Salesforce JavaScript Developer Practice Exam (Sample)

Study Guide

Everything you need from our exam experts!

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**

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.

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!

Questions

1. What is the outcome of attempting to change the dept property of returnTarget after calling Object.freeze(returnTarget)?

A. The dept property will change to "Finance"

B. The dept property remains unchanged

C. A new property can be added

D. The entire object can be deleted

2. What does JSON.parse do?

A. Convert JavaScript objects into JSON strings

B. Parse a JSON string and convert it to a JavaScript object

C. Validate JSON syntax only

D. Store JSON data across sessions

3. Which are the primary components of the Salesforce Lightning Framework?

A. Visualforce pages and Apex controllers

B. Aura components and Lightning Web Components (LWC)

C. JavaScript modules and CSS stylesheets

D. HTML templates and server-side scripts

4. Define a closure in JavaScript.

A. A closure is a block of code that defines an error

B. A closure is a property of an object

C. A closure is a function that retains access to its lexical scope even when the function is executed outside that scope

D. A closure is a variable that cannot be modified

5. What will be the output of the following code: greetign = {};?

A. {}

B. ReferenceError: greetign is not defined

C. undefined

D. SyntaxError

6. What is the output of the following code snippet when logged to the console? let num = 10; const increaseNumber = () => num++; const increasePassedNumber = number => number++; const num1 = increaseNumber(); const num2 = increasePassedNumber(num1); console.log(num1); console.log(num2);

A. 10, 10

B. 10, 11

C. 11, 11

D. 11, 12

7. What is the role of the wire service in LWC?

A. To manage server-side logic

B. To provide a way to read data from an Apex method

C. To enhance the styling of components

D. To handle user input

8. In LWC, how do you handle events bubbling?

A. By stopping the event flow entirely

B. By using the event.stopPropagation() method

C. By cancelling the event listener

D. By allowing the event to proceed naturally

9. What will the output of the following code be: let array1 = ['one', 'two']; let array2 = ['three', 'four']; array1.push(...array2); console.log(...array1);?

A. one two

B. one two three four

C. [one, two, three, four]

D. ["one", "two", "three", "four"]

10. What is the role of the wire service in LWC?

A. To handle all events within the component

B. To automatically retrieve data from Salesforce and keep component data in sync

C. To manage the rendering of HTML elements

D. To facilitate API requests to external services

Answers

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

Explanations

1. What is the outcome of attempting to change the dept property of returnTarget after calling Object.freeze(returnTarget)?

A. The dept property will change to "Finance"

B. The dept property remains unchanged

C. A new property can be added

D. The entire object can be deleted

When you call `Object.freeze(returnTarget)` on an object in JavaScript, you make that object immutable. This means that its properties cannot be changed, added to, or removed. The `dept` property of `returnTarget` will remain unchanged because freezing the object prevents any alterations. This immutability applies to all properties of the object. When you try to assign a new value, such as "Finance," to the `dept` property after the object has been frozen, JavaScript will silently ignore the change in non-strict mode or throw a `TypeError` in strict mode, thus ensuring that the original state of the object is preserved. In summary, freezing an object guarantees that its properties stay intact, leading to the conclusion that the `dept` property remains unchanged after the object has been frozen.

2. What does JSON.parse do?

A. Convert JavaScript objects into JSON strings

B. Parse a JSON string and convert it to a JavaScript object

C. Validate JSON syntax only

D. Store JSON data across sessions

JSON.parse is a method used to convert a JSON string into a JavaScript object. When you have a string that follows the JSON format, calling JSON.parse on that string allows the JavaScript engine to interpret the data contained within it and construct the corresponding JavaScript object. This is crucial when working with data that is received as a JSON string, for example, from an API response or a JSON file, as it enables developers to manipulate and access the data using standard JavaScript object notation. This functionality is essential for the integration of web applications, as JSON is a common format for data exchange. By parsing the JSON string, developers can work with the data in a structured way, making it easier to access specific properties and values of the object. The other options do not accurately describe the purpose of JSON.parse. While converting JavaScript objects to JSON strings is handled by JSON.stringify, validating JSON syntax requires a different approach, as JSON.parse is capable of throwing an error if the string is not valid JSON. Finally, storing JSON data across sessions involves mechanisms like localStorage or sessionStorage, rather than a method for parsing JSON data.

