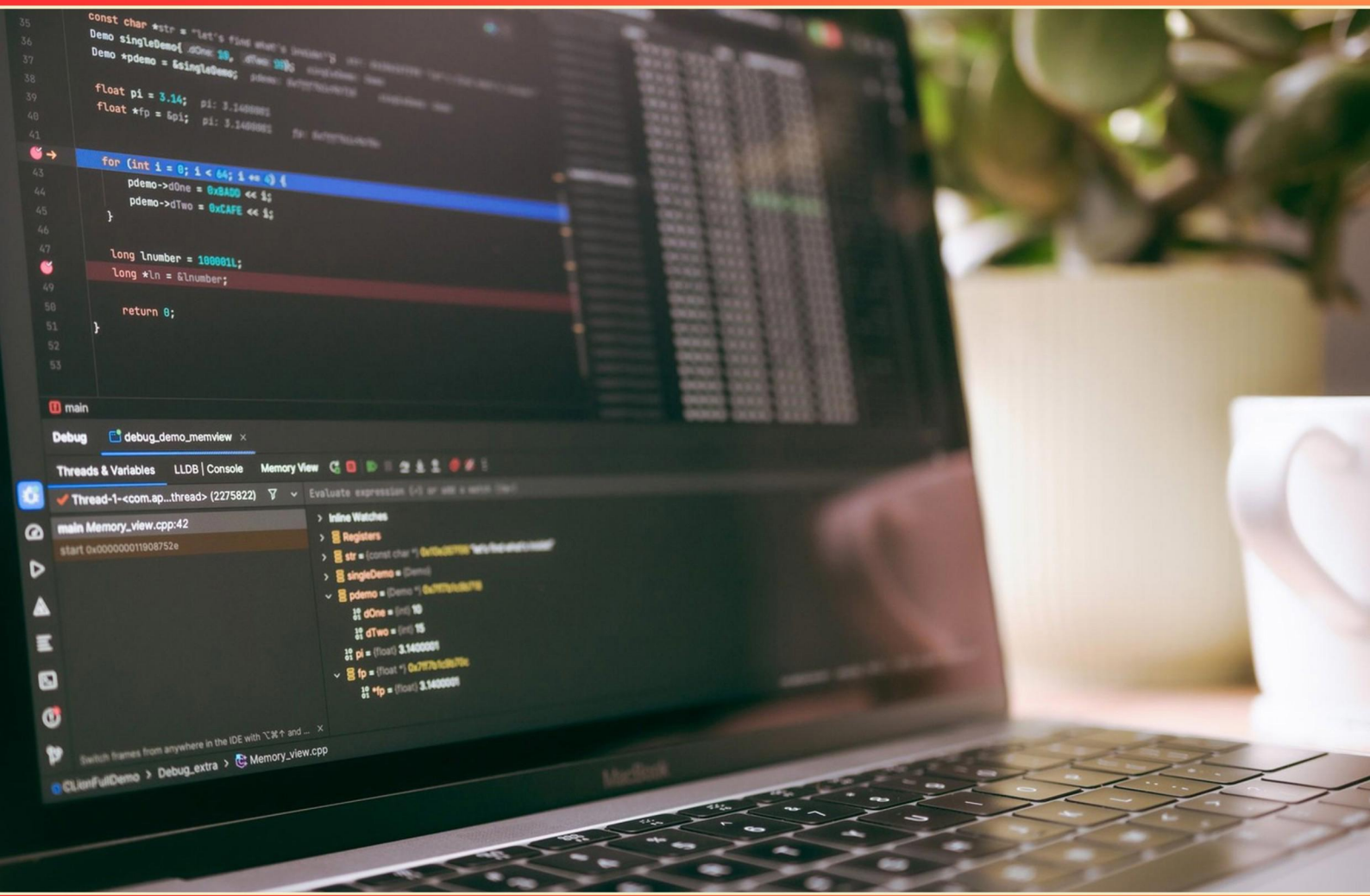
REACTJS PRACTICE TEST (SAMPLE)

STUDY GUIDE



SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://reactjs.examzify.com

Copyright © 2026 by Examzify - A Kaluba Technologies Inc. product.

ALL RIGHTS RESERVED.

No part of this book may be reproduced or transferred in any form or by any means, graphic, electronic, or mechanical, including photocopying, recording, web distribution, taping, or by any information storage retrieval system, without the written permission of the author.

Notice: Examzify makes every reasonable effort to obtain accurate, complete, and timely information about this product from reliable sources.

SAMPLE

Table of Contents

Copyright 1

Table of Contents 2

Introduction 3

How to Use This Guide 4

Questions 5

Answers 8

Explanations 10

Next Steps 15

SAMPLE

Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- Practice answering questions under realistic conditions,
- Improve accuracy and speed,
- Review explanations to strengthen weak areas, and
- Approach the exam with greater confidence.

