

Advanced Techniques



SoftUni Team

Technical Trainers



SoftUni



Software University

<https://softuni.bg>

sli.do

#react

Table of Contents

1. Context
2. HOC
3. Reducers
4. Error Boundaries
5. Unit Testing with JEST





Context

What is Context?

- **State Management**

- React **Context API** allows you to manage and share state across multiple components without having to pass **props** down manually at every level, making it easier to manage global state

- **Provider-Consumer Pattern**

- It uses a Provider to supply the state and a Consumer or the **useContext** hook to access the state in any component within the provider's tree.

- **Avoids Prop Drilling**

- By enabling a more efficient way to pass data through the component tree, it helps to avoid "**prop drilling**", where props are passed through many intermediate components unnecessarily.



```
export const Context = React.createContext();

export const ContextProvider = ({children}) => {
  const [state, setState] = React.useState({});

  return (
    <Context.Provider value={state}>
      {children}
    </Context.Provider>
  )
}
```



Higher-Order Components

Advanced Composition and Decoration

Higher-Order Components

- A **higher-order component (HOC)** is an advanced technique in React for reusing component logic
- **HOCs** are not part of the React API
- **HOC** is a function that takes a component and returns a new component



- Components are the primary unit of code reuse
 - Some patterns aren't straightforward for traditional components
- Whereas as component transforms props into UI
 - **HOC** component transform a component into another component

```
const EnhancedComponent = higherOrderComponent(WrappedComponent);
```

```
function hocFunc(WrappedComponent) {  
  return function Component(props) {  
    render() {  
      return <WrappedComponent {...props} />;  
    }  
  };  
}
```



UseReducer

- **useReducer**
 - An alternative to useState
 - Accepts a reducer of type **(state, action) => newState**
 - Return the current state paired with a **dispatch** method
 - Preferable when you have complex state logic

```
const [state, dispatch] = useReducer(reducer, initialState);
```



Error Boundaries

Error Boundaries

- **Error boundaries** are React components
 - **Catching, logging** and **displaying** JS error anywhere in their child component tree
- They catch errors during rendering
- **Do not** catch errors for
 - Event handlers
 - Asynchronous code
 - Server-side rendering



- A component becomes an **error boundary** if it defines
 - **static `getDerivedStateFromError`**
 - Render a fallback UI after an error has been thrown
 - **`componentDidCatch`**
 - Log error information
- You can use it as a regular component

```
<ErrorBoundary>  
  <MyWidget />  
</ErrorBoundary>
```

- Error boundaries work like a JavaScript **catch {}** block for component
- Only **class component** can be error boundaries
- Declare an error boundary component **once** and use it throughout your application

- You may wrap **top-level** route components to display some error message
- Wrapping individual widgets in an error boundary
 - Protect them from crashing the rest of the app



JEST

Unit Testing

What is JEST?

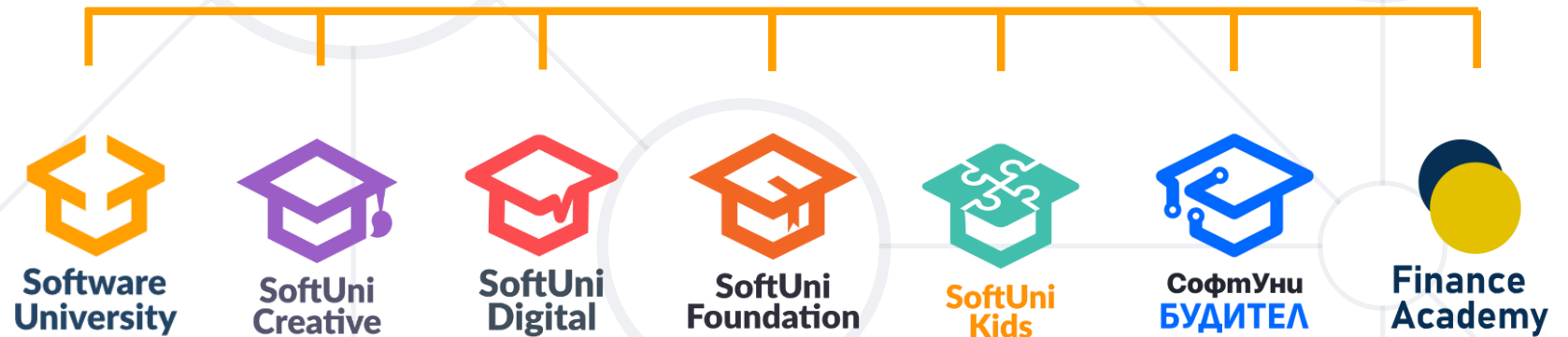
- Jest is a JavaScript unit testing **framework**
 - Used by Facebook to **test** services and React applications
- Jest acts as a **test runner, assertion library and mocking library**
- Jest provides **Snapshot testing**
 - Create a rendered 'snapshot' and compare it to a previous
- <https://jestjs.io/>



- Context provides way to pass data through the component without passing the props manually
 - Context API
- HOC
- `useReducer`
- Error Boundaries
- Unit Testing



Questions?



SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

