Advanced React JS

Advanced React Hooks

Learning Objectives

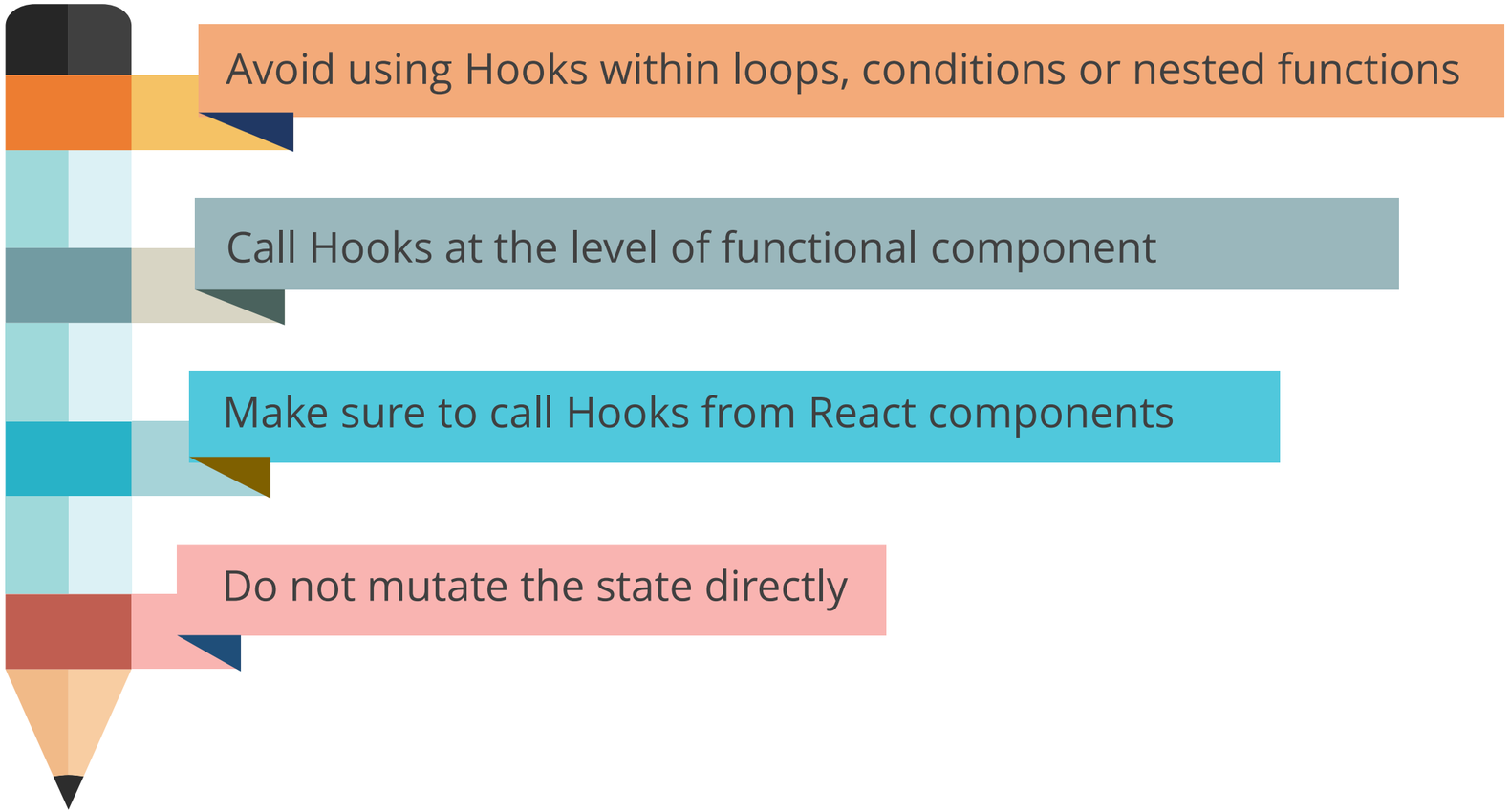
By the end of this lesson, you will be able to:

- Define the concept of React Hooks and their role in functional components
- Design and implement custom Hooks to reuse complex stateful logic across multiple components
- Use useMemo Hooks to optimize components
- Describe some of the key aspects of React performance to enhance React application efficiency and user experience

