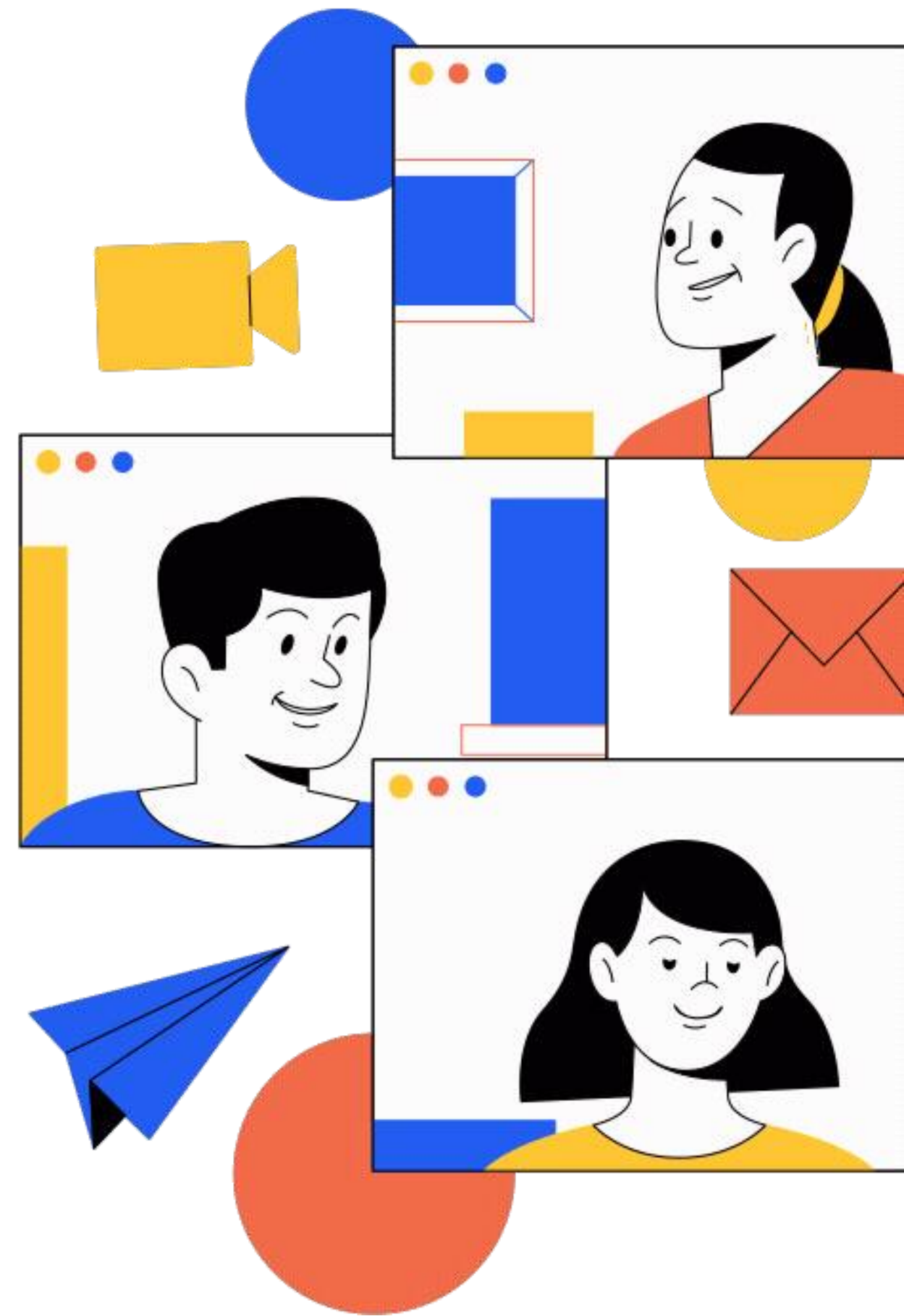


Week 2 - React

Are you ready for some state management and Hooks?