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

SAMPLE

How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here's how to get the most out of your Examzify study guide:

1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

6. Repeat and Review

Don't just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There's no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

SAMPLE

Questions

SAMPLE

1. JSX element

A. A basic unit of JSX found in a JavaScript file which is treated as, and has the functionality of, a JavaScript expression.

B. A React component syntax for class definitions

C. A way to define CSS styles inline

D. A runtime function to access DOM elements

2. Which statement correctly describes how to access the props object inside a React class component?

A. Use this.props.

B. Use this.state.

C. Use props API directly

D. Use this.context.

3. You cannot call this.setState from the body of this update lifecycle method.

A. ComponentWillUpdate

B. ComponentWillMount

C. Render

D. ComponentDidUpdate

4. What is the primary purpose of a custom hook in React?

A. To share reusable logic across components

B. To replace components with hooks

C. To manipulate the DOM directly

D. To render components more quickly

5. Which statement best describes a React component?

A. A small, reusable chunk of code that renders HTML

B. An event handler function

C. A CSS stylesheet

D. A DOM node

SAMPLE

6. How do you pass a prop to a component in JSX?

A. By giving an attribute to the component instance

B. By mutating the component's state

C. By calling the prop as a function inside render

D. By using a global variable

7. What does the dependencies section in package.json contain?

A. Contains the scripts used to run the app

B. Lists public assets to be served

C. Contains all the required node modules and versions

D. Specifies alias commands for react-scripts

8. A component 'mounts' when ...

A. A Component Renders For The First Time

B. A Component Updates

C. A Component Unmounts

D. A Component Is Created But Not Rendered

9. Mounting lifecycle events only execute ...

A. The First Time That A Component Renders

B. During Every Render

C. Just Before Unmount

D. After Updates

10. Which command generates a boilerplate version of a React application?

A. npm init react-app

B. npm install -g create-react-app

C. npx create-react-app

D. yarn create react-app

SAMPLE

Answers

SAMPLE

1. A
2. A
3. A
4. A
5. A
6. A
7. C
8. A
9. A
10. D

SAMPLE

Explanations

SAMPLE

1. JSX element

A. A basic unit of JSX found in a JavaScript file which is treated as, and has the functionality of, a JavaScript expression.

B. A React component syntax for class definitions

C. A way to define CSS styles inline

D. A runtime function to access DOM elements

A JSX element is the basic unit you write in JSX inside a JavaScript file and is treated as a JavaScript expression. When you write something like Hello or <MyComponent prop="value" />, you're describing what React should render. JSX is compiled to React.createElement calls, which produce React element objects that React uses to build the UI. JSX elements can include props and children and can be nested or stored in variables just like any other expression. They're not class definitions for components, not inline CSS definitions, and not runtime functions for accessing the DOM. This makes the described option the correct one, since it captures that a JSX element is a basic, expression-like unit in JSX.

2. Which statement correctly describes how to access the props object inside a React class component?

A. Use this.props.

B. Use this.state.

C. Use props API directly

D. Use this.context.

Props passed to a class component are accessed via this.props. In a class component, React attaches the external data from the parent to the component instance, and you read it through this.props in render and other methods. You can destructure for convenience, for example: const { title, onSubmit } = this.props. Remember, props come from outside and shouldn't be mutated; state (this.state) is for internal, changing data. There isn't a separate "props API" to call directly, and this.context is for context values, not the general props you receive from a parent. So this.props is the correct way to access the props inside a class component.

3. You cannot call this.setState from the body of this update lifecycle method.

A. ComponentWillUpdate

B. ComponentWillMount

C. Render

D. ComponentDidUpdate

In the update path, React is about to re-render with new props or state. Calling this.setState inside the pre-render update hook would schedule another update while React is still handling the current one, which can trigger extra renders and even cause an unstable loop. Because of that, you should not set state from this update-stage method. If you need to react to prop changes, do so in safer places: derive values in a constructor or a static getDerivedStateFromProps (for modern patterns), or perform related actions after the update in componentDidUpdate with proper guards. Note that componentWillUpdate itself is deprecated in newer React versions, with these safer patterns taking its place.

SAMPLE

4. What is the primary purpose of a custom hook in React?

A. To share reusable logic across components

B. To replace components with hooks

C. To manipulate the DOM directly

D. To render components more quickly

Custom hooks are a way to share stateful logic across multiple components. They let you take pieces of behavior that involve React features like state or effects and put them into a single function so several components can reuse that logic without repeating code. A custom hook is simply a function whose name starts with use and that can call other hooks inside it, returning values or helpers that components can consume. This approach keeps components focused on rendering while the shared behavior lives in one place, making code easier to read, test, and maintain. It's not about replacing components with hooks, and it doesn't by itself make rendering faster or replace DOM manipulation—any DOM work still goes through the usual hooks like useEffect or refs. For example, a hook that handles data fetching, loading state, and error handling can be reused by any component that needs to fetch data, illustrating how custom hooks promote reuse and clean composition.