Introduction To React Hooks

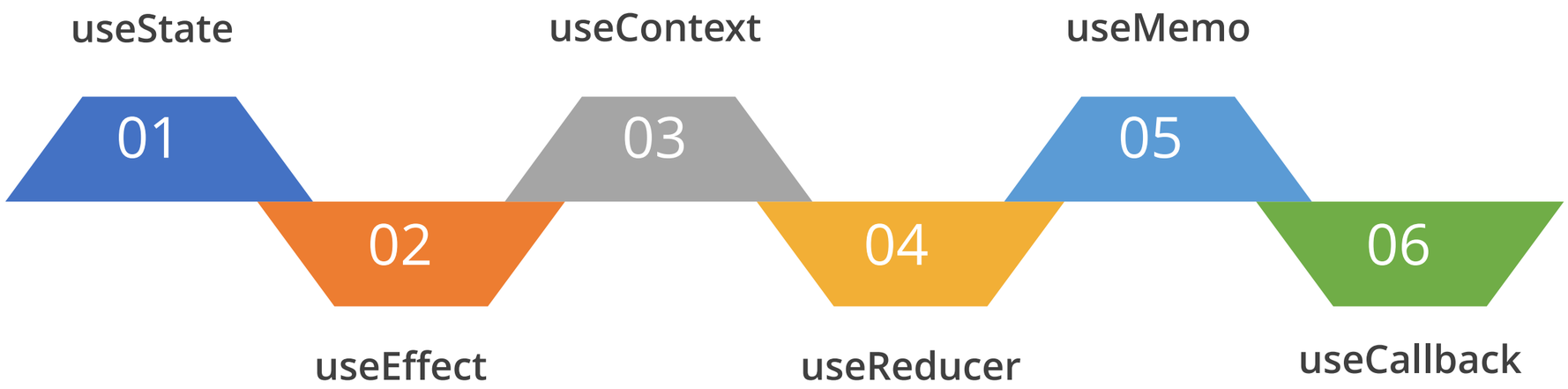
Introduction to React Hooks

These are JavaScript functions that let you Hook into React state and lifecycle features from functional components. Hooks have four golden rules, which are as follows:



Types of React Hooks

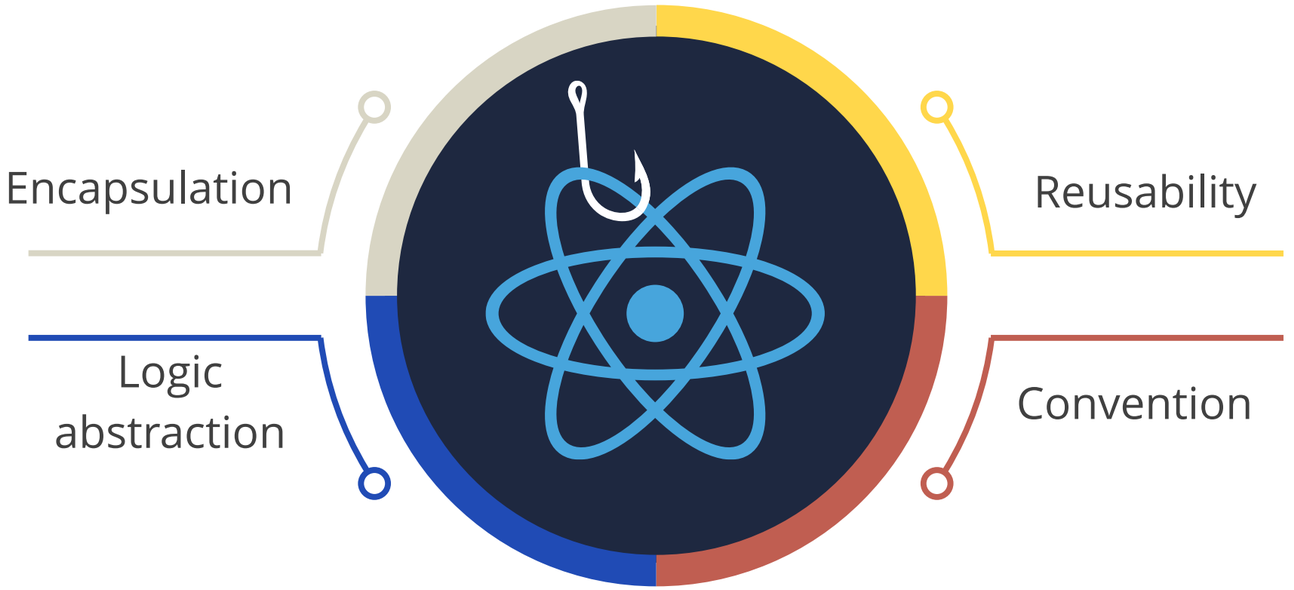
There are many different React Hooks available, each with its purpose.
The most common types are as follows:



Custom Hooks

What Are Custom Hooks?