Ahmed Fouad Lotfy

React Session Lead



Agenda

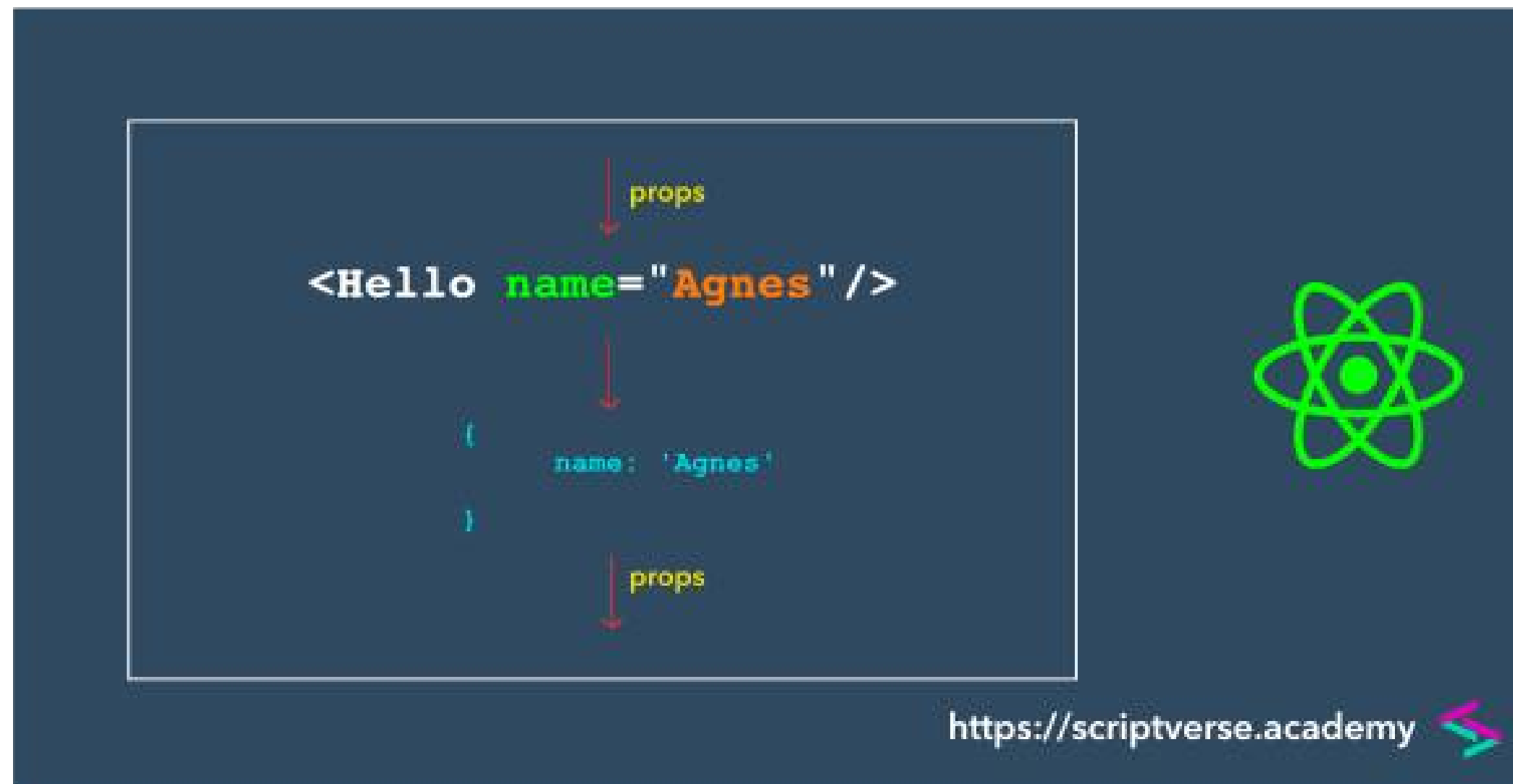


What we'll cover in this session

- What are State & Props?
- Difference between State and Props
- Managing State in React
- What are React Hooks?
- More about React Hooks
- Live Demo

