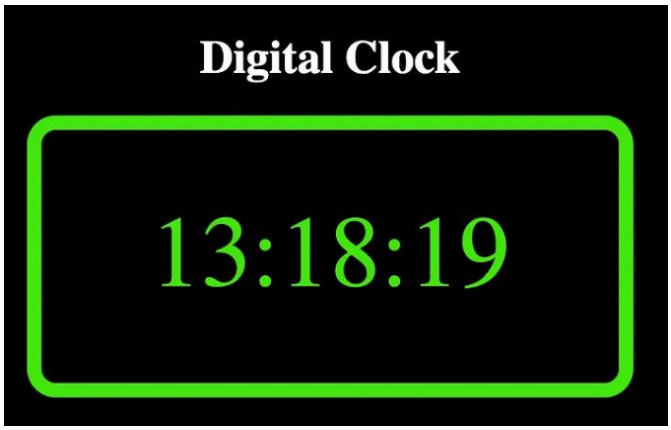
C5.1_

051ED52_Digital Clock



Introducción

En este tutorial, crearemos una aplicación de reloj digital simple usando React.js. Este proyecto nos enseñará cómo utilizar setInterval para actualizaciones en tiempo real e introducirá el gancho useEffect para administrar efectos secundarios en aplicaciones React.

Explicación del hook `useEffect` en React para principiantes:

Imagina que tienes un amigo mágico llamado React que te ayuda a crear juguetes (componentes) para tu ciudad de juguetes (aplicación web). Ahora, a veces quieres decirle a React: "¡Oye, cuando esto suceda, haz algo especial!"

`useEffect` es como un mensaje mágico que le das a React. Dices: "Reacciona, si algo cambia o si la ciudad de juguetes apenas comienza, haz algo especial". Es como decirle a React que vigile ciertas cosas y realice hechizos mágicos cuando esas cosas sucedan.

Por ejemplo, es posible que desees que React te avise cuando llegan juguetes nuevos o cuando es necesario arreglar un juguete. ¡Ahí es cuando usas `useEffect` para comunicarte con tu amigo mágico y hacer que tu ciudad de juguetes sea aún más emocionante!

Explicación técnica del gancho `useEffect` en React:

En React, los componentes pueden hacer cosas especiales cuando se crean, actualizan o eliminan. El gancho `useEffect` es como un compañero que te ayuda a gestionar estas acciones especiales, llamadas efectos secundarios.

1. **Component Did Mount/El componente se montó:**

cuando construyes un nuevo juguete (component), "useEffect" es como la primera vez que juegas con él. Se ejecuta inmediatamente después de que se crea el juguete.

```
useEffect(() => {
    // Special magic when the toy is created!
  }, []);
```

2. **Component Did Update/El componente se actualizó:**

si cambia algo en su juguete, como darle un nuevo color, `useEffect` le ayuda a hacer algo especial después del cambio.

```
useEffect(() => {
    // ¡Magia especial cuando el juguete cambia!
  }, [color]);
```

3. **Component Will Unmount/El componente se desmontará:**

cuando decides guardar un juguete y dejar de jugar con él, "useEffect" te ayuda a limpiar y hacer cualquier magia de último momento

```
useEffect(() => {
    return () => {
      // ¡Magia especial antes de guardar el juguete!
    };
  }, []);
```

Entonces, "useEffect" es como tu ayudante para manejar tareas especiales relacionadas con tus juguetes en la ciudad del juguete. Mantiene las cosas organizadas y garantiza que tu magia se lance en el momento adecuado.

Ahora volvamos a construir nuestro proyecto:

Requisitos previos
Asegúrese de tener Node.js y npm (Node Package Manager) instalados en su sistema.

Paso 1: configurar una nueva aplicación React
Abra su terminal y ejecute el siguiente comando para crear una nueva aplicación React:

```
npx create-react-app digital-clock-app
```

Este comando configurará un nuevo proyecto de React con los archivos y dependencias necesarios.

Paso 2: cree la app Clock

Navegue hasta la carpeta src en el directorio de su proyecto y abra el archivo App.js. Reemplace su contenido con el siguiente código:

```
import React, { useState, useEffect } from "react";
import "./App.css";

function App() {
  const [time, updateTime] = useState(new Date());
  useEffect(() => {
    // timer updation logic
    const timer = setInterval(() => {
      updateTime(new Date());
    }, 1000);
    return () => clearInterval(timer);
  }, []);
  return (
    <div className="App">
      <div className="elementcontainer">
        <h1>Digital Clock</h1>
        <div className="timeparent">
          <div className="timecontainer">
            {/* print the string prettily */}
            <span className="time">{time.toLocaleTimeString()}</span>
          </div>
        </div>
      </div>
    </div>
  );
}

export default App;
```

Explicación

Hook useEffect

El hook `useEffect` en React se usa para realizar efectos secundarios en componentes de funciones.
En nuestro componente de aplicación, usamos `useEffect` para configurar un intervalo que actualiza el estado del tiempo cada segundo (1000 milisegundos).
La función `useEffect` toma dos argumentos: una función de devolución de llamada y una serie de dependencias. En este caso, la matriz vacía [] significa que el efecto solo se ejecutará una vez cuando se monte el componente.

`***setInterval*** `

`setInterval` es una función de JavaScript que llama repetidamente a una función proporcionada en un intervalo específico.
En nuestro gancho useEffect, configuramos un intervalo para actualizar el estado de tiempo cada segundo.
Agregué una referencia en el pie de página para diseñar el reloj tal como se muestra en la vista previa de la aplicación.

Paso 3: Crear estilo CSS

Cree un archivo llamado `App.css` en la carpeta src y agregue algo de CSS básico para diseñar nuestra aplicación:

```
body {
  background-color: black;
  margin: 0;
  padding: 0;
}

.App {
  text-align: center;
  color: white;
  height: 100vh;
  width: 100vw;
}
.timeparent {
  display: flex;
  justify-content: center;
  align-items: center;
}
.time {
  color: white;
  font-size: 70px;
  position: relative;
  display: block;
  margin-top: 45px;
  color: rgb(117, 249, 77);
}
.timecontainer {
  text-align: center;
  height: 175px;
  width: 400px;
  border: 10px solid rgb(117, 249, 77);
  border-radius: 20px;
```

```
}

@media only screen and (max-width: 600px) {
  .timecontainer {
    margin-left: 10px;
    margin-right: 10px;
  }
}
```