These are functions that connect or integrate React state and lifecycle functionalities into function components. Some important points to note about custom Hooks are as follows:



These Hooks help to reuse logic related to managing state between components.

Creating a Simple React Custom Hook

Duration: 20 min

Problem Statement:

You have been assigned the task to perform the steps to create a simple react Hook application to demonstrate the basic click.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed:

1. Create and set up the React JS project
2. Create a custom Hook file
3. Use a custom Hook in user-defined components
4. Import user-defined functional components in the App.js file
5. Run the application

Creating Custom Hooks for Reusing Stateful Logic

Using custom Hooks in React is a way to reuse logic across different components. The steps to effectively utilize custom Hooks for this purpose are as follows:

Creating Form Validation Using a Custom Hook

Problem Statement:

Duration: 20 min

You have been assigned the task to perform the steps to create a form validation using custom Hook for enhanced user experience.

ASSISTED PRACTICE

Assisted Practice: Guidelines

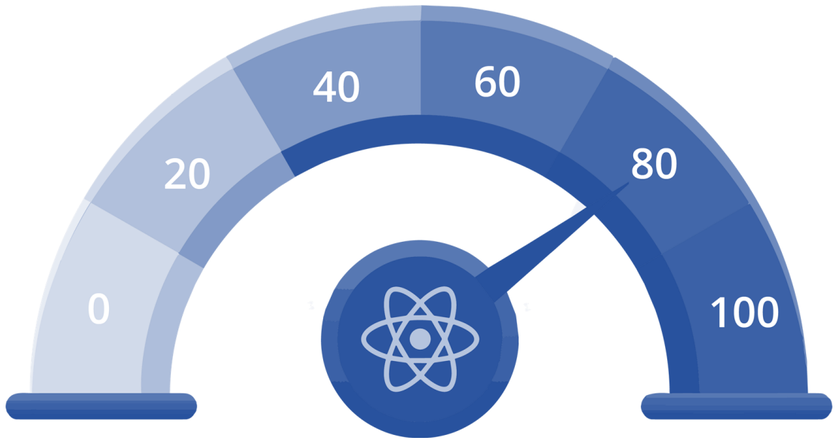
Steps to be followed:

1. Create and set up the React JS project
2. Create a custom Hook file
3. Use a custom Hook in user-defined components
4. Import user-defined functional components in the App.js file
5. Add bootstrap features in the index.html file
6. Run the application

React Performance Optimization

What Is React Performance Optimization?

It is the process that involves enhancing the responsiveness and performance of a React application.



By optimizing React applications, developers can ensure that their web applications load quickly, run smoothly, and deliver a seamless user experience.

Key Aspects of React Performance Optimization

Minimize re-renders: Efficiently update the UI by comparing changes between the previous and current state using the virtual document object model

Optimize data fetching: Optimize data fetching by utilizing cache and implementing stale-while-revalidate data

Optimize images and assets: Compress images and other assets, such as videos and fonts to reduce their file size and loading time

Use code splitting: Break down the application into smaller bundles using techniques like Webpack's code splitting

Key Aspects of React Performance Optimization

Optimize dependencies: Utilize tools to analyze the bundle size and identify unused dependencies

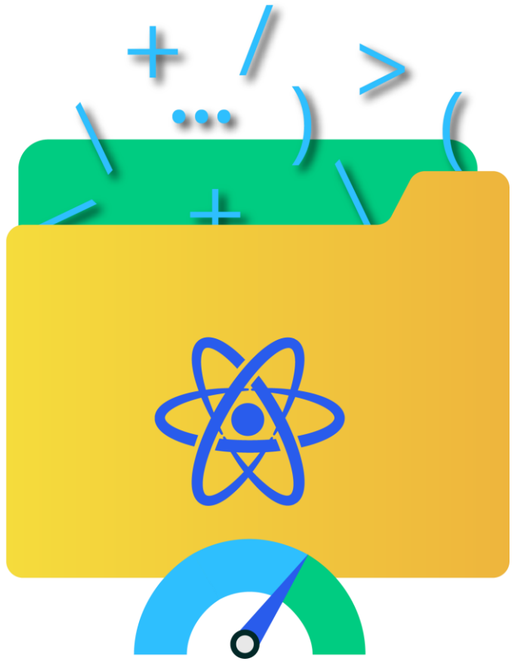
Memoize components: Use **React.memo** to memoize components, preventing unnecessary re-renders when their props have not been changed

Utilize react libraries: Leverage performance-focused libraries for efficient rendering of large lists or tables

Monitor performance: Monitor the application's performance regularly to identify and proactively address performance bottlenecks

What Is Memoization?