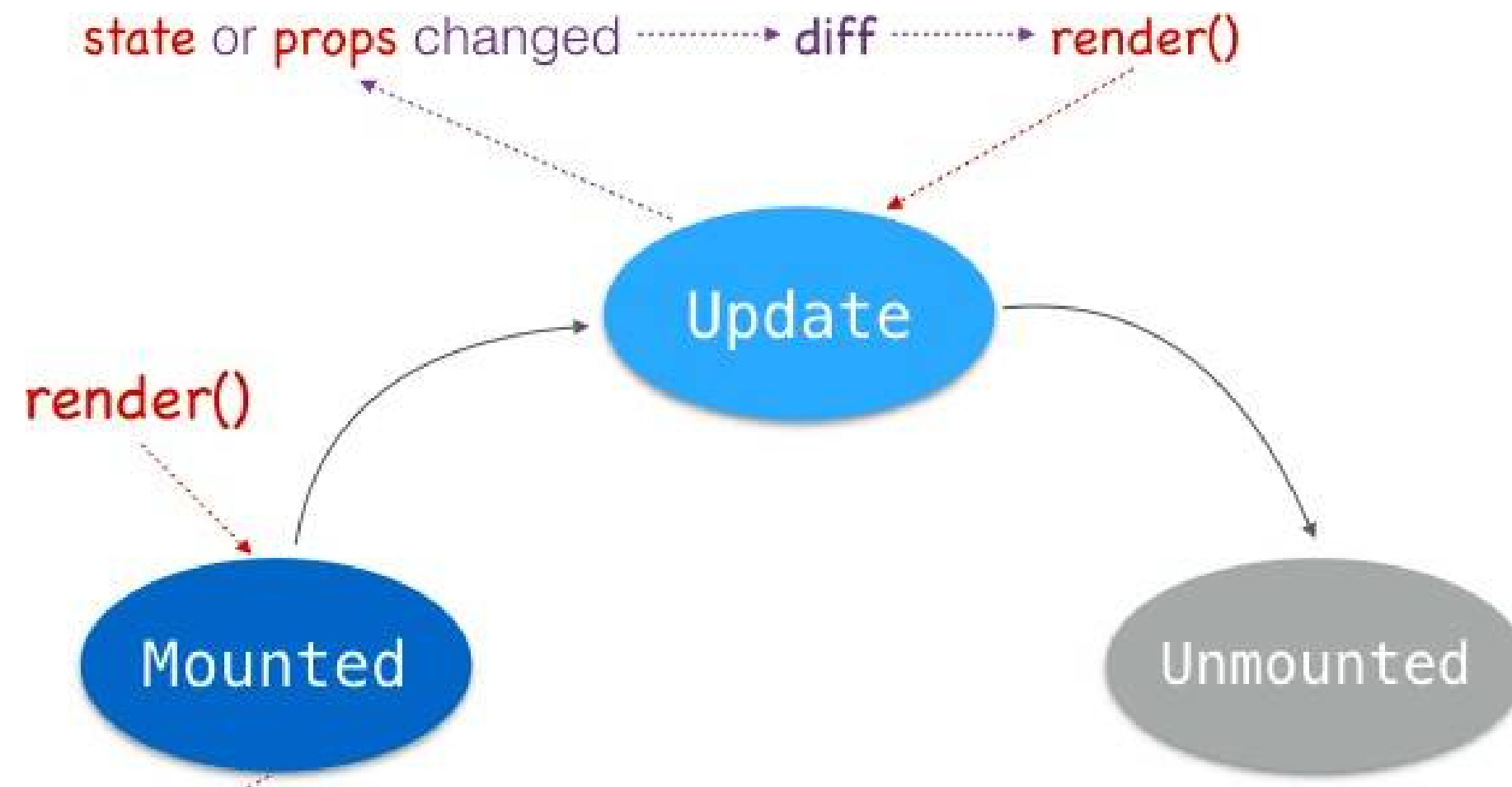
What is Props?

- Props is acronym for Properties, They are **read-only** JS objects which must be kept pure and immutable.
- They are passed from parent to child components.



What is **State**?

- State is mutable JS objects that used by react to determine or represent information about the component's current situation.
- If any part of these states change, The component will re-render .



Props



- Props are **read-only (Immutable)** JS objects.
- Passed from parent to child.
- If you want to change a prop, You must use a **callback function**.

State

- **Mutable** JS objects.
- Hold internal information about a component.
- State has methods to modify its value.
- State updates are asynchronous.



