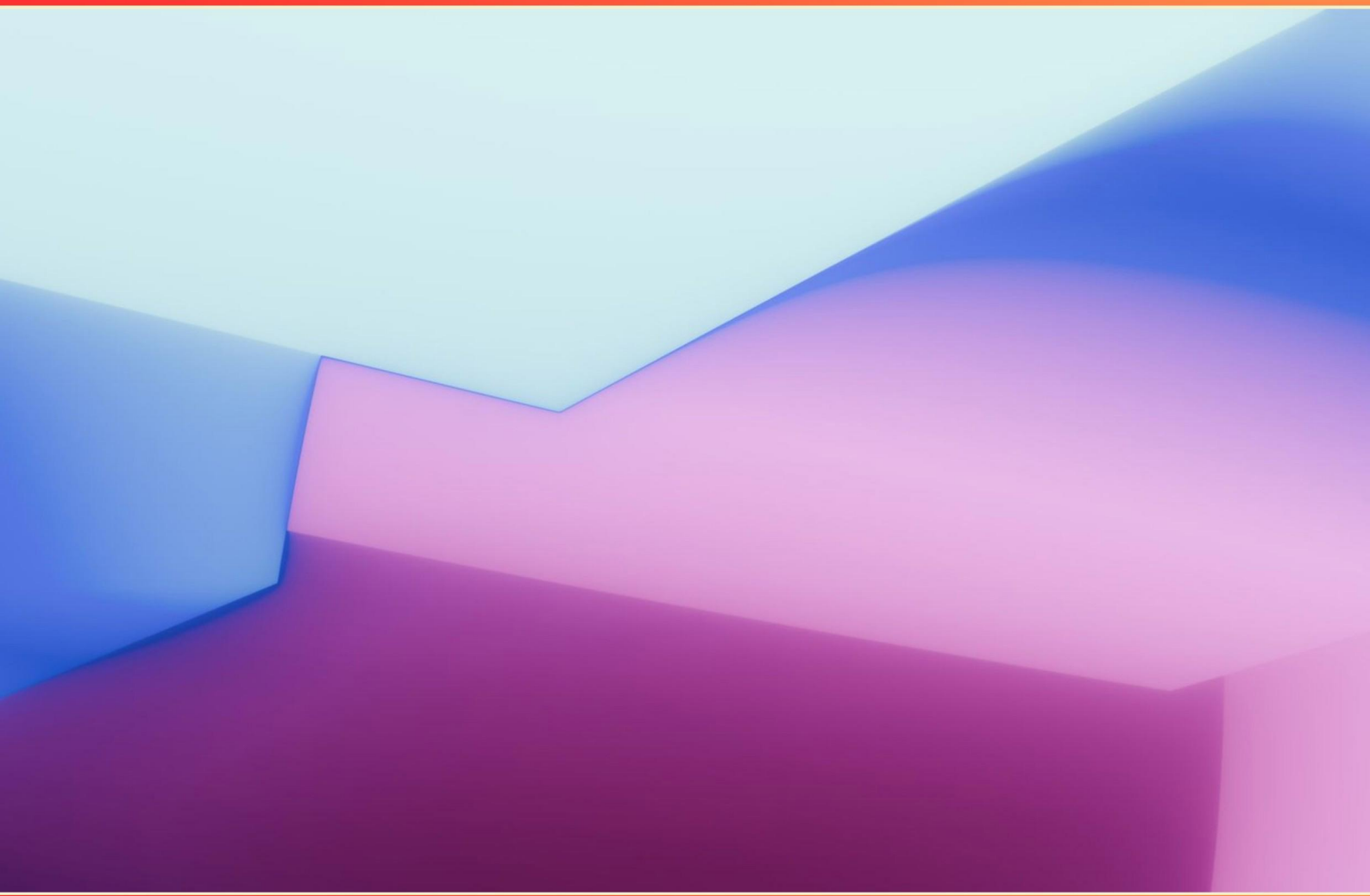
SALESFORCE JAVASCRIPT DEVELOPER I CERTIFICATION PRACTICE EXAM (SAMPLE)

STUDY GUIDE



SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://salesforcejsdev1.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

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. Which of the following correctly lists basic DOM data types?

