



ESCUELA DE
INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



Día, Fecha:

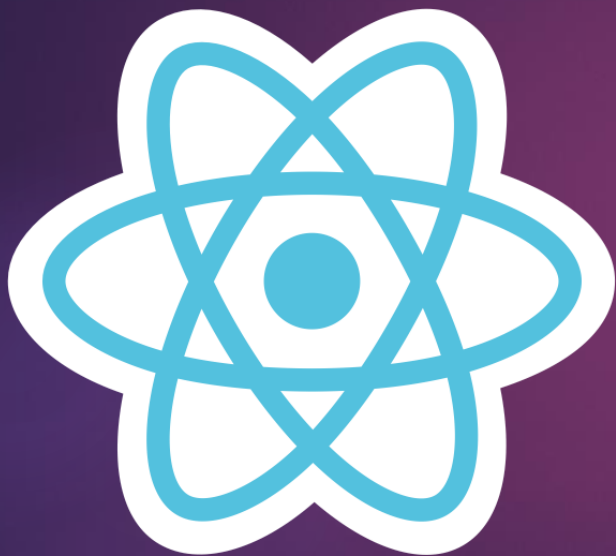
Jueves, 19/10/2023

Hora de inicio:

07:10

Introducción a la programación y computación 1 [F]

José Eduardo Morales García



React

React es una biblioteca de JavaScript ampliamente utilizada para construir interfaces de usuario interactivas y eficientes.

¿Qué es React?

React, también conocido como React.js o ReactJS, es una biblioteca de JavaScript desarrollada por Facebook. Se utiliza para construir interfaces de usuario altamente interactivas y eficientes. A diferencia de un marco completo, React se enfoca en la capa de la vista de una aplicación web, permitiéndote crear componentes reutilizables que representan partes de la interfaz de usuario.

¿Cómo funciona React?

React funciona construyendo una jerarquía de componentes. Cuando los datos cambian, React compara el Virtual DOM (una representación virtual de la interfaz) con el DOM real y actualiza solo las partes que han cambiado. Esto minimiza la carga en el navegador y mejora el rendimiento de la aplicación.

Componentes en React

Los componentes son la piedra angular de React. Pueden ser funcionales o de clase. Los componentes funcionales son funciones de JavaScript que devuelven elementos JSX, mientras que los componentes de clase son clases de JavaScript que extienden la clase Component de React. Los componentes permiten la creación de elementos de interfaz reutilizables.

Virtual DOM

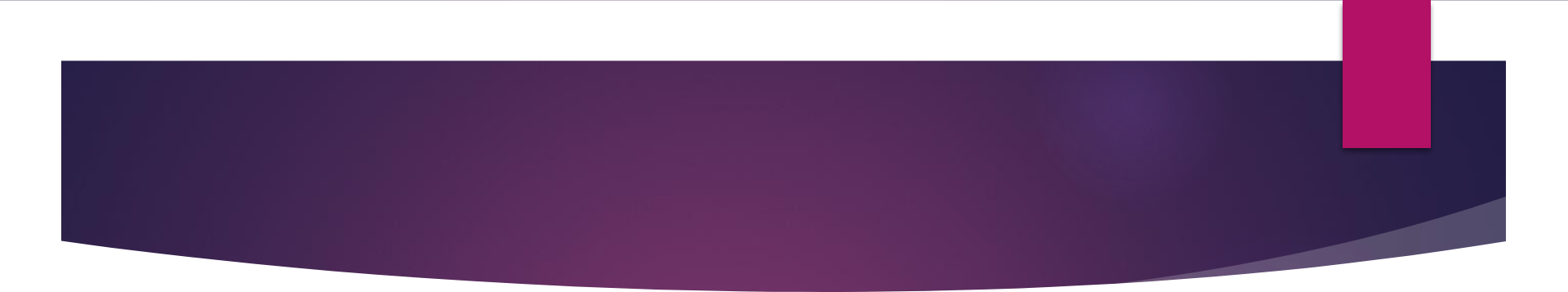
El Virtual DOM es una representación virtual de la estructura de la interfaz de usuario. React lo utiliza para realizar comparaciones eficientes entre el estado anterior y el estado actual de la aplicación. Esta técnica minimiza la manipulación del DOM real y, por lo tanto, mejora el rendimiento.

JSX (JavaScript XML)

JSX es una extensión de JavaScript que permite escribir elementos de interfaz de manera declarativa y similar a HTML. JSX facilita la creación de componentes y su renderización en React. Aunque se parece a HTML, se compila a JavaScript puro para su ejecución.

Instalación de Node.js

Para trabajar con React en Windows, primero debemos instalar Node.js, que incluye npm (Node Package Manager). Puedes descargar el instalador desde el sitio web oficial de Node.js en nodejs.org.



Una vez que hayas descargado el instalador, ejecuta el archivo y sigue las instrucciones de instalación. Asegúrate de que Node.js y npm estén correctamente instalados ejecutando `node -v` y `npm -v` en la línea de comandos.

Creación de una aplicación React

Para crear una nueva aplicación React, puedes utilizar el comando **create-react-app nombre-de-tu-app** en la línea de comandos. Esto generará una estructura de proyecto inicial con todas las configuraciones necesarias.

```
Success! Created fronted at C:\Users\edush\OneDrive\Escritorio\Proyecto\fronted  
Inside that directory, you can run several commands:
```

```
npm start
```

```
Starts the development server.
```

```
npm run build
```

```
Bundles the app into static files for production.
```

```
npm test
```

```
Starts the test runner.
```

```
npm run eject
```

```
Removes this tool and copies build dependencies, configuration files  
and scripts into the app directory. If you do this, you can't go back!
```

```
We suggest that you begin by typing:
```

```
cd fronted
```

```
npm start
```

```
Happy hacking!
```

Estructura de una Aplicación React

Una aplicación React típica consta de componentes, archivos de configuración, y una carpeta de dependencias. Los componentes principales generalmente se encuentran en la carpeta `src`. Esta estructura permite organizar tu código de manera efectiva.