It is a method to optimize programming by storing and reusing expensive function call results, reducing unnecessary calculations. It is commonly utilized for rendering optimization in React.

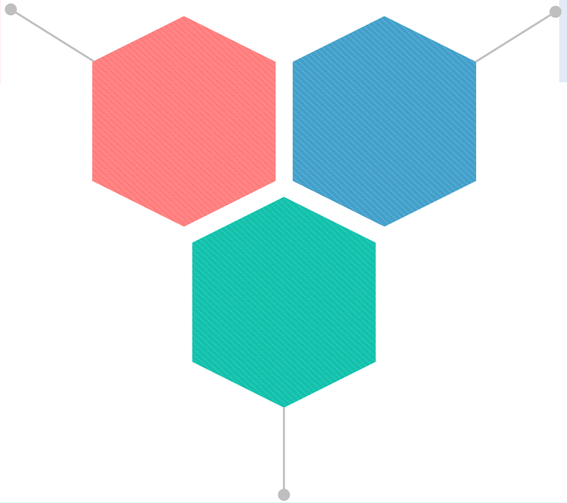


The **React.memo** higher-order component and the **useMemo** Hook allows remember details (memoization) of simple components and values.

Why Should Memoization Be Used?

Performance optimization

Reduces memory usage



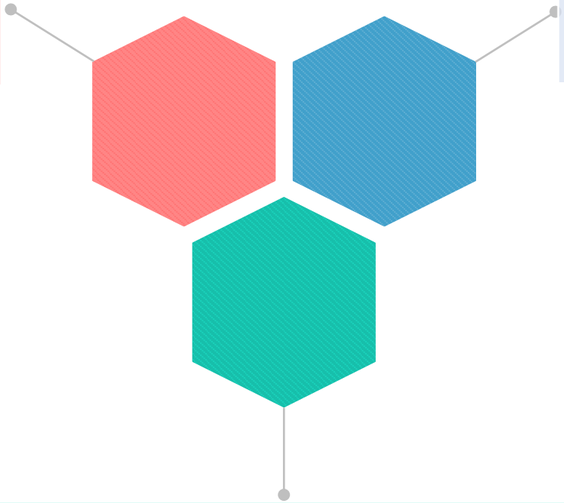
Reduces unnecessary re-renders

Improves application responsiveness

Why Should Memoization Be Used?

