

# Test Results and Interview Guide

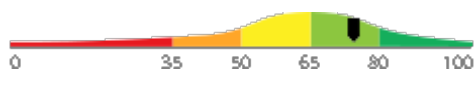
Candidate: **Elizabeth Wantsajob**  
Assessment: Javascript Programming (Web) (Short)  
Completed: June 27, 2026  
Prepared for: Sara Maple  
Example Company

## What's Included

- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

**Important Note:** The Javascript Programming (Web) (Short) assessment measures key factors related to high performance and tenure in this job. Attribute types measured vary by test, but can include cognitive ability, skills, knowledge, personality characteristics, emotional intelligence, and past behavioral history. This report includes a one page summary, followed by detailed results with an embedded interview guide. Note that these results should always be used as a part of a balanced candidate selection process that includes independent evaluation steps, such as interviews and reference checks.

## Overall

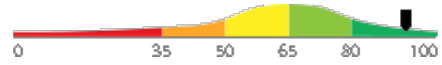
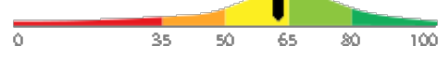



Candidate	Score	Interpretation
<b>Elizabeth Wantsajob</b> beth.wantsajob@gmail.com Javascript Programming (Web) (Short) June 27, 2026	<div style="background-color: #28a745; border-radius: 50%; width: 40px; height: 40px; display: flex; align-items: center; justify-content: center; margin: 0 auto;">74</div>	

The candidate demonstrates a solid and competent grasp of JavaScript programming for web applications, including scope management, object destructuring, DOM manipulation, browser event handling, and asynchronous operations using promises. Minor gaps may exist in advanced design patterns, performance optimization, or nuanced exception handling, but the candidate is well-suited for entry-level to mid-level web development responsibilities.

**Key**





- Candidate Score
- Higher Risk
- Lower Risk

## Competency Summary

Competency	Score	Interpretation
<b>Skills/Knowledge (relates to immediate readiness)</b>		
DOM Manipulation and Browser Events	93	
Functions, Classes, and Modules (Coding Tasks)	62	
Variables, Scope, and Data Structures (Coding Tasks)	62	
Functions, Classes, and Modules	85	
Variables, Scope, and Data Structures	68	

## Comparison

Percentile scores indicate how the candidate compares to other test-takers within various groups. The candidate scored equal to or better than the fraction of test-takers indicated by the percentile.

Test-Taker Group	Percentile	0	10	20	30	40	50	60	70	80	90	100	
Global	74th												
North America	61st												
United States	61st												
Example Company	68th												

## Detail

Candidate: Elizabeth Wantsajob, beth.wantsajob@gmail.com  
 Assessment: Javascript Programming (Web) (Short)  
 Authorized: June 27, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com  
 Started: June 27, 2026, 1:08:29PM EDT  
 Completed: June 27, 2026, 1:08:29PM EDT  
 Overall Score: 74

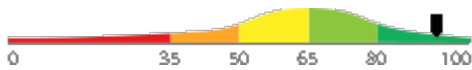
## Knowledge and Skills Detail

This section contains a list of job-related knowledge areas and skills that have been evaluated. Low scores in these areas often indicate that additional learning may be required before top performance can be achieved.

Detail
Interview Guide

### DOM Manipulation and Browser Events

Score: 93



*Description:*

Covers how JavaScript interacts with the Document Object Model to read and change web page content, style, and structure. Includes selecting elements, responding to user actions through event listeners, and validating form input.

*Interpretation:*

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits an advanced and comprehensive understanding of DOM manipulation and browser event handling in JavaScript. They are highly proficient in dynamically reading and modifying web page content, structure, and style, as well as implementing sophisticated event-driven logic and robust form validation.

How does event delegation work in JavaScript, and why might you use it instead of attaching individual event listeners to each element in a list?

- ★  
1
- ★  
2
- ★  
3
- ★  
4
- ★  
5

Cannot explain event delegation or how events bubble through the DOM.

Understands event bubbling but cannot clearly explain the performance or maintenance benefits.

Explains bubbling, target checking, and clearly articulates performance and dynamic element benefits.

How would you use JavaScript to find an element on a web page and change its text content when a user clicks a button?

