React Hooks

Making React Simpler

Introduction to React Hooks

React Hooks are a feature introduced in React 16.8 that allows you to use state and other React features in functional components.

What are React Hooks?

React Hooks are functions that enable you to add state and other React features to functional components without using class components.

- Example: useState, useEffect, useContext

useState Hook

The useState Hook allows you to add state to functional components. It returns an array with two elements: the current state value and a function to update the state.

```
const [count, setCount] = useState(0);
```

useEffect Hook

The useEffect Hook enables you to perform side effects in functional components. It runs after rendering and can be used for tasks like data fetching, subscriptions, or manipulating the DOM.

```
useEffect(() => {
    // Code to run after each render
    // API Call's can go here
  }, []);
```

useContext Hook

The useContext Hook allows you to access the value of a Context in a functional component without using a Context.Consumer component.

Context: Represents a globally accessible state or data that can be shared and consumed by multiple components in a React application. More on this later!

```
const value = useContext(MyContext);
```

Benefits of React Hooks

- React Hooks simplify state management and code organization, promote reusability, and reduce the need for class components. They make functional components more powerful and expressive.
- Direct Benefits:
  - Improved readability
  - Easier code maintenance
  - Simplified testing

Conclusion

Recap: React Hooks are functions that provide state and other React features to functional components.

Key Takeaway: Utilizing React Hooks enhances the development experience by enabling state management, side effects, and improved code organization in functional components, contributing to more efficient and modern React applications.