Introduction to React Hooks

Web Development Boot Camp

Lesson 20.1

Partner Activity: Managing React State

Take a moment and discuss the different methods used in React state management with the person sitting next to you.

Consider the following:

What are some of the advantages/disadvantages with these methods?

Suggested Time:
5 minutes

Managing state can be difficult because there is no one-size-fits-all solution.

But there is another way . . .

In This Lesson, We Will Cover Two Hooks

01 `useState`: Allows you to use state in a functional component.

02 `useEffect`: Replaces lifecycle methods like `componentDidMount` and `componentDidUpdate`.

03 **Custom Hooks:** Create your own reusable hooks!

Effect is a term used to describe the result of affecting the “outside world.” This includes data fetching, subscribing to events, and making changes to the DOM.

Comparing Ways to Manage State

01 Class Components with `setState()`

Advantages

- Component and children will re-render with up-to-date data.

Disadvantages

- Updating state from nested components can be difficult.
- Since state only flows one way, all components that need access to the state must be children of the same stateful component.

02 Functional Components with `useState()`

Advantages

- Easier to read, debug, and no need to use “this”
- Access to “Hooks”

Disadvantages

- Will need to use other “Hooks” to manage complex levels of state.
- Not supported with older codebases, will still have to use Class Components for state.

As of React 16.8,
Facebook recommends to use functional components
whenever possible.

Introducing React Hooks

Hooks are functions that let you "hook into" React state and lifecycle features from stateless components.

In This Lesson, We Will Cover Two Hooks

01 `useState`: Allows you to use state in a functional component.

02 `useEffect`: Replaces lifecycle methods like `componentDidMount` and `componentDidUpdate`.

03 **Custom Hooks:** Create your own reusable hooks!

Effect is a term used to describe the result of affecting the "outside world."
This includes data fetching, subscribing to events, and making changes to the DOM.

The Two Rules of Hooks

01

Do not call hooks from within loops, conditionals, or nested functions.

- It is important that hooks are always called in the same order, like component lifecycle methods.

- It is also what makes it possible for React to store the state of hooks when using `useState` or `useEffect`.

02

Do not call hooks from within regular JavaScript functions.

- This makes it so that all stateful logic is easy to find for the developer (you).

<Time to Code>