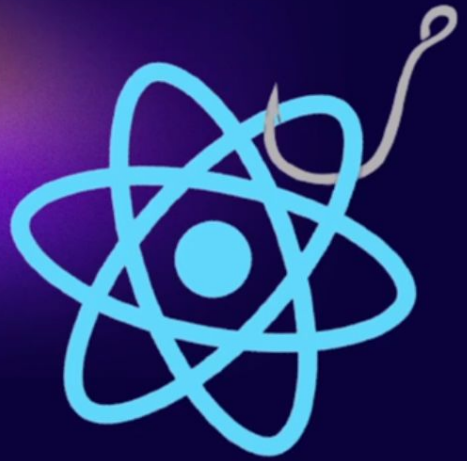
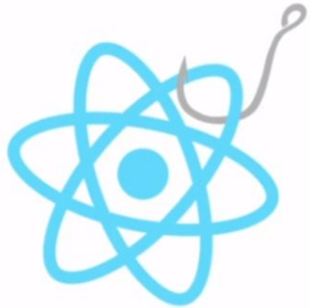
Deep dive into useState()

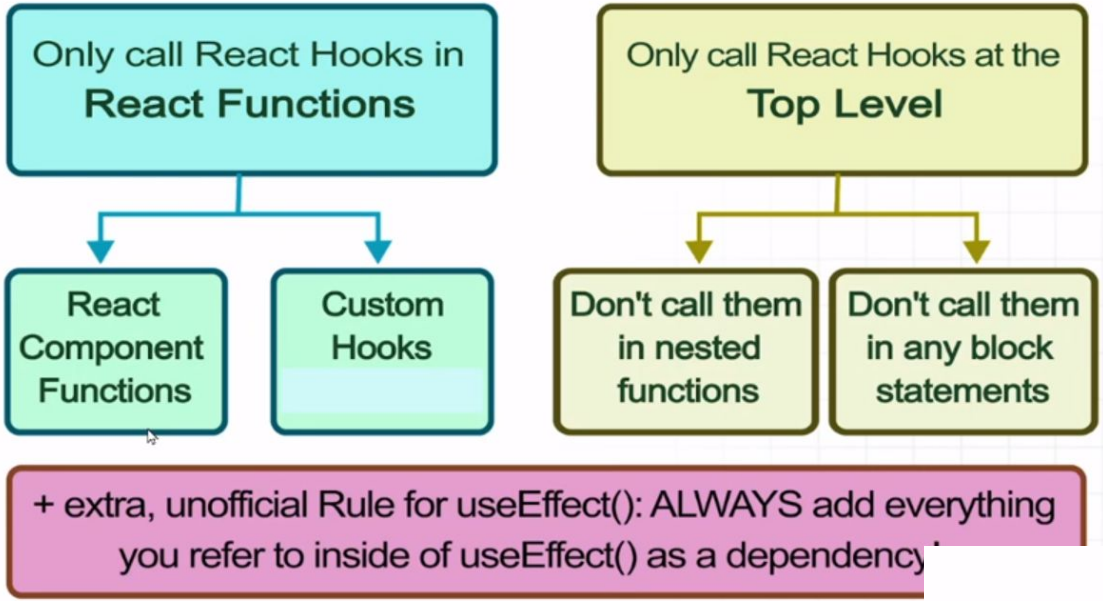


REACT HOOKS

Hooks were first introduced in React 16.8. They let us use more of React's features-like managing our component's state, or performing an after effect when certain changes occur in state(s) without writing a class.



RULES OF REACT HOOKS



useState()

What is state in react?

- A built-in react object
- Used to contain data about the component
- Changeable over time
- React component rendering and states are dependent

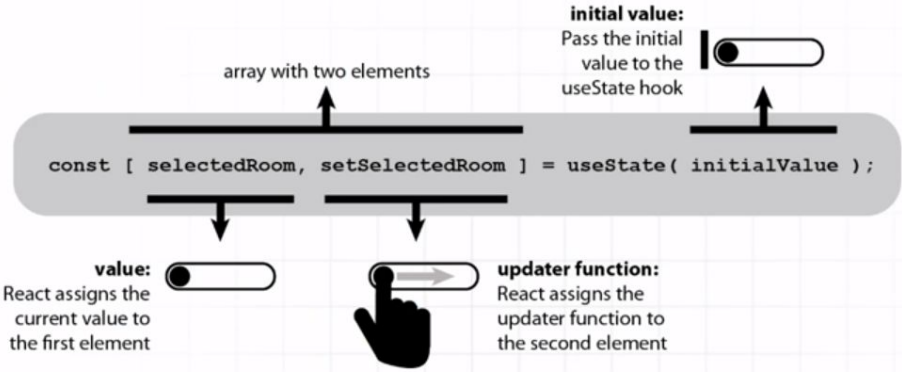
An example of state -

The contents of a form that the user is filling out. As they interact with the form fields, the component is continuously updating its state and re-rendering in order to keep the form data up to date.