- ★  
1
- ★  
2
- ★  
3
- ★  
4
- ★  
5

Cannot identify correct DOM selection methods or describe how to attach an event listener.

Describes the general approach but makes errors in syntax or method names.

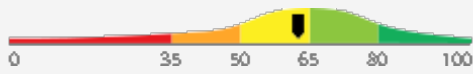
Accurately describes querySelector, addEventListener, and textContent with correct syntax.

Detail

Interview Guide

**Functions, Classes, and Modules (Coding Tasks)**

Score: 62



*Description:*

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

*Interpretation:*

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?



1

Cannot write correct allocation code; unaware of free() or memory leak risks.



2

Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.



3



4

Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.



5

Can you explain what a pointer is in C and describe a situation where you would use one?



1

Vague or incorrect definition; cannot describe a practical use case.



2

Correct basic definition; gives a simple but valid use case with some gaps.



3



4

Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

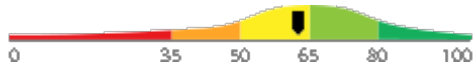


5

Detail Interview Guide

**Variables, Scope, and Data Structures (Coding Tasks)**

Score: 62



*Description:*

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

*Interpretation:*

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?



1

Cannot write correct allocation code; unaware of free() or memory leak risks.



2

Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.



3



4

Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.



5

Can you explain what a pointer is in C and describe a situation where you would use one?



1

Vague or incorrect definition; cannot describe a practical use case.



2

Correct basic definition; gives a simple but valid use case with some gaps.



3



4

Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.



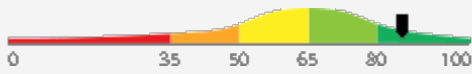
5

Detail

Interview Guide

**Functions, Classes, and Modules**

Score: 85



*Description:*

Covers how JavaScript uses functions, classes, and ES modules to organize and structure application logic. Includes understanding function declarations, arrow functions, class syntax, inheritance, and how to import and export code across files.

*Interpretation:*

Candidate should achieve superior job performance in this area with little or no training.

The candidate demonstrates a comprehensive and advanced understanding of JavaScript functions, classes, and ES modules, reflecting strong proficiency across all key areas including arrow functions, class inheritance, and modular code organization. This level of performance indicates the ability to effectively structure and maintain complex JavaScript application logic.

How do JavaScript ES modules help organize a web application, and how would you structure a small project using import and export statements across multiple files?



1

Cannot explain what modules are or how import/export syntax works.



2

Understands basic import/export but cannot explain named vs. default exports or module scope.



3



4

Clearly explains module scope, named vs. default exports, and describes a logical file structure.



5

What is the difference between a regular function and an arrow function in JavaScript, and when would you choose one over the other?



1

Cannot identify syntax differences or any behavioral distinction between the two.



2

Identifies syntax differences but cannot clearly explain how 'this' binding differs.



3



4

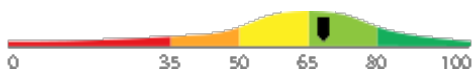
Accurately explains 'this' binding, lack of arguments object, and gives practical use cases.



5

**Variables, Scope, and Data Structures**

Score: 68



*Description:*

Covers how JavaScript variables are declared using var, let, and const, and how scope rules determine where variables are accessible. Includes working with arrays and objects, including common operations like iteration, destructuring, and state management.

*Interpretation:*

Candidate should achieve above average job performance in this area with little or no training.

The candidate demonstrates a solid and competent understanding of JavaScript programming structures, syntax, and common design patterns used in client-side web application development. They are likely capable of writing, debugging, and maintaining application logic involving modules, DOM manipulation, asynchronous operations, and error handling, with some room for continued growth in more advanced or nuanced implementation areas.

How would you use array destructuring and object destructuring to simplify code that manages application state, and what are some common mistakes developers make when using destructuring?



1

Cannot explain destructuring syntax or confuses it with other assignment patterns.



2

Demonstrates basic destructuring syntax but misses edge cases like default values or renaming.



3



4

Fluently explains both forms, covers defaults, renaming, nested patterns, and common pitfalls.



5

Can you explain the difference between var, let, and const, and describe a situation where using the wrong one could cause a bug in your code?



1

Cannot distinguish between var, let, and const or explain scope differences.



2

Explains basic differences but struggles to connect scope rules to real bugs.



3



4

Clearly explains hoisting, block vs. function scope, and gives a concrete bug example.