Creating the State

```
1 //Method 1: assign a variable called state
2 state = {
3   greetings: 'Hello World',
4 };
5
6 //Method 2: using a constructor
7 constructor(props) {
8   super(props);
9   this.state = {
10     greetings: 'Hello World',
11   };
12 }
```

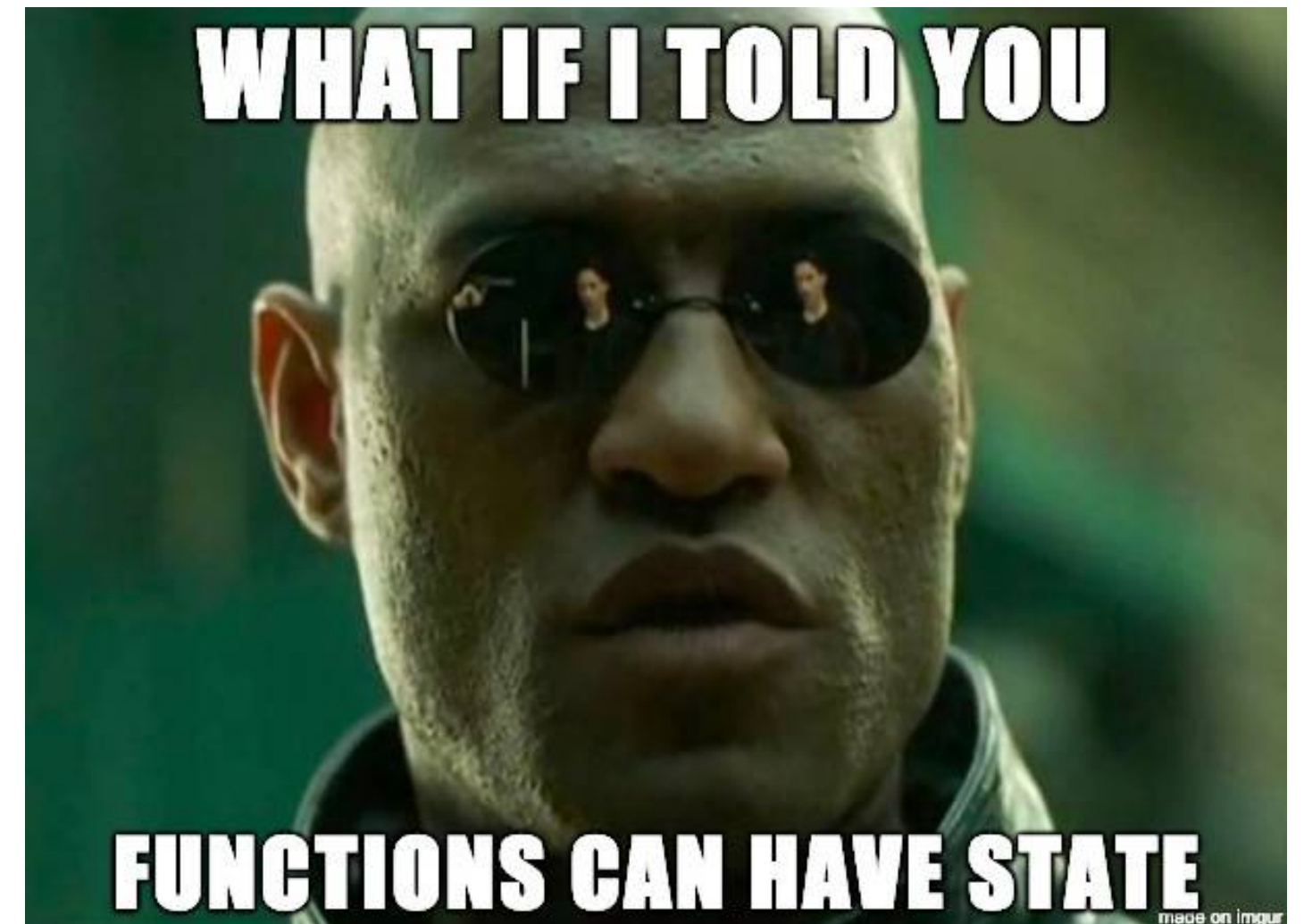


Updating the State

```
1 handleChangeName = () => {
2   //Method 1: re-assign using an object
3   this.setState({
4     greetings: 'Hello React',
5   });
6   //Method 2: re-assign using the previous state
7   this.setState((prevState) => ({
8     greetings: prevState.greetings + 'again!',
9   }));
10 }
```