3. Which are the primary components of the Salesforce Lightning Framework?

A. Visualforce pages and Apex controllers

B. Aura components and Lightning Web Components (LWC)

C. JavaScript modules and CSS stylesheets

D. HTML templates and server-side scripts

The primary components of the Salesforce Lightning Framework are Aura components and Lightning Web Components (LWC). This framework allows developers to create dynamic, responsive user interfaces for applications that run on the Salesforce platform. Aura components serve as the original component model that supports a rich, event-driven architecture with features such as two-way data binding. They enable the creation of single-page applications and can communicate with each other through events. Lightning Web Components, introduced later, leverage modern web standards and provide a lightweight alternative to Aura components. They focus on performance and ease of use by utilizing standard JavaScript, making it easier for developers who are already familiar with modern web development practices. Together, these two types of components offer flexibility and efficiency in building robust applications on Salesforce, reinforcing the Lightning Framework as a powerful tool for developers. While Visualforce pages and Apex controllers are used in Salesforce as well, they do not constitute components of the Lightning Framework, but rather are part of the older Salesforce application architecture. JavaScript modules and CSS stylesheets are fundamental to web development, but they do not encapsulate the core structure of the Lightning Framework as specifically as Aura and LWC do. HTML templates and server-side scripts are also not primary components of the Lightning Framework, as they refer

4. Define a closure in JavaScript.

A. A closure is a block of code that defines an error

B. A closure is a property of an object

C. A closure is a function that retains access to its lexical scope even when the function is executed outside that scope

D. A closure is a variable that cannot be modified

A closure in JavaScript is defined as a function that retains access to its lexical scope, even when the function is executed outside that scope. This concept is fundamental in JavaScript due to its function scoping and the ability for inner functions to remember the environment in which they were created. When a function is defined within another function, the inner function forms a closure, capturing the variables that are within its lexical scope. This means that the inner function can access those variables even after the outer function has finished executing. This capability allows for the creation of private variables, encapsulated behavior, and helps maintain state in a controlled manner. For instance, consider a function that returns another function, which then references variables from its parent function. Despite the parent function having completed execution, the inner function still has access to those variables. This feature is widely used in JavaScript for various purposes such as creating factories, event handlers, and modules. This understanding of closures is essential for effective JavaScript programming, particularly in relation to asynchronous programming, callbacks, and event handling, where functions may be executed outside their original scope.

5. What will be the output of the following code: greetign = {};?

A. {}

B. ReferenceError: greetign is not defined

C. undefined

D. SyntaxError

The output of the code `greetign = {};` is an empty object represented as `{}`. When you execute this line, you are creating a variable named `greetign` and assigning it an empty object. In JavaScript, this is a valid operation, and it does not produce an error. This empty object can be used to store key-value pairs or functions later, but at this stage, it simply initializes `greetign` as an object with no properties. The notation `{}` is the standard way to define an object in JavaScript, and when logged or printed, it displays as `{}` indicating that it has been created successfully but contains no contents. The other options, like the ReferenceError, would occur if you tried to access `greetign` without declaring it first, or if there was a typo in the variable name at a point where it's supposed to be recognized. The `undefined` output applies when a variable is declared but not initialized with any value, which is different from creating an object. Lastly, a SyntaxError would be returned if the code broke JavaScript syntax rules, but the line in question follows proper syntax.

6. What is the output of the following code snippet when logged to the console? let num = 10; const increaseNumber = () => num++; const increasePassedNumber = number => number++; const num1 = increaseNumber(); const num2 = increasePassedNumber(num1); console.log(num1); console.log(num2);

A. 10, 10

B. 10, 11

C. 11, 11

D. 11, 12

The correct output of the code snippet relates to how the variables and functions interact in JavaScript, particularly in regards to the concepts of variable scope and return values. Initially, `num` is set to 10. The `increaseNumber` function increments `num` by 1 each time it's called and returns the original value of `num` before the increment occurs. So when `increaseNumber()` is invoked, it increases `num` from 10 to 11 but returns 10. This value, which is 10, is assigned to `num1`. Next, `increasePassedNumber` is a function that takes a number as an argument and increments that number, returning the original input value before the increment occurs. Here, `num1` (which is 10) is passed to this function. It effectively increments the value of `number` passed to it, so it receives 10, increments it, but returns the original value of 10. This is assigned to `num2`. Thus, when logging `num1`, we get 10 (the value returned by `increaseNumber` before the increment), and when logging `num2`, we also get 10 (the value passed to `increasePassedNumber`

