hr | avatar
Illuminating Talent

Test Results and Interview Guide

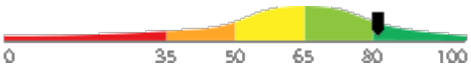
Candidate: **Elizabeth Wantsajob**
Assessment: Javascript Programming (Web)
Completed: June 27, 2026
Prepared for: Sara Maple
Example Company

What's Included

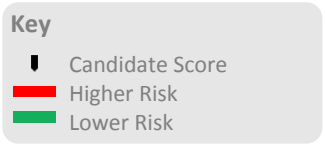
- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The Javascript Programming (Web) 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.

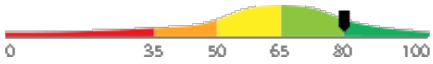
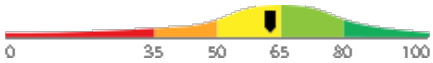
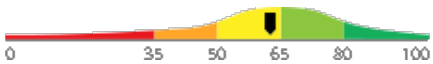
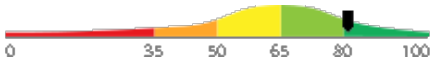
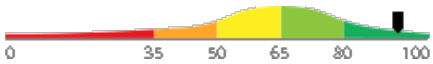
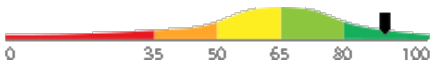
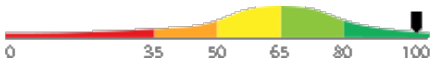
Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com Javascript Programming (Web) June 27, 2026	81	

The candidate exhibits a comprehensive and proficient command of JavaScript programming, encompassing advanced syntax, modular and class-based architecture, DOM manipulation, asynchronous data fetching, input validation, and robust error and exception handling. This level of performance reflects strong readiness for mid-level web development roles and the ability to independently write, debug, and maintain complex client-side web application components.



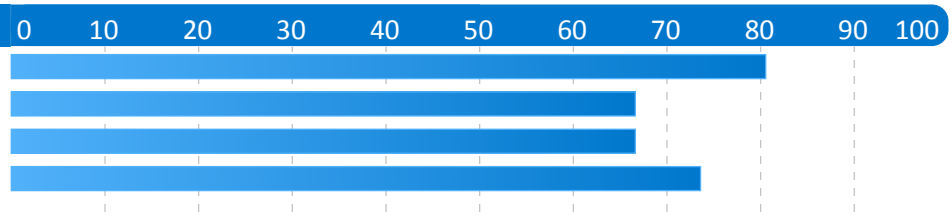
Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Asynchronous JavaScript and Fetch API	80	
Functions, Classes, and Modules (Coding Tasks)	62	
Variables, Scope, and Data Structures (Coding Tasks)	62	
DOM Manipulation and Browser Events	81	
Error Handling and Debugging	93	
Functions, Classes, and Modules	89	
Variables, Scope, and Data Structures	97	

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
Global	81st
North America	67th
United States	67th
Example Company	74th



Artificial Intelligence (AI) Generated Scores

This table includes one or more scores derived from a large language model AI query. AI-derived scores are non-deterministic. That is, they are not precisely repeatable. Therefore, these scores should always be treated as supplementary information and should never be used exclusively or compared to hard cutoff values.