React Hooks

- Hooks are the new feature introduced in the React 16.8 version.
- Hooks allows you to use **state** and other **React features** without writing a class.
- Hooks are the functions which "**hook into**" React state and **lifecycle features** from function components.
- Hooks does not work inside classes



More about React Hooks

Rules of Hooks

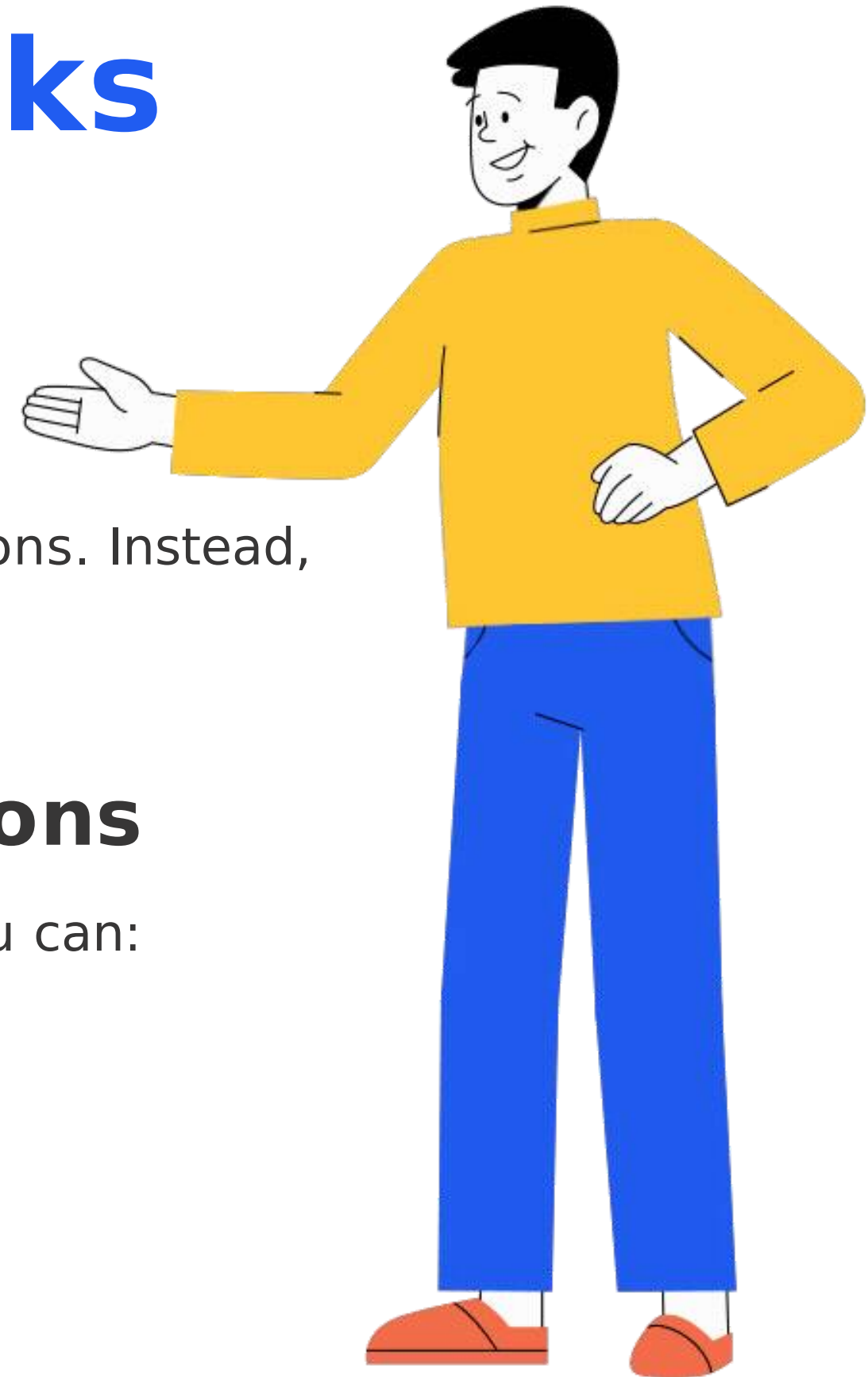
1. Only Call Hooks at the Top Level

Don't call Hooks inside loops, conditions, or nested functions. Instead, before any early returns.

2. Only Call Hooks from React Functions

Don't call Hooks from regular JavaScript functions. Instead, you can:

- Call Hooks from React Function Components.
- Call Hooks from a custom hook.



