Semana 5: Manejo de Estado y Ciclo de Vida

useState y useEffect

`useState`

- **Definición**: `useState` es un Hook de React que permite añadir estado reactivo a componentes funcionales.
- **Ejemplo**:

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

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Has clicado {count} veces</p>
      <button onClick={() => setCount(count + 1)}>
        Clic
      </button>
    </div>
  );
}
```

`useEffect`

- **Definición**: `useEffect` es un Hook que se utiliza para ejecutar efectos secundarios en componentes funcionales, como peticiones de datos, suscripciones o cambios manuales en el DOM.
- **Ejemplo**:

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

function Example() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    // Actualiza el título del documento usando la API del navegador
    document.title = `Has clicado ${count} veces`;
  });

  return (
    <div>
      <p>Has clicado {count} veces</p>
      <button onClick={() => setCount(count + 1)}>
        Clic
      </button>
    </div>
  );
}
```

Semana 5: Manejo de Estado y Ciclo de Vida

Ciclo de vida de los componentes

- **Componentes Funcionales**: Con Hooks, los componentes funcionales pueden usar `useEffect` para manejar los ciclos de vida.
- **Etapas del Ciclo de Vida**:
  - **Montaje**: Cuando el componente se renderiza por primera vez.
  - **Actualización**: Cuando el estado o las props del componente cambian.
  - **Desmontaje**: Cuando el componente se elimina del DOM.

Ejemplos Prácticos

Ejemplo de Montaje y Actualización

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

function Timer() {
  const [seconds, setSeconds] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      setSeconds(seconds => seconds + 1);
    }, 1000);
    return () => clearInterval(interval);
  }, []);

  return <div>Segundos transcurridos: {seconds}</div>;
}
```

Ejemplo de Desmontaje

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

function ExternalDataLoader() {
  const [data, setData] = useState(null);

  useEffect(() => {
    fetchData().then(data => setData(data));
    return () => {
      // Limpieza o desmontaje
      cleanUpData();
    };
  }, []);

  // ... renderizado de datos
}
```

Estas diapositivas ofrecen una vista general de cómo usar `useState` y `useEffect` para manejar el estado y los ciclos de vida en componentes funcionales, proporcionando ejemplos prácticos para una mejor comprensión.