Improved code clarity and maintainability

Improves code testability

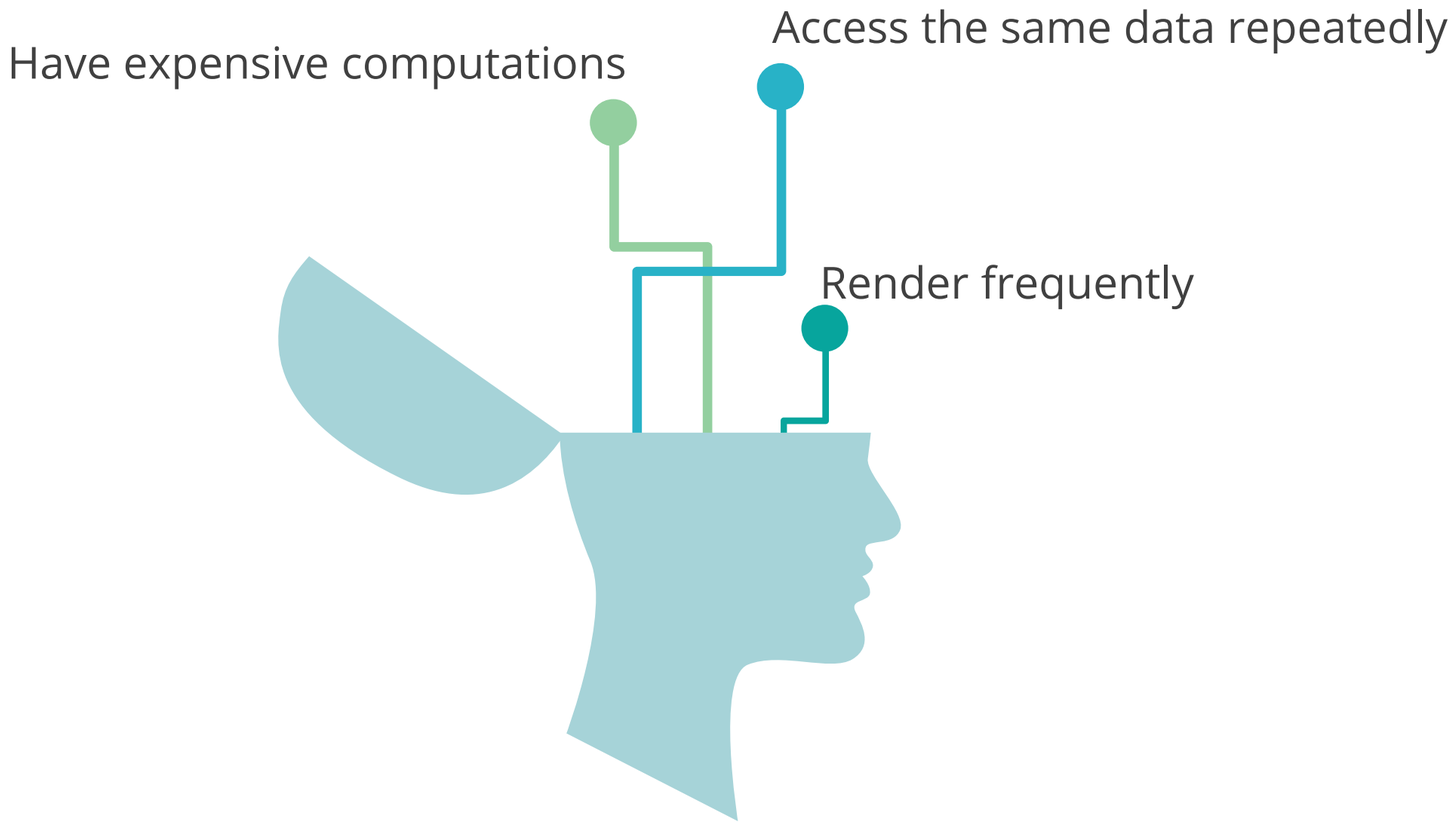


Makes components more predictable

Reduces code complexity

When To Use Memoization In React?

It is most beneficial in situations where components:



How To Use Memoization In React?

React provides two built-in Hooks for memorization, **useMemo** and **useCallback**.

useMemo

```
const memoizedValue = useMemo(() => {
  // Expensive computation or logic
}, [dependency1, dependency2, ...]);
```

useCallback

```
const memoizedFunction = useCallback((parameter1,
parameter2) => {
  // Function logic
}, [dependency1, dependency2, ...]);
```

useMemo is used to memoize a function's result.

useCallback is used to memoize a function itself.

These take two arguments, the function to memoize and an array of dependencies.

Using the useMemo Hook

Duration: 10 min

Problem Statement:

You have been assigned a task to perform the steps involved in demonstrating the useMemo Hook to optimize the performance.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed:

1. Create and set up React JS project
2. Perform expensive function calls: without **useMemo**, with **useMemo**, and with **useMemo** with dependencies
3. Test the application

ASSISTED PRACTICE

Key Takeaways

- React Hooks are JavaScript functions that let you Hook into React state and lifecycle features from function components.
- Custom Hooks promote code reuse and enhance component organization.
- useMemo and useCallback Hooks effectively memoize functions and values.
- Memoization can significantly improve performance by caching expensive calculations.

Knowledge Check

Knowledge Check

1

What is the purpose of the useState Hook?

A. To manage local state in a function component

B. To handle side effects in a function component

C. To provide access to React context

D. To manage state based on a reducer function

Knowledge Check 1

What is the purpose of the useState Hook?

A. To manage local state in a function component

B. To handle side effects in a function component

C. To provide access to React context

D. To manage state based on a reducer function

The correct answer is **A**

The useState Hook handles local state in a function component. It helps to announce and change state variables without using a class component.

Knowledge Check 2

Which React hook is used for memoizing the callback functions in functional components?

A. useCallback

B. useMemo

C. useEffect

D. None of the above

Knowledge Check 2

Which React hook is used for memoizing the callback functions in functional components?

A. useCallback

B. useMemo

C. useEffect

D. None of the above

The correct answer are **A**

The useCallback Hook is used to memoize callback functions in functional components, preventing unnecessary re-creations of the function on each render.

Knowledge Check

3

In React, what function can be used for Memoizing a functional component?

A. React.optimize()

B. React.memo()

C. React.cache()

D. React.pureComponent()

Knowledge Check 3

In React, what function can be used for Memoizing a functional component?

A. React.optimize()

B. React.memo()

C. React.cache()

D. React.pureComponent()

The correct answer are **B**

React.memo() in React can be used to memoize functional components, preventing unnecessary re-renders when the props of a component have not changed.