Estimated Value	Score	Confidence	Interpretation
Knowledge, Skills, and Abilities Summary	-	-	Summary Points (AI): • (Generic Text for Sample Report) Strong performer in Drag and Drop Files tasks, indicating comfort with file management and basic computer interactions. • Demonstrates solid numerical accuracy in Recognizing and Confirming Numbers, a valuable asset in detail-oriented roles. • Moderate overall performance in Analytical Thinking and Attention to Detail, with adequate grammar skills but room for improvement. • Struggles with Reading and Analyzing Problems, which may limit effectiveness in roles requiring critical reading and complex problem-solving. • Lowest performance in Navigating Between Screens, suggesting difficulty with multi-screen software workflows that could impact productivity in computer-intensive roles. Narrative (AI): Elizabeth Wantsajob demonstrates a mixed profile of knowledge, skills, and abilities across the assessed competencies. Elizabeth shows a strong aptitude in Drag and Drop Files, performing well on this technical task and suggesting she is comfortable with this type of computer interaction. This is a notable strength that would translate well into roles requiring file management and basic computer navigation tasks. In the area of Analytical Thinking and Attention to Detail, Elizabeth performs at a moderate level. She demonstrates solid ability in Recognizing and Confirming Numbers, which suggests she is careful and accurate when working with numerical data — a valuable skill in detail-oriented work environments. Her Grammar performance is adequate but leaves room for improvement, indicating she may occasionally make written communication errors. Her weakest area within this competency is Reading and Analyzing Problems, where she struggled to consistently interpret and work through written problem scenarios. This may impact her effectiveness in roles that require critical reading, written comprehension, or complex problem-solving. Elizabeth's most significant area for development is Navigating Between Screens, where she scored considerably lower than the other competencies. This suggests she may have difficulty efficiently moving through software interfaces or multi-screen workflows, which could slow productivity in roles that rely heavily on navigating computer applications or data entry systems. Overall, Elizabeth brings some useful technical strengths, particularly in file management and numerical accuracy, but would benefit from targeted development in software navigation and analytical problem-solving to be fully effective in roles that demand these skills. Computed on: April 2, 2026, 11:09:49PM EDT

Detail

Candidate:	Elizabeth Wantsajob, beth.wantsajob@gmail.com
Assessment:	Javascript Programming (Web)
Authorized:	June 27, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 27, 2026, 1:20:17PM EDT
Completed:	June 27, 2026, 1:20:17PM EDT
Overall Score:	81

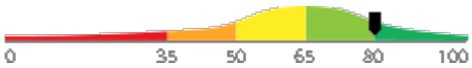
Knowledge and Skills Detail

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

Asynchronous JavaScript and Fetch API

Score: 80



Description:

Covers how JavaScript handles operations that take time, such as retrieving data from a server. Includes understanding promises, async/await syntax, and using the Fetch API to send and receive data from network endpoints.

Interpretation:

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

The candidate demonstrates a strong and comprehensive command of asynchronous JavaScript and the Fetch API, including advanced use of promises, async/await syntax, and network communication patterns. They are well-equipped to independently design, implement, and troubleshoot complex asynchronous operations in professional web programming environments.

Walk me through how you would use the Fetch API with async/await to retrieve JSON data from a server endpoint and display it on a web page, including how you would handle errors.

1	2	3	4	5
Cannot write or describe a fetch call or explain how async/await relates to Promises.		Describes a basic fetch call but omits error handling or misunderstands async/await flow.		Accurately describes fetch, await, JSON parsing, try/catch error handling, and DOM update steps.

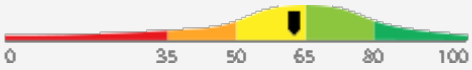
What is a Promise in JavaScript, and how does it help you work with code that takes time to finish, like loading data from a server?

1	2	3	4	5
Cannot explain what a Promise is or how it differs from synchronous code.		Gives a general description of Promises but cannot explain resolve, reject, or chaining.		Clearly explains Promise states, chaining with .then/.catch, and connects to real async use cases.

Detail

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.

Interview Guide

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	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

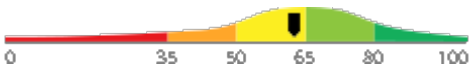
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

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.

Interview Guide

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	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

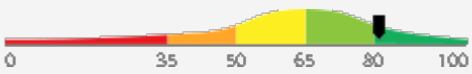
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

DOM Manipulation and Browser Events

Score: 81



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.

Interview Guide

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?

