REACT HOOKS

WHAT IS A HOOK?

- Hook is a special function that lets you hook into the React features

WHEN DO YOU USE A HOOK?

- If you write a function component and realize you need to add a some state to it, previously you had to convert it to a class.
- Now you can hook it inside the existing function component.

```
class Example extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0
    };
  }
```

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

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);
```

TYPES

- State Hook
- Effect Hook

STATE HOOK

PREREQUISITES

- this.setState()

➤ StateHook Equivalent

```
class Example extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0
    };
  }

  render() {
    return (
      <div>
        <p>You clicked {this.state.count} times</p>
        <button onClick={() => this.setState({ count: this.state.count + 1 })}>
          Click me
        </button>
      </div>
    );
  }
}
```

➤ State Hook : useState

useState returns a pair: the current state & function that lets you update it. You only need to pass the initial state value to the hook.

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

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

EFFECT HOOK

PREREQUISITES

- componentDidMount()
- componentDidUpdate()
- componentDidUnmount()

➤ EffectHook Equivalent

```
class Example extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0
    };
  }

  componentDidMount() {
    document.title = `You clicked ${this.state.count} times`;
  }

  componentDidUpdate() {
    document.title = `You clicked ${this.state.count} times`;
  }

  render() {
    return (
      <div>
        <p>You clicked {this.state.count} times</p>
        <button onClick={() => this.setState({ count: this.state.count + 1 })}>
          Click me
        </button>
      </div>
    );
  }
}
```

➤ Effect Hook : useEffect

useEffect serves same purpose as ‘componentDidMount’, ‘componentDidUpdate’, ‘componentDidUnmount’.

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

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

  // Similar to componentDidMount and componentDidUpdate:
  useEffect(() => {
    // Update the document title using the browser API
    document.title = `You clicked ${count} times`;
  });

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

RULES:

- Only call hooks at the top level.
  - Don't call Hooks inside loops, conditions or nested functions
  - This ensures that Hooks are called in same order they are rendered and that it preserves the correct state of hooks
- Only call hooks from React functions
  - Don’t call hooks from Javascript functions
  - This ensures that all stateful logic in component is clearly visible from its source code