13.02.2023

Advanced React

Refs

Refs

Refs provide a way to access DOM nodes or React elements created in the render method.

When to use

Managing focus, text selection, or media playback.

Triggering imperative animations.

Integrating with third-party DOM libraries.

When not to use

Avoid using refs for anything that can be done declaratively.

React.createRef()

The ref prop is used to return a reference to the element.

When a ref is passed to an element in render, a reference to the node becomes accessible at the current attribute of the ref

React.forwardRef()

Ref forwarding technique is used for exposing DOM Refs to Parent Components.

This is generally not recommended because it breaks component encapsulation.

https://codesandbox.io/s/refs-4jjc55?file=/src/App.tsx

Portals

Portals

Portals provide a first-class way to render children into a DOM node that exists outside the DOM hierarchy of the parent component.
It is useful for implementing popups, toasts, tooltips, etc.

Portals

Event Bubbling will work according to React tree ancestors, regardless of the Portal node location in the DOM.

Context and lifecycle work the same way since the Portal still exists in the React tree.

https://codesandbox.io/s/portal-ogtrlg

Hooks

Hooks

Hooks let you use state and other React features without writing a class. They let you “hook into” React state and lifecycle features from function components.

Hooks are made for function components.

Hooks: Motivation

It’s hard to reuse stateful logic between components.

Hooks allow you to reuse stateful logic without changing your component hierarchy.

Complex components become hard to understand.

Hooks let you split one component into smaller functions based on what pieces are related (such as setting up a subscription or fetching data)

Classes confuse both people and machines.

Hooks let you use more of React's features without classes.

Hooks: Rules

Hooks are JavaScript functions, but they impose two additional rules:

Only call Hooks at the top level.

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

By following this rule, you ensure that Hooks are called in the same order each time a component renders.

Only call Hooks from React function components.

Don’t call Hooks from regular JavaScript functions.

By following this rule, you ensure that all stateful logic in a component is clearly visible from its source code.

Basic Hooks

Basic hooks

React.useState()

Think of useState Hook as combination of **this.state** and **this.setState.**

React.useEffect()

Think of useEffect Hook as **componentDidMount, componentDidUpdate,** and **componentWillUnmount** combined.

React.useContext()

Think of useContext Hook as **Context.Consumer .**

https://codesandbox.io/s/basic-hooks-1i0lj7

Context

Context

Context is designed to share data that can be considered “global” for a tree of React components.

Context: When to use

To avoid prop drilling

If you have static data that undergoes lower-frequency updates
such as preferred language, time changes, location changes,
and user authentication,
passing down props with React Context may be the best option.

Context: When not to use

If your state is frequently updated, React Context may not be as effective or efficient as for example a state management tool like **React Redux** or **MobX**.

React.CreateContext()

Context.Provider() **Context.Consumer()** or **Class.contextType (staticContextType)**

You can only subscribe to a single context using this API.

https://codesandbox.io/s/context-644d1y

Exercise time

https://codesandbox.io/s/exercise-1-jmmp4j

Additional Hooks

Aditional hooks

React.useCallback()

React Hook that lets you cache a function definition between re-renders.

React.useMemo()

React Hook that lets you cache the result of a calculation between re-renders.

React.useRef()

React Hook that lets you reference a value that's not needed for rendering.

https://codesandbox.io/s/additional-hooks-t55soi

Exercise time

https://codesandbox.io/s/exercise-2-26fov8

Additional Resources

1. https://beta.reactjs.org/
2. https://reactjs.org/
3. https://kentcdodds.com/blog?q=react
4. https://www.youtube.com/watch?v=eFGeStq8dZo - Using **useEffect** Effectively
5. https://www.youtube.com/watch?v=BXTU4NmMu8A - React deep dive

Evolution Engineering | ACCELERATING EVOLUTION