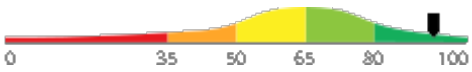
1	2	3	4	5
Cannot explain event delegation or how events bubble through the DOM.		Understands event bubbling but cannot clearly explain the performance or maintenance benefits.		Explains bubbling, target checking, and clearly articulates performance and dynamic element benefits.

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.

Error Handling and Debugging

Score: 93



Description:

Covers how to identify and fix problems in JavaScript code. Includes using try/catch blocks to manage runtime errors, reading browser console output, using breakpoints and developer tools to trace bugs, and identifying performance issues.

Interpretation:

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

The candidate demonstrates an advanced and comprehensive mastery of error handling and debugging within JavaScript web programming. They are highly proficient in leveraging try/catch blocks, browser developer tools, breakpoints, and console output to efficiently diagnose runtime errors and performance issues, reflecting a strong command of best practices in this domain.

How and when would you use a try/catch block in JavaScript, and what information can you get from the error object that is caught to help diagnose a problem?

1	2	3	4	5
Cannot write a try/catch block or explain when it would be useful.		Writes basic try/catch syntax but cannot explain error object properties or re-throwing errors.		Explains try/catch/finally, error.message, error.stack, custom errors, and appropriate use cases.

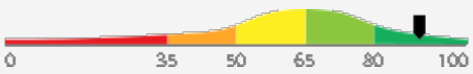
If your JavaScript code throws an unexpected error and the page stops working, what steps would you take to find and fix the problem?

1	2	3	4	5
Cannot describe any structured debugging approach or mention browser developer tools.		Mentions checking the console but cannot describe using breakpoints or reading stack traces.		Describes reading console errors, stack traces, setting breakpoints, and isolating the problem area.

Detail

Functions, Classes, and Modules

Score: 89



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.

Interview Guide

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?

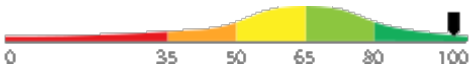
1	2	3	4	5
Cannot explain what modules are or how import/export syntax works.		Understands basic import/export but cannot explain named vs. default exports or module scope.		Clearly explains module scope, named vs. default exports, and describes a logical file structure.

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

1	2	3	4	5
Cannot identify syntax differences or any behavioral distinction between the two.		Identifies syntax differences but cannot clearly explain how 'this' binding differs.		Accurately explains 'this' binding, lack of arguments object, and gives practical use cases.

Variables, Scope, and Data Structures

Score: 97



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

The candidate exhibits a strong and comprehensive mastery of JavaScript programming concepts, structures, and design patterns for client-side web applications. They demonstrate advanced proficiency across the full breadth of the subject area, including modular architecture, asynchronous communication, DOM manipulation, exception handling, and browser performance analysis, indicating readiness to independently contribute to or lead development efforts at a mid-level or higher capacity.

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	2	3	4	5
Cannot explain destructuring syntax or confuses it with other assignment patterns.		Demonstrates basic destructuring syntax but misses edge cases like default values or renaming.		Fluently explains both forms, covers defaults, renaming, nested patterns, and common pitfalls.

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	2	3	4	5
Cannot distinguish between var, let, and const or explain scope differences.		Explains basic differences but struggles to connect scope rules to real bugs.		Clearly explains hoisting, block vs. function scope, and gives a concrete bug example.

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.

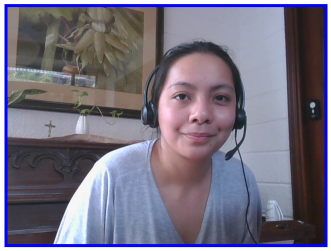
Question or Task	Response

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.
 	
The program declares a function called duplicate_array that:
1. Takes a const int pointer to a source array and its length as parameters.
2. Uses calloc to allocate a new int array of the same length.
3. Returns NULL if calloc fails.
4. Copies each element from the source array into the new array using pointer arithmetic (not array subscript notation).
5. Returns the pointer to the newly allocated copy.
 	