5. Which statement best describes a React component?

A. A small, reusable chunk of code that renders HTML

B. An event handler function

C. A CSS stylesheet

D. A DOM node

React components are the building blocks of a UI. They're small, reusable pieces of code that describe a part of the interface and render HTML (via JSX) for given inputs. A component can be a function or a class, and it can accept inputs called props and manage internal state to control what it renders. By composing multiple components, you build up complex interfaces from simple, isolated pieces. That's why the statement that best describes a React component is the one that frames it as a small, reusable chunk of code that renders HTML. An event handler is just a function responding to actions, not the component itself. A CSS stylesheet provides styling, not UI structure. A DOM node is the actual element created in the browser, not the reusable piece of code that defines how UI should look.

6. How do you pass a prop to a component in JSX?

A. By giving an attribute to the component instance

B. By mutating the component's state

C. By calling the prop as a function inside render

D. By using a global variable

Props are passed to a component in JSX by adding attributes to the component element. Those attributes become the props that the component receives. For example, rendering <Greeting name="Alice" /> passes a prop named name with the value "Alice" to Greeting, and inside Greeting you can access it as props.name (or by destructuring to { name }). You can pass any valid expression, like <Greeting name={userName} /> or <Greeting active={isActive} />. Props are read-only from the child's perspective; if you need to change data, you typically use state or callbacks passed down from the parent. Mutating the component's state is about internal data, not how you pass data to a child. Calling a prop as a function inside render is a pattern for using function props, not the act of passing a prop itself. Using a global variable could be used in the value, but the standard mechanism is still through attributes on the JSX element.

SAMPLE

7. What does the dependencies section in package.json contain?

A. Contains the scripts used to run the app

B. Lists public assets to be served

C. Contains all the required node modules and versions

D. Specifies alias commands for react-scripts

The dependencies section lists the libraries your app needs to run, along with the version ranges for those libraries. That means it specifies the required node modules and their versions, so when you install, tools like npm or yarn fetch those packages into node_modules to make the app work in production or development. This is why the best answer is that it contains all the required node modules and versions. Context helps: there's also a devDependencies section for packages used only during development (like testing or build tools), which is why not everything you install belongs under dependencies. The other options don't fit because scripts for running the app live in a separate scripts section, public assets aren't tracked in package.json, and alias commands for react-scripts are not stored in dependencies.

8. A component 'mounts' when ...

A. A Component Renders For The First Time

B. A Component Updates

C. A Component Unmounts

D. A Component Is Created But Not Rendered

Mounting is the initial insertion of a component into the DOM, which happens when it renders for the first time. So the component mounts on its first render when it becomes part of the UI, and then you'll typically see lifecycle events like componentDidMount afterward. After mounting, further changes to state or props cause re-renders that are described as updates, not mounts. If a component is created but never rendered, it never gets into the DOM, so it isn't mounting.

9. Mounting lifecycle events only execute ...

A. The First Time That A Component Renders

B. During Every Render

C. Just Before Unmount

D. After Updates

Mounting lifecycle events happen only when a component is added to the DOM for the first time. This occurs after the component is created and renders for the first time, marking the initial mount (for class components, methods like componentDidMount run after that first render). They don't run on subsequent renders, which are handled by the updating phase (componentDidUpdate) or by unmounting (componentWillUnmount) when the component is removed. So the best description is that these events occur during the initial render.

SAMPLE

10. Which command generates a boilerplate version of a React application?

A. npm init react-app

B. npm install -g create-react-app

C. npx create-react-app

D. yarn create react-app

Bootstrapping a React project means quickly generating the standard project structure, scripts, and configuration so you can start building right away. Using Yarn's create flow runs the create-react-app package to scaffold a new app, producing the boilerplate files and folders you expect (like src, public, a default package.json with start/build/test scripts, and a working React setup). It doesn't require installing a global CLI; Yarn handles downloading and executing the create-react-app scaffolder and then creates the project folder you name. This approach is convenient and aligns with Yarn's way of launching scaffolding tools. Other options either rely on a global installation, use nonstandard commands, or involve alternative tooling. The npx form is also a valid way to generate the boilerplate, but the given command exemplifies the Yarn create workflow. To use it, you'd run something like: yarn create react-app my-app.

SAMPLE

Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we'd love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://reactjs.examzify.com

We wish you the very best on your exam journey. You've got this!

SAMPLE