✗ Don't call Hooks inside loops, conditions, or nested functions.

✗ Bad practice

```
const App = () => {  
  
  // Nested functions  
  const handler = () => () => {  
    const [count, setCount] = React.useState(0);  
  }  
  
  return <h1>Do not call React hooks inside nested functions</h1>;  
};
```

✗ Bad practice

```
const App = () => {  
  
  for (let index = 0; index < 10; index++) {  
    let [count, setCount] = React.useState(0);  
  }  
  
  return <h1>Do not call React hooks inside loops</h1>;  
};
```

✗ Bad practice

```
const App = () => {  
  
  if (true){  
    let [count, setCount] = React.useState(0);  
  }  
  
  return <h1>Do not call React hooks inside conditions</h1>;  
};
```

✓ Instead, Always use Hooks at the top level of a React function

✓ Good practice

```
const App = () => {  
  const [count, setCount] = React.useState(0);  
  React.useEffect(sideEffectCallback);  
  const [person, setPerson] = React.useState({});  
  
  // Loops, conditions, nested functions, etc...  
  
  return <h1>Good example</h1>;  
};
```

✗ Don't call Hooks from regular JavaScript functions.

✗ Bad practice

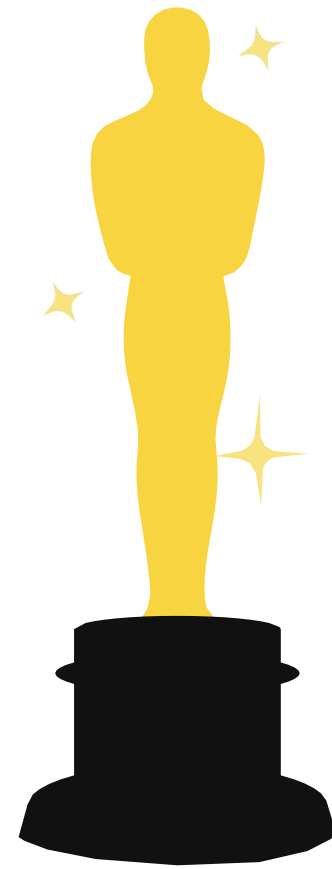
```
function useCustomHook() {  
  return [count, setCount] = React.useState(0);  
}
```

```
function regularFunc() {  
  const [count] = useCustomHook();  
}
```

✓ Instead, Call hooks from React function component or Custom Hooks

✓ Good practice

```
function useCustomHook() {  
  const [count, setCount] = React.useState(0);  
  
  // on mount hook  
  React.useEffect(() => {  
    setInterval(() => {  
      setCount(state => state + 1);  
    }, 1000);  
  }, []);  
  
  return count;  
}  
  
const App = () => {  
  const count = useCustomHook();  
  
  // Loops, conditions, nested functions, etc...  
  
  return <h1>Good example: {count}</h1>;  
};
```