In main, the program:
1. Declares and initializes a stack array of 4 integers with values 5, 15, 25, 35.
2. Calls duplicate_array to create a heap-allocated copy.
3. Checks for NULL and prints an error and returns 1 if the call failed.
4. Prints each element of the duplicate using a loop.
5. Frees the duplicate array.
 	
Use appropriate indentation, common C coding conventions, and add brief inline comments where needed. Type your completed source code as your response.
	#include <stdio.h>
	#include <stdlib.h>
	
	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 < 4; i++) {
	printf("duplicate[%d] = %d\n", i, *(duplicate + i));
	}
	
	/* Free the duplicate array */
	free(duplicate);
	return 0;
	}
	
	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

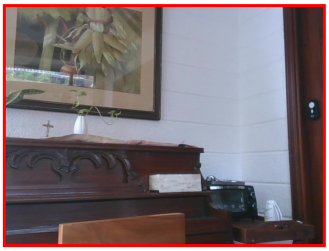
The following photos of the candidate and any identification were uploaded during the assessment session.

Photo Analysis Results	
- Risk:	Medium risk of cheating based on image inconsistencies
- 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%)

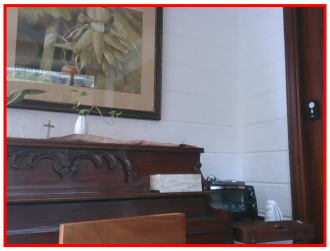


Pre/Post-Test Photo

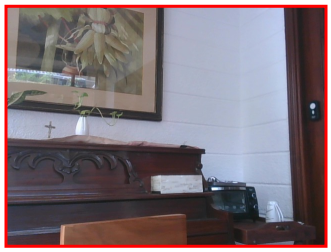
ID Photo



In-Test Error Detected (No Face Detected)

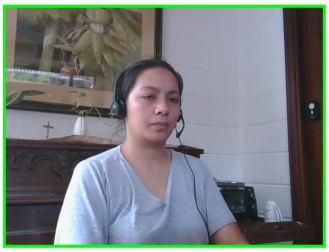


In-Test Error Detected (No Face Detected)

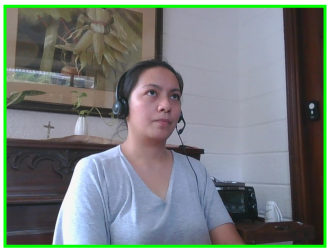


In-Test Error Detected (No Face Detected)

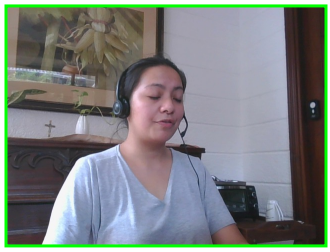
In-Test Photo



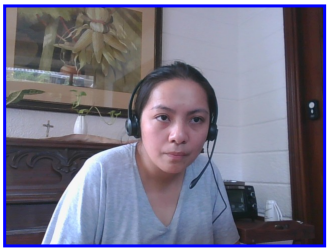
In-Test Photo



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.
- 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: 20786-1, Key: 0-0, Rpt: 68, Prd: 9608, Created: 2026-06-27 13:20 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 (%)
Asynchronous JavaScript and Fetch API	80.6388	Numeric Score	80.6388	14.2857
DOM Manipulation and Browser Events	81.4107	Numeric Score	81.4107	14.2857
Error Handling and Debugging	93.3917	Numeric Score	93.3917	14.2857
Functions, Classes, and Modules	89.9676	Numeric Score	89.9676	14.2857
Functions, Classes, and Modules (Coding Tasks)	62.9784	Numeric Score	62.9784	14.2857
Variables, Scope, and Data Structures	97.5935	Numeric Score	97.5935	14.2857
Variables, Scope, and Data Structures (Coding Tasks)	62.9784	Numeric Score	62.9784	14.2857
Weighted Average:				81.2799
Final Overall Score:				81

Notes

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