A. Document, node, list, element

B. Document, node, element, nodeList, attribute, namedNodeMap

C. Element, variable, attribute

D. List, map, set, document

2. In which scenario is it preferable to use `let` instead of `var`?

A. When the variables are needed globally

B. When you need block scope for your variables

C. When you are declaring constants

D. When working with variable hoisting issues

3. Which option describes a situation when the npm install command would succeed but issue a warning?

A. The installed package has no dependencies

B. A version of the package is not compatible

C. Package has unmet peer dependencies

D. The package does not exist in the npm registry

4. What is the primary tool for debugging LWC code?

A. Salesforce Developer Console

B. Browser's developer tools

C. Visual Studio Code

D. Salesforce CLI

5. What is the correct syntax for defining a variable in JavaScript?

A. var myVariable;

B. define myVariable;

C. set myVariable;

D. let myVariable;

SAMPLE

6. What does `this.template.querySelector()` allow a developer to do in Lightning Web Components (LWC)?

A. Access DOM elements within the component's template

B. Define a class method for another component

C. Retrieve data from a server asynchronously

D. Trigger events without user action

7. What can you use to communicate between parent and child components in LWC?

A. Global event listeners

B. Custom events

C. Static resources

D. Shared variables

8. In JavaScript, what will happen if you try to access a variable declared with let before it is initialized?

A. It returns a ReferenceError.

B. It returns undefined.

C. It returns null.

D. It runs successfully.

9. What will the console output be when matching against the regex /[A-Z]/g in the string 'The quick brown fox jumps over the lazy dog. It barked.'?

A. ["T","I"]

B. ["T"]

C. ["I"]

D. No match found

10. What does the return value of a function declared with the function keyword default to if not specified?

A. 0

B. undefined

C. NaN

D. empty string

SAMPLE

Answers

SAMPLE

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

SAMPLE

Explanations

SAMPLE

1. Which of the following correctly lists basic DOM data types?

A. Document, node, list, element

B. Document, node, element, nodeList, attribute, namedNodeMap

C. Element, variable, attribute

D. List, map, set, document

The correct answer includes a comprehensive list of the basic DOM data types that are integral to working with the Document Object Model in web development. The Document represents the entire page and serves as the entry point to the DOM structure. The node is a fundamental unit within the DOM, as all elements, attributes, text, and other objects within the structure are classified as nodes. The element is a specific type of node that corresponds to HTML or XML elements. The nodeList is a collection of nodes that can include various types of nodes from the DOM. The attribute represents the properties of elements, while the namedNodeMap allows you to access all the attributes of a specified element. This answer encapsulates all the essential data structures you would typically interact with in the DOM, covering both the types of nodes and their characteristics. Understanding these types is crucial for any practical application involving JavaScript and the DOM, such as manipulating web pages dynamically. The other provided options do not comprehensively capture the fundamental elements and structures of the DOM. For example, while some might mention a few correct terms, they lack the depth by omitting key data types essential for a complete understanding of DOM interactions.

2. In which scenario is it preferable to use `let` instead of `var`?

A. When the variables are needed globally

B. When you need block scope for your variables

C. When you are declaring constants

D. When working with variable hoisting issues

Using `let` is preferable when you need block scope for your variables. In JavaScript, `var` declares a variable that is function-scoped or globally-scoped, depending on where it is declared. This can lead to unintended behavior, especially in situations like loops or conditional statements, where you might want a variable to maintain its value only within a specific block of code. For example, if you declare a loop variable using `var`, it will be accessible outside the loop after the loop has completed. In contrast, if you use `let`, the variable will be confined to the block in which it was declared, preventing access from outside that block. This allows for better control over variable lifetimes and helps prevent bugs associated with variable scope. Using `let` enhances code readability and maintainability by making it clear which parts of your code can interact with the variable. This is particularly useful in scenarios like closures or when dealing with asynchronous code, as it helps avoid unintended references to a variable that has changed during execution.

SAMPLE

3. Which option describes a situation when the npm install command would succeed but issue a warning?

A. The installed package has no dependencies

B. A version of the package is not compatible

C. Package has unmet peer dependencies

D. The package does not exist in the npm registry

When the npm install command is executed, it can sometimes succeed while still issuing a warning. This commonly occurs in situations where a version of the package being installed is not fully compatible with the version of another package it depends on. In this context, compatibility issues can arise when a package depends on specific versions of other packages or has required features that are not present in the versions available in the current environment. npm will proceed with the installation of the package requested but will also notify the user of any compatibility concerns via a warning. This allows developers to be aware of potential issues that may arise during runtime without halting the installation process entirely. Understanding these nuances is essential for maintaining a healthy dependency tree in an application, as ignoring such warnings may lead to unexpected behavior in the application later on.

4. What is the primary tool for debugging LWC code?

A. Salesforce Developer Console

B. Browser's developer tools

C. Visual Studio Code

D. Salesforce CLI

The primary tool for debugging Lightning Web Components (LWC) code is the browser's developer tools. These tools provide developers with a robust set of utilities to inspect, modify, and debug HTML, CSS, and JavaScript within the web application environment. Utilizing the console for logging information, setting breakpoints, and watching expressions allows developers to trace the flow of their LWC code in real-time as it interacts with the Salesforce platform. Additionally, the Elements panel enables inspection of the document structure and styles, which is essential for diagnosing layout issues or verifying the proper rendering of components. Other tools, while useful in the broader context of Salesforce development, do not cater specifically to the immediate real-time debugging needs of LWC. For instance, the Salesforce Developer Console is a powerful tool for inspecting and running Salesforce Apex code, but it lacks the real-time interaction capabilities needed for LWC. Visual Studio Code is an excellent IDE for writing and editing code but does not offer the integrated, in-browser debugging features that the developer tools provide. Salesforce CLI plays a critical role in managing Salesforce projects and deployments but does not have direct debugging capabilities for LWC in a browser environment. Overall, leveraging the browser's developer tools is essential for effectively diagnosing and resolving issues within

SAMPLE

5. What is the correct syntax for defining a variable in JavaScript?

A. var myVariable;

B. define myVariable;

C. set myVariable;

D. let myVariable;

The correct syntax for defining a variable in JavaScript can be achieved using several keywords, including 'var' and 'let'. Using 'var' is a traditional way of declaring a variable. When you use `var myVariable;`, it creates a variable named `myVariable` that can be assigned a value later. This syntax is straightforward and has been part of JavaScript since its inception. Additionally, using `let myVariable;` is also valid and represents a more modern approach to variable declaration. 'let' allows for block-scoped variables, giving developers flexibility with variable scope. While both 'var' and 'let' are correct for variable declaration in their respective contexts, the more traditional syntax highlighted in the question reflects the foundational way of defining variables in JavaScript. The other options do not conform to JavaScript syntax for variable declaration. 'define myVariable;' and 'set myVariable;' are not recognized keywords in JavaScript, making them invalid. Hence, the choice that communicates the correct syntax must align with established JavaScript language rules.

6. What does `this.template.querySelector()` allow a developer to do in Lightning Web Components (LWC)?

A. Access DOM elements within the component's template

B. Define a class method for another component

C. Retrieve data from a server asynchronously

D. Trigger events without user action

The method `this.template.querySelector()` in Lightning Web Components (LWC) is specifically used to access DOM elements that are defined within the component's template. When a developer uses this method, they can select a single element from the template using a CSS selector. This is particularly useful for interacting with elements, applying styles, manipulating attributes, or retrieving values from input fields directly within the component. The significance of accessing DOM elements lies in the ability to enhance user interactivity and dynamically control various aspects of the user interface based on component logic. For instance, if a developer needs to focus on an input field when a certain event occurs, they can use `this.template.querySelector()` to accurately reference that field and call methods on it, such as `.focus()`. Other options such as defining class methods for other components, retrieving data from a server asynchronously, or triggering events without user action pertain to different functionalities and are not applicable to the context of accessing DOM elements within the LWC structure. Therefore, the answer emphasizes the direct relationship between the method and its purpose of DOM manipulation within the component's view.