5

## IT Coding Tasks

During the assessment, the candidate was asked to write one or more programs or scripts. Their responses are included below for review.

Question or Task	Response
<p>Complete the provided partial C program by filling in the missing sections marked with TODO comments. Your completion must use standard C keywords and libraries.</p> <p>The program declares a function called <code>duplicate_array</code> that:</p> <ol style="list-style-type: none"> <li>1. Takes a const int pointer to a source array and its length as parameters.</li> <li>2. Uses <code>calloc</code> to allocate a new int array of the same length.</li> <li>3. Returns NULL if <code>calloc</code> fails.</li> <li>4. Copies each element from the source array into the new array using pointer arithmetic (not array subscript notation).</li> <li>5. Returns the pointer to the newly allocated copy.</li> </ol> <p>In main, the program:</p> <ol style="list-style-type: none"> <li>1. Declares and initializes a stack array of 4 integers with values 5, 15, 25, 35.</li> <li>2. Calls <code>duplicate_array</code> to create a heap-allocated copy.</li> <li>3. Checks for NULL and prints an error and returns 1 if the call failed.</li> <li>4. Prints each element of the duplicate using a loop.</li> <li>5. Frees the duplicate array.</li> </ol> <p>Use appropriate indentation, common C coding conventions, and add brief inline comments where needed. Type your completed source code as your response.</p>	<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt;  int *duplicate_array(const int *src, int length) {     /* TODO: Use calloc to allocate a new array of 'length' integers, return        NULL if calloc fails, copy elements from src using pointer arithmetic,        and return the new pointer. */     calloc(303); }  int main(void) {     /* TODO: Declare and initialize a stack array of 4 integers: 5, 15, 25, 35,        then call duplicate_array and store the result. Check for NULL and        print an error message returning 1 if it failed. */     array[4]={5,15,25,35};      int i;      /* Print each element of the duplicate */     for (i = 0; i &lt; 4; i++) {         printf("duplicate[%d] = %d\n", i, *(duplicate + i));     }      /* Free the duplicate array */     free(duplicate);     return 0; }</pre>

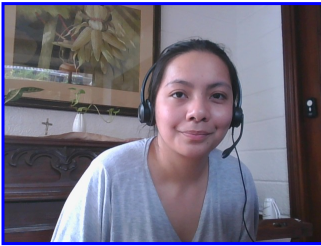
**Comments (AI):** The code segment has several syntax errors and incomplete implementation. The `duplicate_array` function does not correctly allocate memory or copy elements. The main function has syntax errors and does not properly call the `duplicate_array` function. However, the structure and intent of the code are somewhat clear, and the code attempts to follow the requirements.

## Identity Confirmation Photos

The following photos of the candidate and any identification were uploaded during the assessment session.

### Photo Analysis Results

<b>- Risk:</b>	<b>Medium risk of cheating based on image inconsistencies</b>
- Percent match among processed faces	100%
- Total images processed	17
- Total images with valid faces	14 (82%)
- Total pairs of faces compared	13
- Pairs in which faces matched	13 (100%)



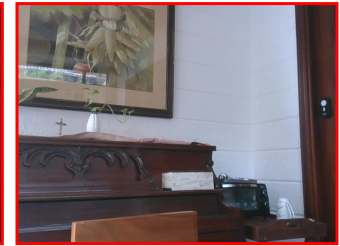
Pre/Post-Test Photo



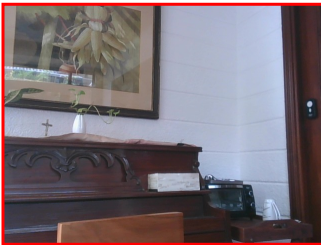
ID Photo



In-Test Error Detected (No Face Detected)



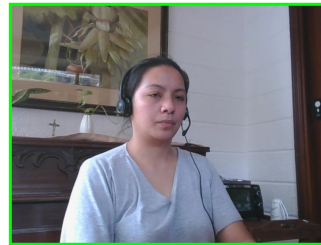
In-Test Error Detected (No Face Detected)



In-Test Error Detected (No Face Detected)