Most Famous/Used React Hooks



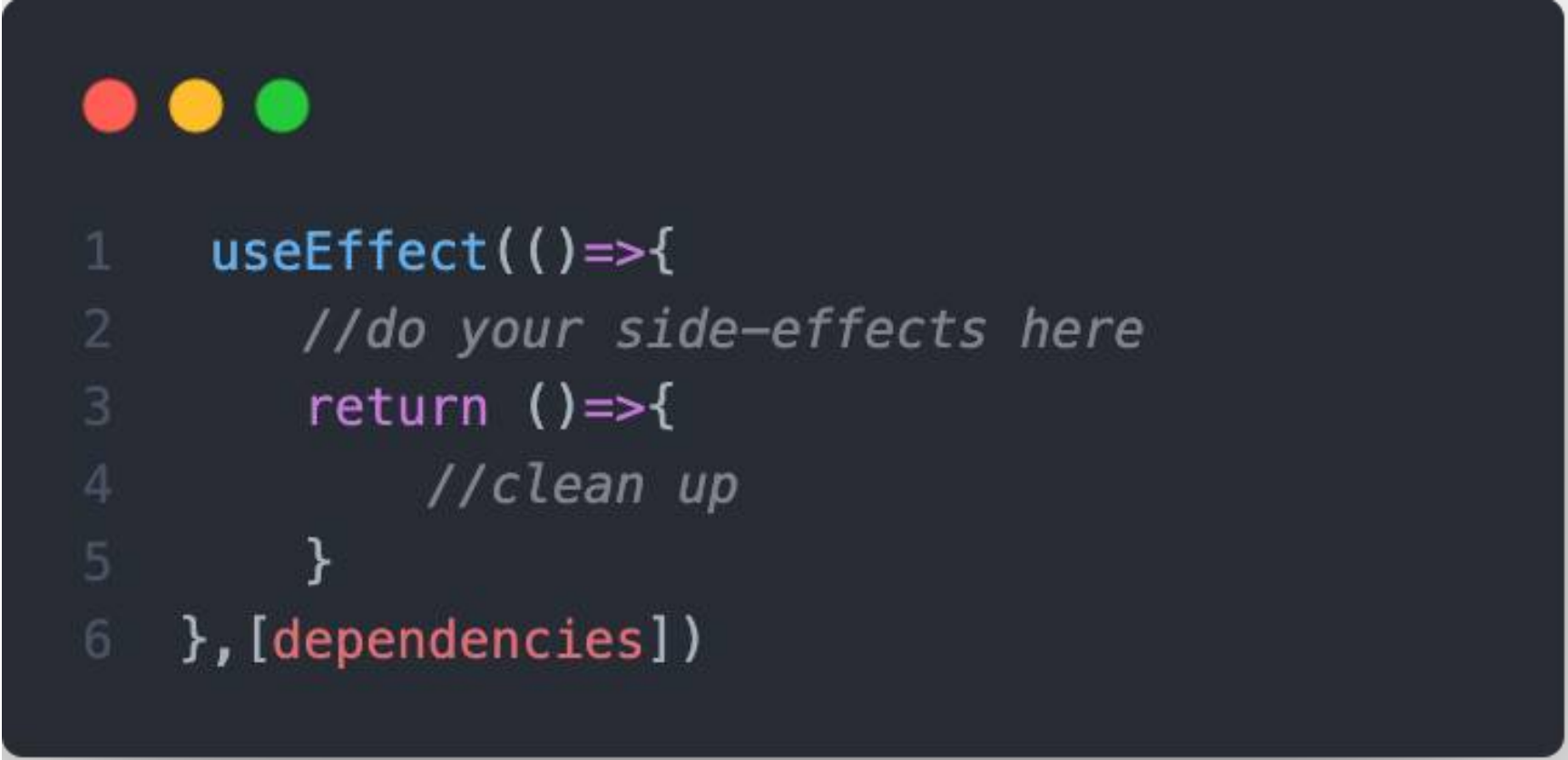
1. useState

- The React useState Hook allows us to track state in a function component.
- useState takes the **initial state** as its argument
- useState hook returns an array that contains two elements **[currentState, setState]**

```
1 //Using a class component
2 class Message extends React.Component {
3   constructor(props) {
4     super(props);
5     this.state = {
6       message: '',
7     };
8   }
9   render() {
10    return <div>{this.state.message}</div>;
11  }
12 }
13 //Using functional component and React Hooks
14 const Message = () => {
15   const [message, setMessage] = useState('');
16
17   return <div>{message}</div>;
18 };
19
```


2. useEffect

- The Effect Hook lets you perform side effects in function components.
- **Examples for Side effects**
 - Data Fetching
 - Manually changing DOM elements
 - Setting up subscription
- *useEffect* hook runs immediately after the component is mounted.
- *useEffect* will also run on changing any member of the dependency array.



```
1  useEffect(()=>{  
2    //do your side-effects here  
3    return ()=>{  
4      //clean up  
5    }  
6  },[dependencies])
```