What is useState?

useState exists –

- Allows to have state variables in functional components
- Takes the initial state to this function
- Returns a variable with the current state value (not necessarily the initial state) and another function to update this value



Updater function (under the hood) in useState

```
const [counter, setCounter] = useState(0);

// Use setState function form because the
//new state depends on the previous one
const clickHandlerDecrease = () => {

    // Converting the prevState to number
    //to avoid errors
    setCounter(prevState => prevState - 1);
};
```

An arrow function

```
const [counter, setCounter] = useState(0);

const clickHandlerDecrease = () => {

    setCounter(counter - 1);
};
```

Easier version

5 use cases of useState()

1. State management
1. Conditional rendering
1. Toggle flags (true/false)
1. Counter
1. Store API data in state

State management

```jsx
const UseCaseStateManagement = props => {
    const [state, setState] = useState('initial value');

    setState('new value');
    console.log(state);

    return (
        <>
            <h2>useState use case</h2>
            <h3>State management</h3>
            <hr />
            <p>{state}</p>
        </>
    );
};
```

Creates infinite loop

```jsx
const UseCaseStateManagement = props => {
    const [state, setState] = useState('initial value');

    console.log('🔄 This is a re-render');

    const clickHandler = () => {
        setState('new value');
    };

    return (
        <>
            <h2>useState use case</h2>
            <h3>State management</h3>
            <hr />
            <button onClick={clickHandler}>Set state</button>
            <p>{state}</p>
        </>
    );
};
```

Conditional rendering

```
const UseCaseConditionalRender = props => {
    const [condition, setCondition] = useState(false);

    const clickHandler = () => {
        setCondition(true);
    };

    return (
        <>
            <hr />
            <h2>useState use case</h2>
            <h3>Conditional Rendering</h3>
            <button onClick={clickHandler}>Set condition</button>
            {condition && <p>Hello!</p>}
        </>
    );
};
```

boolean initial value

Toggle flags (true/ false)

```
const UseCaseToggle = props => {
    const [mode, setMode] = useState(false);

    // Use setState function form because the new state depends on the previous one
    const clickHandler = () => {
        setMode(!mode);
    };

    const toggledClass = mode ? light : dark;

    return (
        <div className={toggledClass}>
            <hr />
            <h2>useState use case</h2>
            <h3>Toggle flags</h3>
            <button onClick={clickHandler}>Toggle display mode</button>
        </div>
    );
};
```

Counter

```jsx
const UseCaseCounter = props => {
    const [counter, setCounter] = useState(0);

    // Use setState function form because the new state depends on the previous one
    const clickHandlerDecrease = () => {
        // Converting the prevState to number to avoid errors
        setCounter(counter - 1);
    };

    const clickHandlerIncrease = () => {
        setCounter(counter + 1);
    };

    return (
        <>
            <hr />
            <h2>useState use case</h2>
            <h3>Counter</h3>
            <button onClick={clickHandlerDecrease}>--</button>
            <span> {counter} </span>
            <button onClick={clickHandlerIncrease}>++</button>
        </>
    );
};
```

numeric initial value

Store API Data In State

```
const UseCaseApi = props => {
    const [starship, setStarship] = useState('');
    const [isLoading, setIsLoading] = useState(false);

    const clickHandler = async () => {
        setIsLoading(true);

        const response = await fetch('https://swapi.dev/api/starships/10');
        const data = await response.json();
        setStarship(JSON.stringify(data, null, "\t"));

        setIsLoading(false);
    };

    let message = '';
    if (isLoading) {
        message = <p>Getting data...</p>;
    }

    return (
        <>
            <hr />
            <h2>useState use case</h2>
            <h3>Get API data and store it in state</h3>
            <button onClick={clickHandler}>Get Millennium Falcon data</button>
            <p>{message}</p>
            <pre>{starship}</pre>
        </>
    );
};
```

empty string as initial value

Thank you!