wawiwa
React
2022 © Wawiwa Tech | Confidential

wawiwa

Construcción de un Hook Personalizado

Hasta este punto, hemos usado los Hooks construidos en React (como useState, useEffect y todas estas funciones especiales de React que empiezan con “use”).

En este capítulo aprenderemos sobre la poderosa funcionalidad de React que nos permite **extraer y reusar lógica de estado entre componentes.**

Los hooks personalizados proveen una manera de **compartir lógica entre múltiples componentes,** haciendo posible re-usar su lógica en múltiples lugares en tu aplicación.

Vamos a aprender:

- Qué son los hooks personalizados
- Cómo construir tu propio Hook
- Por qué podrías querer construir eso
- Ejemplos de Hooks personalizados

wawiwa

¿Qué son los Hooks Personalizados?

- Una función de Javascript reutilizable
- La función empieza con la palabra “use” y llama a una o más de los hooks incorporados (como useState y useEffect)
- Nos permiten extraer y reutilizar la lógica del estado entre componentes

wawiwa

¿Por qué querríamos construir nuestros propios Hooks?

- **Reutilización** - encapsula el control de estado, efectos secundarios y otras lógicas en funciones reusables que pueden ser usadas a través de multiples componentes.

- **Legibilidad mejorada** - romper lógica compleja en funciones más pequeñas y reutilizables. Esto hace que sea más fácil entender el propósito de nuestro código.

- **Abstracción** - los hooks personalizados nos pueden proveer una capa de abstracción que esconde los detalles de la implementación del manejo del estado, sus efectos secundario, y otra lógica, haciendo que sea más fácil entender y usar el componente que usa ese hook.

wawiwa

¿Cómo construir un Hook personalizado?

1. Crea un nuevo archivo para tu hook personalizado. Dale un nombre descriptivo que comience con la palabra “use”.
2. Importa de la biblioteca React los hooks que necesites.
3. Escribe una función que comience con la palabra “use” y retorna el estado y las funciones para actualizar el estado.
4. Usa el hook personalizado en tu componente.

   Puedes usarlo como cualquier otro hook, llamando a la función del hook en el nivel más alto de tu componente y desestructurando los valores retornados.

wawiwa

Ejemplo de un hook personalizado - Control de un estado alternante

En el ejemplo siguiente, tenemos un simple **hook que controla un estado alternante**:

useToggle.js :

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

export default function useToggle(initialValue = false) {
  const [value, setValue] = useState(initialValue);
  const toggleValue = () => setValue(!value);
  return [value, toggleValue];
}
```

App.js :

```
import React from 'react';
import useToggle from './useToggle';
function App() {
    const [isToggled, toggleIsToggled] = useToggle();
    return (
      <div>
        {isToggled ? 'Toggled On' : 'Toggled Off'}
        <br />
        <button onClick={toggleIsToggled}>Toggle</button>
      </div>
    );
}
export default App;
```

wawiwa

Ejercicio 15

Crea un hook de React contador.

El hook personalizado va a manejar el estado del conteo.

1. Crea el archivo useCounter.js
2. Escribe la función useCounter.js que retorne un objeto con tres propiedades: count, increment, decrement. Exporta la función con “default export”.
3. Configura el conteo inicial a 0.
4. Escribe una función increment que configure el counter+1 . Escribe una función similar para decrement.
5. En el componente padre, importa el hook useCounter y úsalo para controlar el estado counter. Muestra el valor del conteo y los botones.