7. What is the role of the wire service in LWC?

A. To manage server-side logic

B. To provide a way to read data from an Apex method

C. To enhance the styling of components

D. To handle user input

The wire service in Lightning Web Components (LWC) serves as a powerful mechanism for managing data interactions seamlessly between the client-side component and server-side resources. This service facilitates the retrieval of data from various sources, including Apex methods, without the need for intricate and manual call handling. By utilizing the wire service, developers can easily connect their components to data sources, enabling them to automatically fetch data. The retrieved data is then reactively reflected in the component state, ensuring that any changes in the data source are automatically updated in the UI. For instance, when an Apex method is set up to return data, using the wire service simplifies the process of invoking that method and handling the response. This is incredibly useful in ensuring that the component consumes data effectively, promotes efficiency in development, and reduces boilerplate code associated with data retrieval processes. Furthermore, the reactiveness provided by the wire service keeps the UI in sync with the backend data, enhancing the overall functionality and user experience of the application. Options that involve managing server-side logic, enhancing styling, or handling user input do not capture the primary function of the wire service, which is predominantly focused on data retrieval and synchronization with the UI in LWC.

8. In LWC, how do you handle events bubbling?

A. By stopping the event flow entirely

B. By using the event.stopPropagation() method

C. By cancelling the event listener

D. By allowing the event to proceed naturally

In Lightning Web Components (LWC), managing event bubbling is crucial for controlling how events propagate through the component tree. The selected approach involves using the event.stopPropagation() method, which is the standard way to prevent an event from bubbling up to parent components. When this method is invoked on an event within a child component, it stops the event from reaching any ancestor components. This can be particularly useful when you want to ensure that a specific action is contained within a child component without triggering any parent handlers. For instance, if you have a button click event in a child component that should not affect the parent when clicked, invoking event.stopPropagation() will achieve that. The other choices do not directly address the need for managing event bubbling as effectively. Stopping the event flow entirely would prevent any handling of the event, including that which might be necessary at the child level. Cancelling the event listener does not apply here, because it's generally used to remove event listeners rather than manage event flow. Allowing the event to proceed naturally means accepting the default behavior of bubbling, which may not be the desired outcome in all scenarios. Therefore, using event.stopPropagation() stands out as the most precise method for controlling the bubbling of events in LWC.

9. What will the output of the following code be: let array1 = ['one', 'two']; let array2 = ['three', 'four']; array1.push(...array2); console.log(...array1);?

A. one two

B. one two three four

C. [one, two, three, four]

D. ["one", "two", "three", "four"]

In the given code, the `array1` is initialized with the elements 'one' and 'two', while `array2` contains 'three' and 'four'. The key operation here is the `push` method, which is called with the spread operator (`...array2`). This operator expands the contents of `array2` and adds them as separate elements to `array1`. After executing `array1.push(...array2)`, `array1` will then contain four elements: 'one', 'two', 'three', and 'four'. When logging `...array1` to the console, the spread operator is again used, which takes each element of `array1` and outputs them as individual arguments. Therefore, the console output will display each element in `array1` separated by a space. Thus, the output of the code will be "one two three four", confirming that the selected answer is accurate.

10. What is the role of the wire service in LWC?

A. To handle all events within the component

B. To automatically retrieve data from Salesforce and keep component data in sync

C. To manage the rendering of HTML elements

D. To facilitate API requests to external services

The wire service in Lightning Web Components (LWC) plays a critical role in automatically retrieving data from Salesforce and ensuring that the component's data remains in sync with the underlying data source. It leverages reactive programming principles, which means that any changes in the underlying data result in automatic updates to the component without the need for manual intervention. When a wire service is used, it establishes a connection to the Salesforce data while efficiently handling the data retrieval process. This means that as the component is rendered or when relevant data changes, the wire service will re-fetch data as needed, allowing developers to focus on building interactive user interfaces instead of managing data state explicitly. This automatic synchronization is key when working on LWC, as it enhances performance and keeps the user interface responsive to changes in real-time. The wire service efficiently manages data access, reduces the amount of code developers need to write for fetching data, and leverages built-in caching mechanisms to minimize server requests, ultimately leading to a smoother user experience.

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://salesforce-javascriptdeveloper.examzify.com

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