Iniciar la Aplicación React

Navega a la carpeta de tu aplicación usando el comando `cd nombre-de-tu-app` y luego ejecuta `npm start` en la línea de comandos. Esto iniciará un servidor de desarrollo y abrirá la aplicación en tu navegador predeterminado.

Desarrollo en React

Puedes editar los archivos en la carpeta src de tu proyecto y ver los cambios en tiempo real en tu aplicación mientras desarrollas. React recarga automáticamente la página cuando detecta modificaciones en los archivos.

Componentes y Props

En React, los datos se pasan entre componentes a través de props (propiedades). Esto permite la reutilización de componentes y la comunicación entre ellos. Las props son valores inmutables que se transmiten de los componentes padres a los hijos.

Estado en React

Los componentes en React pueden tener su propio estado interno que se puede modificar a lo largo del tiempo. Cuando el estado de un componente cambia, React se encarga de actualizar automáticamente la interfaz para reflejar esos cambios.

¿Qué son useState y useEffect?

useState y useEffect son dos de los hooks más fundamentales en React. Estas funciones permiten a los desarrolladores administrar el estado y efectos secundarios en componentes funcionales.

useState en React

useState es un hook que permite a los componentes funcionales de React mantener y gestionar su estado local. Puedes utilizarlo para declarar variables de estado y actualizarlas de manera reactiva. Esto ayuda a que los componentes reflejen cambios en la interfaz de usuario en respuesta a eventos o actualizaciones de datos.

useEffect en React

useEffect es otro hook crucial que se utiliza para gestionar efectos secundarios en componentes funcionales de React. Los efectos secundarios incluyen la ejecución de código después de la representación inicial, la manipulación del DOM y las solicitudes a APIs externas. useEffect se ejecuta después de cada renderizado y te permite controlar cuándo y cómo se realizan estas tareas.

```
import React, { useState } from 'react';

function Counter() {
  // Declarar una variable de estado llamada "count" con un valor inicial de 0
  const [count, setCount] = useState(0);

  const increment = () => {
    // Actualizar el estado "count" cuando se hace clic en el botón
    setCount(count + 1);
  };

  return (
    <div>
      <p>Contador: {count}</p>
      <button onClick={increment}>Incrementar</button>
    </div>
  );
}

export default Counter;
```

```
1 import React, { useState, useEffect } from 'react';
2 import axios from 'axios';
3
4 function FetchData() {
5   const [data, setData] = useState(null);
6
7   useEffect(() => {
8     // Realizar una solicitud GET a una API y actualizar el estado con los datos
9     axios.get('URL_DE_TU_API_AQUI')
10      .then((response) => {
11        setData(response.data);
12      })
13      .catch((error) => {
14        console.error('Error al cargar los datos:', error);
15      });
16   }, []); // El segundo argumento vacío [] indica que se ejecuta solo una vez al montar el componente.
17
18   return (
19     <div>
20       {data ? (
21         <div>
22           <p>Nombre: {data.nombre}</p>
23           <p>Apellido: {data.apellido}</p>
24           <p>Edad: {data.edad}</p>
25         </div>
26       ) : (
27         <p>Cargando datos...</p>
28       )}
29     </div>
30   );
31 }
32
33 export default FetchData;
34
```

```
1  import React, { useState, useEffect } from 'react';
2  import axios from 'axios';
3
4  function App() {
5    const [data, setData] = useState(null);
6    const [materia, setMateria] = useState('');
7    const [nota, setNota] = useState('');
8
9    useEffect(() => {
10      const apiUrl = 'URL_DE_TU_ENDPOINT_AQUI';
11
12      axios.get(apiUrl)
13        .then((response) => {
14          setData(response.data);
15        })
16        .catch((error) => {
17          console.error('Error al cargar los datos:', error);
18        });
19    }, []);
20
21    const handleEnviarSolicitud = () => {
22      const apiUrl = 'URL_DEL_ENDPOINT_PARA_ENVIAR_NOTA_AQUI';
23
24      // Crear un objeto con los datos que se enviarán en el cuerpo de la solicitud
25      const dataToSend = {
26        materia: materia,
27        nota: parseFloat(nota), // Asegurarse de que la nota sea un número
28      };
29
30      // Realizar la solicitud POST
31      axios.post(apiUrl, dataToSend)
32        .then((response) => {
33          setData(response.data);
34        })
35        .catch((error) => {
36          console.error('Error al enviar la solicitud:', error);
37        });
38    };
39  }
```

```
39
40   return (
41     <div className="App">
42       <h1>Ejemplo de Consumo de API con React</h1>
43       {data ? (
44         <div>
45           <p>Nombre: {data.nombre}</p>
46           <p>Apellido: {data.apellido}</p>
47           <p>Edad: {data.edad}</p>
48         </div>
49       ) : (
50         <p>Cargando datos...</p>
51       )}
52
53       <div>
54         <h2>Enviar Nota</h2>
55         <input
56           type="text"
57           placeholder="Materia"
58           value={materia}
59           onChange={(e) => setMateria(e.target.value)}
60         />
61         <input
62           type="number"
63           placeholder="Nota"
64           value={nota}
65           onChange={(e) => setNota(e.target.value)}
66         />
67         <button onClick={handleEnviarSolicitud}>Enviar Nota</button>
68       </div>
69     </div>
70   );
71 }
72
73 export default App;
```



Parte práctica