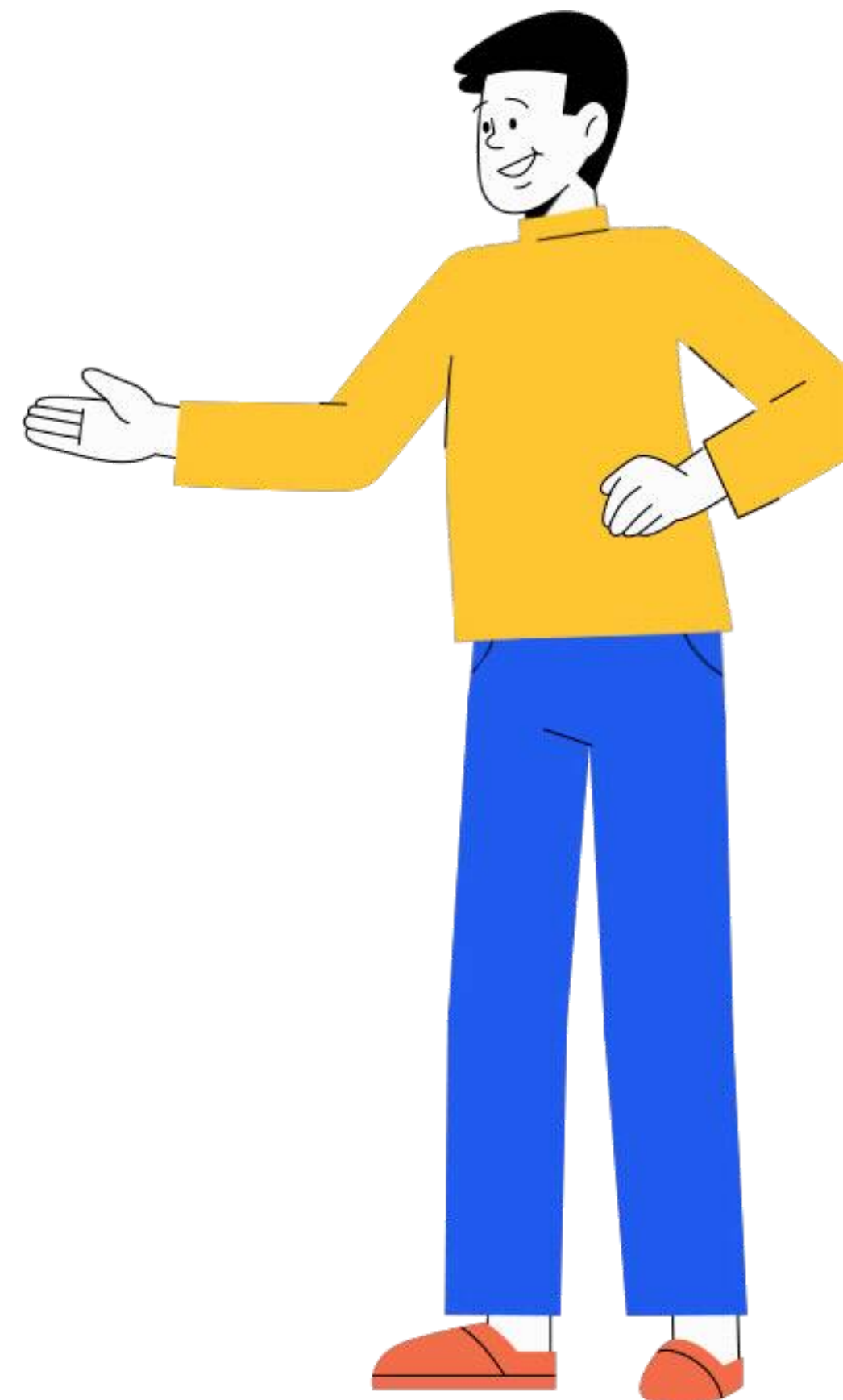
useEffect Example

```
1 import React, { useState, useEffect } from 'react';
2
3 function Example() {
4   const [count, setCount] = useState(0);
5
6   // Similar to componentDidMount and componentDidUpdate:
7   useEffect(() => {
8     // Update the document title using the browser API
9     document.title = `You clicked ${count} times`;
10  }, [count]);
11
12  return (
13    <div>
14      <p>You clicked {count} times</p>
15      <button onClick={() => setCount(count + 1)}>Click me</button>
16    </div>
17  );
18 }
19
```



Custom Hooks

Build your own hook right now



You can find so many pre-made custom hooks at [this link](#)



Demo Time

Grab your coffee and let's get started. ☕

Meme of the day





Any Questions?

A wise man once said,
"There is no dump questions"

Thank you!

For questions, requests and anything, please reach out to me on circle or email me at a.lotfy@fci-cu.edu.eg



Follow me on Github [@ahmeddxfoad](https://github.com/ahmeddxfoad)
code and slides are found at this [github repo](#)