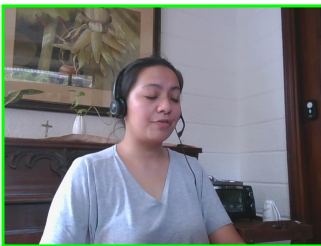
In-Test Photo



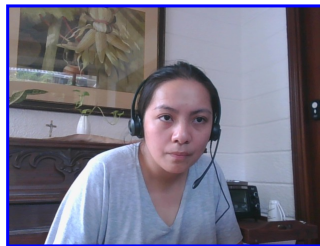
In-Test Photo



In-Test Photo



In-Test Photo



Pre/Post-Test Photo

## Resume or CV

[Summary](#)[Updated on](#)

Motivated career professional with extensive experience in office administration and management. Proven track record of improving efficiency, reducing costs, and enhancing office operations through strategic initiatives and technology implementation.

### Objective

I am seeking a role where I can use my many skills and my exceptional judgment and empathy for customers to make a difference to a growing company.

### Education

- Associate of Applied Science in Office Administration, Portland Community College, 2020

### Experience

- General Office Clerk, Paramount Office Management, 09/2023 – Present
- Administrative Assistant, Global Enterprises Inc., 04/2021 – 08/2023
- Administrative Assistant, Innovative Business Solutions Ltd., 07/2019 – 03/2021

### Other Qualifications

- Microsoft Office Specialist (MOS) Certification
- Certified Administrative Professional (CAP)
- International Association of Administrative Professionals (IAAP) Certification

## Report Preparation Notes

- Hiring decisions should never be based on a single source of information. The most effective use of this assessment report is as a part of a multi-faceted program of candidate evaluation that includes resume review, interviews, and reference checks.
- Overall vs Percentiles Scores: The overall score reflects the success in the test, based on the mean (average) and standard deviation of the test scores. The percentile score reflects the percentage of test-takers who scored equal or below this overall score. We recommend you use the Overall Score as your primary evaluation criteria. However, percentile scores can often be useful in comparing specific candidates against one another and with a group, such as for test takers in a certain organization or within a certain account.
- Note that comparison information is calculated based on completed instances of this assessment at that time the assessment is scored. As additional instances are completed, the comparative data may change. You can always update a report to the current values by clicking on 'Recalculate Percentiles' within the online results viewing pages at [www.hravatar.com](http://www.hravatar.com).
- Most competency scores are norm-based, which means that they can be interpreted in terms of their distance from the average or mean score. For all scales, a score equal to the mean receives a score of 65 and scores above and below this value are set so that a score change of 15 equals one standard deviation.
- For linear competencies, higher is better across the entire scale. For these scales a score between 65 and 80 (light green) represents 0 to 1 standard deviation above the mean and a score above 80 (dark green) represents more than one standard deviation above the mean. Similarly, a score of 50 - 65 (yellow) represents 0 to 1 standard deviation below the mean, while a score of 35 - 50 (orange) equates to 1 to 2 standard deviations below the mean, and a score below 35 represents more than 2 standard deviations below the mean.
- Sim ID: 20787-1, Key: 0-0, Rpt: 68, Prd: 9609, Created: 2026-06-27 13:08 EDT
- UA: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; Touch; rv:11.0) like Gecko

## Score Calculation Detail

The following table provides a summary of how the overall score was calculated from each of the individual competency scores. First, all competency scores are calculated on a scale of 0-100. Note that some competencies use their color category rather than their actual numeric score in the overall calculation. For these, a standard score associated with the assigned color category is used in the overall score calculation rather than the actual numeric score. This is reflected in the "Score Value Used" column. Next, a weighted average of scores is computed using individual competency weights, typically set using job analysis data provided by the US Government Occupational Information Network (O\*Net).

Competency	Score	How applied to overall	Score Value Used	Weight (%)
DOM Manipulation and Browser Events	93.3812	Numeric Score	93.3812	20.0000
Functions, Classes, and Modules	85.6401	Numeric Score	85.6401	20.0000
Functions, Classes, and Modules (Coding Tasks)	62.9784	Numeric Score	62.9784	20.0000
Variables, Scope, and Data Structures	68.5617	Numeric Score	68.5617	20.0000
Variables, Scope, and Data Structures (Coding Tasks)	62.9784	Numeric Score	62.9784	20.0000
Weighted Average:				74.7080
Final Overall Score:				74

## Notes

(This area is intentionally blank - it's reserved as space for your notes.)