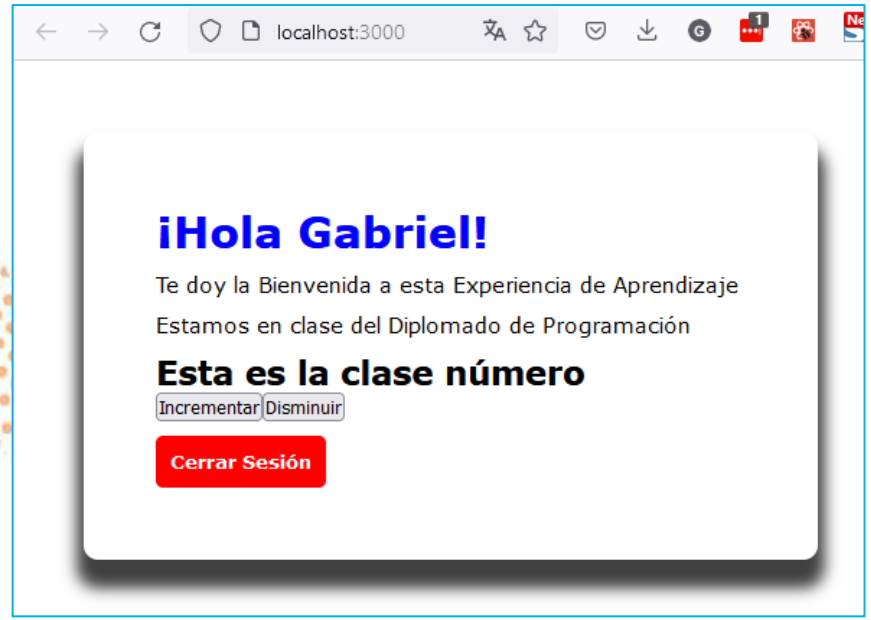
```
2   class Contador extends Component {
3       incrementar(){
4           alert('Incrementar');
5       }
6       disminuir(){
7           alert('Disminuir');
8       }
9       render(){
```

Seguidamente agregaremos dos (2) botones en nuestro componente Contador, cada uno con el atributo onClick para que cuando se pulse cada botón, podamos llamar al método correspondiente y así ejecutar código:

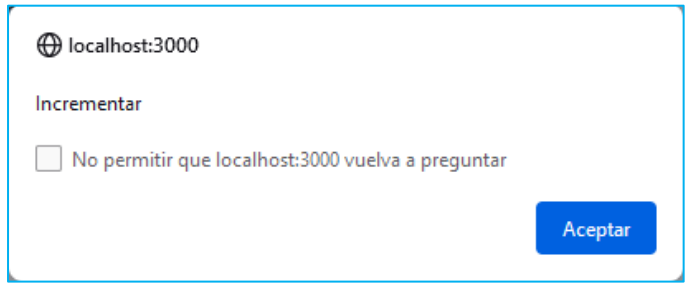
```
10          return(
11              <div>
12                  <h2>Esta es la clase número</h2>
13                  <button onClick={() => this.incrementar()}>Incrementar</button>
14                  <button onClick={() => this.disminuir()}>Disminuir</button>
15              </div>
16          );
```

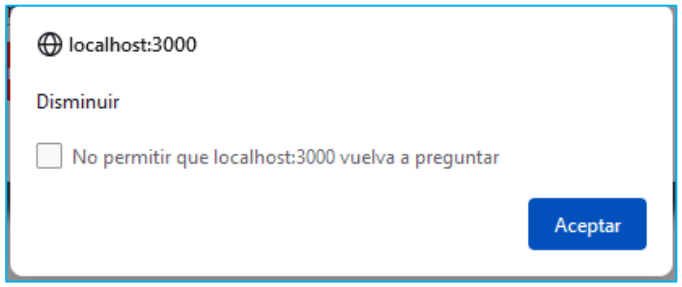
Componentes Basados en Clases

De esta forma hemos logrado que en nuestra pantalla aparezcan los dos botones:



Pero eso no es todo, al hacer clic en cada uno de ellos, aparecerá una alerta:





Componentes Basados en Clases

Ahora para que nuestro componente funcione cómo debería, agregaremos el uso del estado al mismo. Para esto, primeramente debemos ejecutar el método constructor para poder recibir, acceder y trabajar con propiedades en la clase y para poder usar el estado ya que se debe inicializar dentro del constructor de la siguiente forma:

```
2   class Contador extends Component {
3       constructor(props){
4           super(props);
5           this.state = { contador: 0 }
6       }
```

De esta forma ya podemos mostrar en pantalla el valor de nuestro contador que a su vez es nuestro estado. Lo haremos dentro del encabezado <h2> que creamos recientemente, colocando unas llaves {} y dentro de ellas "this.state.contador":

```
15              <div>
16                  <h2>Esta es la clase número {this.state.contador}</h2>
17                  <button onClick={() => this.incrementar()}>Incrementar</button>
18                  <button onClick={() => this.disminuir()}>Disminuir</button>
19              </div>
```

Componentes Basados en Clases

Para continuar lo que debemos programar es que al momento de hacer clic en los botones de disminuir e incrementar, se sume o reste un número al estado anterior, para eso, vamos añadir dos (2) propiedades al llamado de nuestro componente Contador en el index.js:

```
15          <Aula/>
16          <ContadorClass incrementar={1} disminuir={1}/>
```

Estas propiedades las igualamos al número que queremos sumarle o restarle a nuestro contador, seguidamente, dentro de nuestro componente del contador, teniendo en cuenta que ya estamos recibiéndolas en el constructor, las vamos a colocar dentro del llamado de nuestras funciones de incrementar y disminuir en los botones:

```
3   constructor(props){
4       super(props);
5       this.state = { contador: 0 }
6   }
```

```
24              <h2>Esta es la clase número {this.state.contador}</h2>
25              <button onClick={() => this.incrementar(this.props.incrementar)}>Incrementar</button>
26              <button onClick={() => this.disminuir(this.props.disminuir)}>Disminuir</button>
```

Componentes Basados en Clases

Asimismo, dentro de cada una de las funciones invocadas por los botones anteriormente modificados, debemos recibir las propiedades enviadas por ellos (podemos recibirlas con cualquier nombre) y sumarlas/restarlas según sea el caso al valor de nuestro EstadoAnterior (contador) usando el método ".setState" para que nuestro nuevo estado sea el resultado de la suma o resta realizada:

```
7   incrementar(CantidadAIncrementar){
8       this.setState((EstadoAnterior) => {
9           return{
10              contador: EstadoAnterior.contador + CantidadAIncrementar
11          }
12      });
13  }
14  disminuir(CantidadADisminuir){
15      this.setState((EstadoAnterior) => {
16          return{
17              contador: EstadoAnterior.contador - CantidadADisminuir
18          }
19      });
20  }
```

El parámetro `EstadoAnterior` en el método `this.setState(EstadoAnterior)` permite acceder al estado actual del componente y realizar cambios en función de ese estado.

Al pasar el estado anterior como argumento, React garantiza que se realicen las actualizaciones correctas y evita posibles problemas de concurrencia.

Componentes Basados en Clases

Hecho todo lo anterior, lograremos ver en pantalla nuestro contador de clases funcionando: Cada vez que demos clic en el botón “Incrementar" sumaremos un (1) digito a la cuenta y cuándo demos clic en “Disminuir" restaremos un (1) digito: