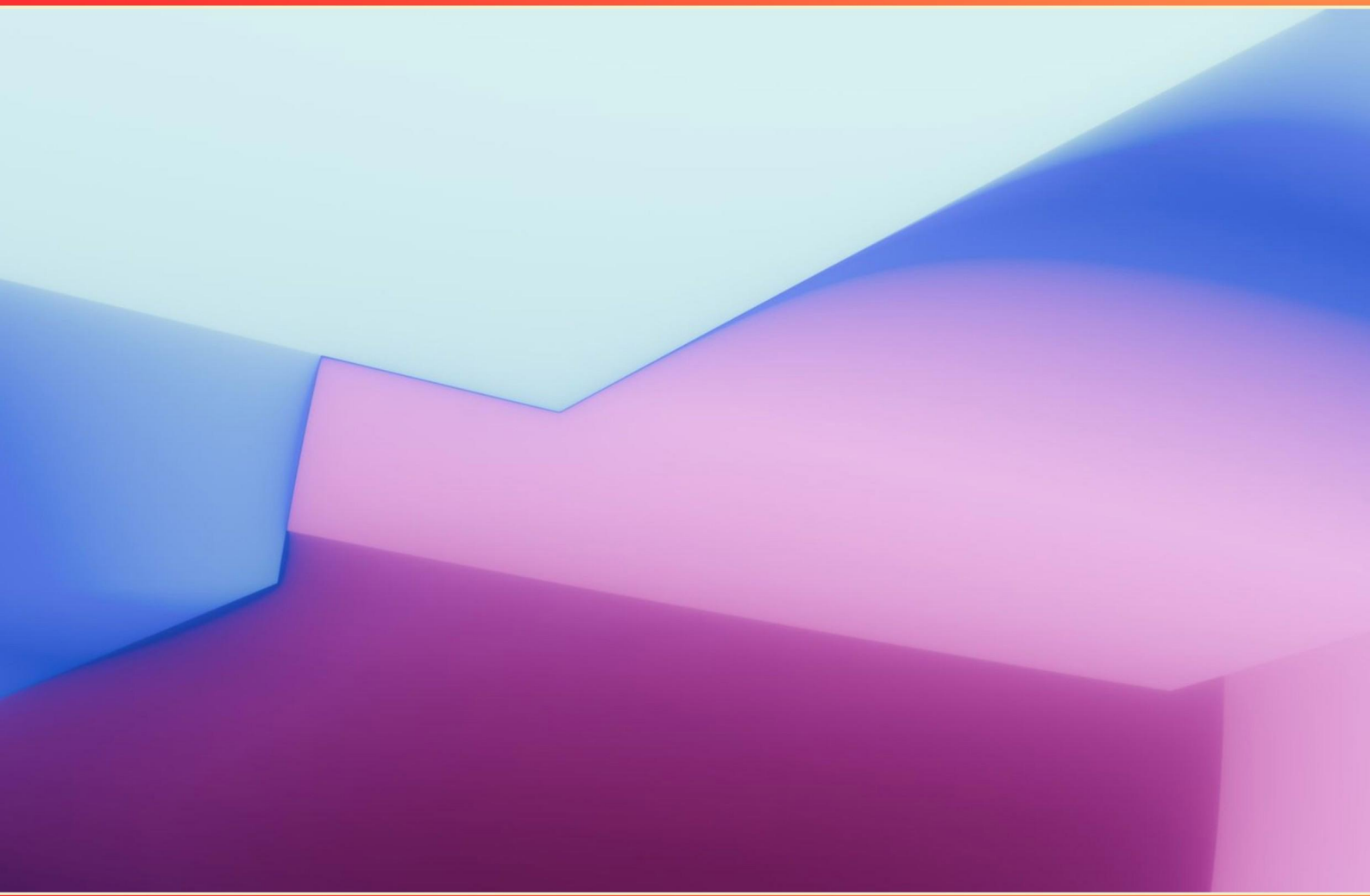
SALESFORCE JAVASCRIPT DEVELOPER I CERTIFICATION PRACTICE EXAM (SAMPLE)

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

Copyright 1

Table of Contents 2

Introduction 3

How to Use This Guide 4

Questions 5

Answers 8

Explanations 10

Next Steps 16

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. What is the output of the following code: [1, 2, 3, 4].reduce((x, y) => console.log(x, y));?

A. 1 2 and 3 3 and 6 4

B. 1 2 and 2 3 and 3 4

C. 1 undefined and 2 undefined and 3 undefined and 4 undefined

D. 1 2 and undefined 3 and undefined 4

2. What is a key characteristic of JavaScript's asynchronous programming model?

A. All code is executed in a sequence without interruption

B. Code execution can be paused until a condition is met

C. Functions are executed in a single-threaded environment

D. Code execution can continue while waiting for external events

3. What distinguishes white-box testing from black-box testing?

A. White-box testing ignores internal code structure.

B. Black-box testing is unaware of the internal code.

C. Only white-box testing uses user experience as feedback.

D. White-box testing knows the internal code being tested.

4. What is the primary difference between for ... of and for ... in when iterating over an array?

A. for ... of iterates via index while for ... in uses elements

B. for ... of iterates via elements while for ... in uses index

C. Both iterate through elements

D. Both iterate through index

5. What dictates that a function is a pure function?

A. It depends on external state or global variables

B. It always changes an input's state

C. It always returns the same output for the same input and does not cause side effects

D. It performs operations like logging or alerting

SAMPLE

6. What will be logged for console.log(Kris.age) when the Person prototype is changed?

A. 18

B. 29

C. undefined

D. TypeError

7. What does the await keyword do in JavaScript?

A. Executes promises in parallel

B. Halts execution until the promise is settled

C. Only runs inside non-async functions

D. Defines a new promise

8. What must you do to integrate LWC with other Salesforce components?

A. Ensure they are in the same namespace

B. Arrange them in adjacent files

C. Ensure they share a common parent or use events

D. Utilize Apex controllers for communication

9. When the innerDiv is clicked, in what order will the event listeners be called according to the provided code?

A. displayInnerMessage, displayOuterMessage

B. displayInnerMessage only

C. displayOuterMessage only

D. displayOuterMessage, displayInnerMessage

10. When defining properties within a constructor, what is being created?

A. A new global variable

B. An instance of an object

C. A method on the constructor

D. A prototype

SAMPLE

Answers

SAMPLE

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

SAMPLE

Explanations

SAMPLE

1. What is the output of the following code: [1, 2, 3, 4].reduce((x, y) => console.log(x, y));?

A. 1 2 and 3 3 and 6 4

B. 1 2 and 2 3 and 3 4

C. 1 undefined and 2 undefined and 3 undefined and 4 undefined

D. 1 2 and undefined 3 and undefined 4

The output of the given code results from how the `reduce` method works in JavaScript. The `reduce` method iterates over the array and applies the provided callback function, which in this case is logging the values of `x` and `y`. When using `reduce`, the first parameter, `x`, represents the accumulator, which initially is set to the first element of the array if no initial value is provided. The second parameter, `y`, represents the current value being processed in the array. In the provided code snippet: - When the first element is processed (1), there's no prior accumulator value, so `x` will be assigned `1`, and `y` will be `2`. The console will output `1 2`. - For the next iteration, `x` retains the first value (the return value of the console.log from the previous iteration is `undefined` as the console.log does not return a value), and `y` will now be `3`. Therefore, `x` is `undefined` here, and the console outputs `undefined 3`. - This process continues with `x` remaining `undefined` and `y` taking the next array values. Thus, when `y`

2. What is a key characteristic of JavaScript's asynchronous programming model?

A. All code is executed in a sequence without interruption

B. Code execution can be paused until a condition is met

C. Functions are executed in a single-threaded environment

D. Code execution can continue while waiting for external events

A key characteristic of JavaScript's asynchronous programming model is that code execution can continue while waiting for external events. This non-blocking behavior allows JavaScript to handle tasks like network requests, timers, and user interactions without halting the execution of the entire program. When an asynchronous operation is initiated, JavaScript does not wait for that operation to complete before moving on to execute the next line of code. Instead, it uses callbacks, promises, or async/await syntax to manage the result of the asynchronous operation once it has finished. This makes JavaScript highly efficient for handling tasks that may take an unpredictable amount of time, such as fetching data from servers, which enhances the user's experience by keeping the application responsive. The other options do not accurately describe this characteristic: the sequential execution of all code would not support the asynchronous model, pausing execution based on conditions describes a synchronous style, and a single-threaded environment refers more to how JavaScript operates in terms of threads rather than its asynchronous capabilities.

SAMPLE

3. What distinguishes white-box testing from black-box testing?

A. White-box testing ignores internal code structure.

B. Black-box testing is unaware of the internal code.

C. Only white-box testing uses user experience as feedback.

D. White-box testing knows the internal code being tested.

The distinction between white-box testing and black-box testing primarily revolves around the tester's awareness of the internal workings of the software being tested. White-box testing involves a comprehensive understanding of the internal code structure, algorithms, and logic. This knowledge enables the tester to design test cases that specifically target paths through the code, ensuring that all branches and possible inputs are examined. In contrast, black-box testing is characterized by a lack of insight into the internal code; instead, it focuses solely on the outputs generated in response to various inputs. Testers operate with the assumption that they do not need to know how the system works internally to validate its functionality. Therefore, recognizing that white-box testing entails knowledge of the underlying code and how it operates confirms that the statement about it knowing the internal code being tested is accurate. This internal perspective is essential for debugging, optimizing performance, and improving overall code quality.

4. What is the primary difference between for ... of and for ... in when iterating over an array?

A. for ... of iterates via index while for ... in uses elements

B. for ... of iterates via elements while for ... in uses index

C. Both iterate through elements

D. Both iterate through index

The distinction lies in how each loop interacts with the structure of an array. The for...of loop is designed to iterate directly over the values (elements) in an iterable object, such as an array. This means that when you implement a for...of loop on an array, you directly access the actual values contained within that array, making it straightforward to work with the data itself. In contrast, the for...in loop is intended to iterate over the enumerable properties of an object, which includes the string keys of properties for arrays. When you use for...in on an array, it produces the array indexes (which are string representations of numbers) rather than the elements themselves. This can lead to unintended behaviors because for...in does not guarantee the order of the iteration, and it is better suited for objects rather than for iterating arrays where the focus is on the values. Therefore, the primary difference is that for...of focuses on retrieving the elements (the actual values) of the array, while for...in focuses on the indexes, leading to the conclusion that for...of iterates via elements while for...in uses indexes.

SAMPLE

5. What dictates that a function is a pure function?

A. It depends on external state or global variables

B. It always changes an input's state

C. It always returns the same output for the same input and does not cause side effects

D. It performs operations like logging or alerting

A pure function is defined by two key characteristics: it consistently produces the same output when given the same input and it does not have any side effects. This means that a pure function's behavior is entirely determined by its input parameters, and it does not rely on or alter any external states, such as variables outside its scope or the global state of the application. By always returning the same output for the same input, pure functions benefit from predictability, making them easier to test and reason about, as their execution does not depend on any external factors. Additionally, the absence of side effects means that calling a pure function does not affect the environment or the state of the application in unexpected ways, leading to safer and more reliable code. The other options describe scenarios that contradict the definition of a pure function: dependence on external state, changing an input's state, or performing operations like logging or alerting introduce variability or side effects, which are precisely what characterizes impure functions. Thus, the mention of returning the same output for the same input and having no side effects is what solidifies the understanding of a pure function in programming.

6. What will be logged for console.log(Kris.age) when the Person prototype is changed?

A. 18

B. 29

C. undefined

D. TypeError

When considering the console.log output for Kris.age after the Person prototype is changed, it is important to understand how prototype inheritance and property lookup work in JavaScript. If Kris is an instance of the Person constructor, this instance would naturally inherit properties from the Person prototype unless those properties are defined directly on the instance. If, at some point, the age property of the Person prototype is changed, then any instance relying on that prototype for its properties will reflect this change if they do not have their own age property defined. For example, if the Person prototype's age property was originally set to a specific value (for instance, 18) and then updated to a new value (like 29), if Kris does not have its own age property defined, then calling console.log(Kris.age) after the change will yield the updated value from the prototype, which in this case would be 29. However, if the option selected is 18, it implies that there might be a case where Kris has its own age property set to 18 or simply pointing toward that prototype directly before any changes were made. The console output would reflect this directly on Kris, rather than the prototype's property. In scenarios where the instance has its own 'age' property

SAMPLE

7. What does the await keyword do in JavaScript?

A. Executes promises in parallel

B. Halts execution until the promise is settled

C. Only runs inside non-async functions

D. Defines a new promise

The await keyword is a fundamental part of asynchronous programming in JavaScript when used in conjunction with the async keyword. It effectively pauses the execution of an asynchronous function until a Promise is resolved or rejected. This means that the line of code containing the await expression will not proceed until the Promise has been settled, allowing for a more synchronous-like flow of code, which can enhance readability and maintainability. When you utilize await, it ensures that any subsequent code that relies on the results of the Promise will operate only after the Promise has settled. This behavior allows developers to write cleaner and more comprehensible code, avoiding the callback hell that often accompanies traditional promise handling using then() and catch() methods. The other options do not accurately capture the functionality of the await keyword. For instance, it does not execute promises in parallel; instead, it makes the code wait for a promise to resolve. It is also important to note that the await keyword can only be used inside functions that are declared with the async keyword, which differentiates it from some incorrectly suggested functions. Lastly, await does not define a new promise; it merely acts upon an existing promise.

8. What must you do to integrate LWC with other Salesforce components?

A. Ensure they are in the same namespace

B. Arrange them in adjacent files

C. Ensure they share a common parent or use events

D. Utilize Apex controllers for communication

To effectively integrate Lightning Web Components (LWC) with other Salesforce components, ensuring that they share a common parent or use events is essential. This is because the communication between components in the Salesforce ecosystem often relies on the component hierarchy and the event-driven architecture that Salesforce employs. When components share a common parent, this allows them to communicate directly through properties and method calls. The parent component can pass data down to child components using properties, and child components can notify the parent of changes or events, enabling a seamless flow of information. Additionally, when components do not share a common parent, using custom events facilitates communication. Events can be dispatched from child components and listened to by parent components, allowing actions in one component to invoke responses in others, irrespective of their actual placement in the component hierarchy. This methodology is foundational in maintaining a clean and efficient communication channel between various components, promoting both modular design and reusability across the application. Implementing this approach ensures that components interact appropriately and that the overall design adheres to the best practices laid out for developing applications in the Salesforce ecosystem.

SAMPLE

9. When the innerDiv is clicked, in what order will the event listeners be called according to the provided code?

A. displayInnerMessage, displayOuterMessage

B. displayInnerMessage only

C. displayOuterMessage only

D. displayOuterMessage, displayInnerMessage

In the context of event propagation in web development, specifically with regard to event bubbling, when an event occurs on an element, it first triggers the event listeners on that element, and then the event bubbles up to its parent elements, triggering their event listeners in the order they appear in the DOM hierarchy. The specified answer indicates that when the inner `div` is clicked, the event listeners will call `displayOuterMessage` first, followed by `displayInnerMessage`. This suggests that the event listener for the parent element (outer `div`) executes before the listener for the child element (inner `div`). This scenario aligns with the concept of event bubbling, which happens when an event is dispatched from a child element and then propagates upwards to its parent elements. Therefore, if there's a click listener attached to both the inner and outer `div`, the listener associated with the outer `div` is executed before the one on the inner `div`. Consequently, the order of execution would be `displayOuterMessage` followed by `displayInnerMessage`. This phenomenon occurs because, in typical event scenarios, the event does not halt at the target element; it continues up the DOM tree unless stopped explicitly by methods like `stopPropagation()` or if the event

10. When defining properties within a constructor, what is being created?

A. A new global variable

B. An instance of an object

C. A method on the constructor

D. A prototype

Defining properties within a constructor is a fundamental aspect of object-oriented programming in JavaScript. When properties are defined inside a constructor function, they are typically attached to the instance of the object that is being created using the `new` keyword. This means that each time the constructor is invoked to create a new object, a separate instance containing its own set of properties is created. By using the `this` keyword within the constructor, you can establish properties that are specific to that instance. For example, if you were to create a `Person` constructor and define `this.name` and `this.age`, every time you instantiate a `Person` object, those properties belong specifically to that object instance, allowing for the creation of unique entities. This encapsulation of data within instances is a core principle of object-oriented design, promoting modularity and reusability of code. In contrast, declaring a variable globally or associating a method to the constructor would not provide the same isolation for the instantiated objects, and defining a prototype would refer to shared properties and methods instead of instance-specific characteristics. Thus, the correct answer highlights the creation of an instance of an object, which is essential for managing object-specific data in JavaScript.

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