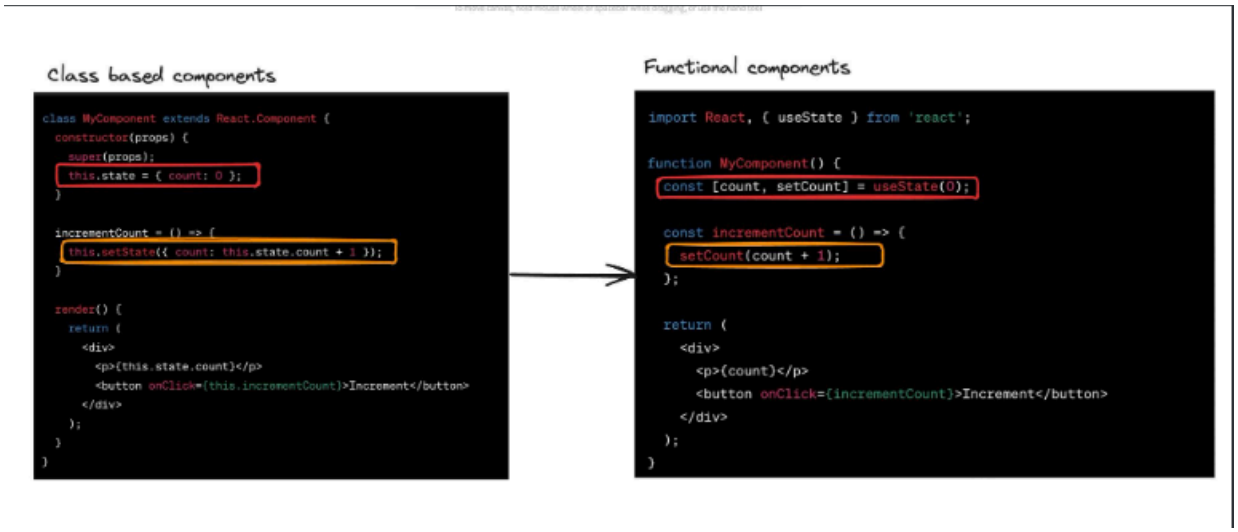
Custom Hooks

What are hooks

Hooks are a feature introduce in **React 16.8** that allow you to use state and other React features without writing a class. They are functions that let you “hook into” React state and lifecycle features from function components.

State



React is backward compatible.

Initialize a new React project

Functional Component

```
import { useState } from 'react'
import './App.css'

function App() {
  return (
    <>
      <MyComponent />
    </>
  )
}

function MyComponent() {
  const [count, setCount] = useState(0);
  // initialize a state variable using useState hook

  const incrementCount = () => {
    setCount(count + 1);
```

```
    // button clicked count increases
  };

  return (
    <div>
      {/* renders a count */}
      <p>{count}</p>
      <button onClick={incrementCount}>Increment</button>
    </div>
  );
}

export default App
```

Class based component

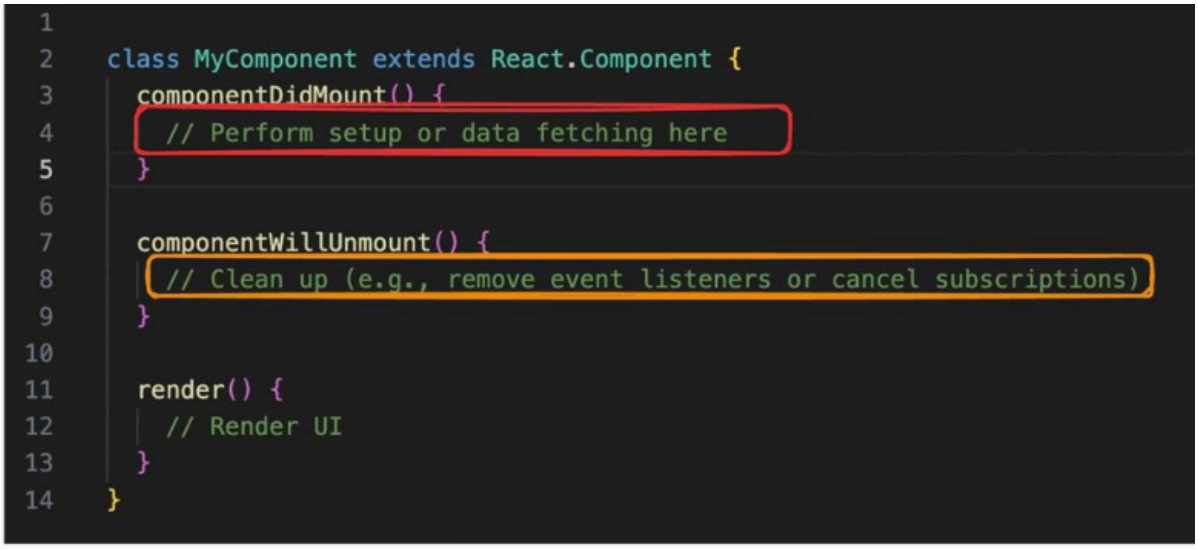
```
import React from 'react'
import './App.css'

function App() {
  return (
    <>
      <MyComponent />
    </>
  )
}
class MyComponent extends React.Component{
  constructor(props){
    super(props);
    this.state = {count: 0};
  }
  incrementCount = () => {
    this.setState({count: this.state.count + 1});
  }
  render(){
    return(
      <div>
        <p>{this.state.count}</p>
        <button onClick={this.incrementCount}>Increment</button>
```

```
      </div>
    )
  }
}
export default App
```

Return without any hooks

Life cycle events

Class based components



```
class MyComponent extends React.Component {
  componentDidMount() {
    // Perform setup or data fetching here
  }

  componentWillUnmount() {
    // Clean up (e.g., remove event listeners or cancel subscriptions)
  }

  render() {
    // Render UI
  }
}
```

Functional Components

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

function MyComponent() {
  useEffect(() => {
    // Perform setup or data fetching here

    return () => {
      // Cleanup code (similar to componentWillUnmount)
    };
  }, []);

  // Render UI
}
```

```jsx
import React from 'react'
import { useState, useEffect } from 'react';
import './App.css'

function App() {
  return (
    <>
      <MyComponent />
    </>
  )
}

function MyComponent() {
  useEffect(() => {
    // Perform setup or data fetching here
    console.log("Component Mounted")

    return () => {
      // Cleanup code (similar to componentWillUnmount)
      console.log("component unmounted")
      // we can return a function from useEffect whenever the dependencies changes or component unmount, fucntion from last render run first then line 16 code runs again
    };
  }, []);

  return <div>
    From inside my Component
  </div>
  // Render UI
}

export default App
```

I want the <MyComponent /> to be rendered for 10 seconds and then removed.(unrendered)