SAMPLE

7. What can you use to communicate between parent and child components in LWC?

A. Global event listeners

B. Custom events

C. Static resources

D. Shared variables

In Lightning Web Components (LWC), custom events are the primary mechanism for communication between parent and child components. When a child component needs to notify its parent about an event or a change, it can dispatch a custom event. This event can carry information (using event detail) that the parent can handle. The parent component can then listen for this custom event and respond accordingly by defining an event handler function. Using custom events allows for clear communication, ensuring that child components remain decoupled from their parents, which enhances reusability and maintainability. This approach aligns with the component-based architecture of LWC, where components interact through well-defined interfaces rather than direct references. While global event listeners and shared variables might be used for component communication, they are not typically leveraged for parent-child relationships within the scope of LWC. Static resources do not apply to this context as they are used for storing files and assets shared across components, not for event communication.

8. In JavaScript, what will happen if you try to access a variable declared with let before it is initialized?

A. It returns a ReferenceError.

B. It returns undefined.

C. It returns null.

D. It runs successfully.

When a variable is declared using the `let` keyword in JavaScript, it is hoisted to the top of its block scope. However, unlike variables declared with `var`, which are initialized to `undefined`, `let` declarations are not initialized until the actual statement of the variable is executed. Therefore, if you attempt to access a `let` variable before its declaration and initialization, JavaScript throws a `ReferenceError`. This error signifies that you are trying to access a variable that is in the "temporal dead zone" — the time period between entering the block scope and the `let` declaration being encountered. This behavior is an important aspect of block scoping in JavaScript, helping to avoid potential bugs that could arise from undefined variables. In contrast, trying to access an undefined variable declared with `var` would simply return `undefined`, which is different from the strict limitation imposed by `let`. As such, understanding how hoisting and scoping work in JavaScript is crucial for writing error-free code.

SAMPLE

9. What will the console output be when matching against the regex /[A-Z]/g in the string 'The quick brown fox jumps over the lazy dog. It barked.'?

A. ["T","I"]

B. ["T"]

C. ["I"]

D. No match found

When using the regular expression /[A-Z]/g to match against the string 'The quick brown fox jumps over the lazy dog. It barked.', the regex is designed to find all uppercase letters in the string. The character class [A-Z] matches any single uppercase letter from A to Z. The global flag "g" ensures that the regex searches through the entire string for all occurrences of uppercase letters, not just the first one. In the provided string, the uppercase letters are 'T' from "The" and 'I' from "It." As the regex scans the string, it identifies both of these uppercase letters. Because the global flag is applied, both matches are collected and returned as an array. Therefore, the output will be an array containing both matched uppercase letters: ["T", "I"]. This demonstrates how regex searches can be effectively utilized to extract specific patterns from strings, and the global flag allows for a comprehensive search throughout the entire text.

10. What does the return value of a function declared with the function keyword default to if not specified?

A. 0

B. undefined

C. NaN

D. empty string

When a function in JavaScript is declared using the function keyword and does not explicitly return a value, the return value defaults to undefined. This means that if the function executes without encountering a return statement, the JavaScript engine will automatically return undefined for that function call. This behavior is fundamental to understanding how functions operate in JavaScript, as it indicates that a function may complete its operations without giving back a specific result. The idea of an unspecified return value being undefined allows developers to utilize functions that perform actions (like logging or modifying data) without needing to provide a return value. Thus, if you were to call a function that lacks a return statement, you would receive undefined, which is a distinct and crucial aspect of JavaScript's handling of functions. This contrasts with other potential options, which represent specific values that do not occur as default return values in this context.

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://salesforcejsdev1.examzify.com

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

SAMPLE