hr | avatar
Illuminating Talent

Test Results and Interview Guide

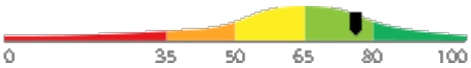
Candidate: **Elizabeth Wantsajob**
Assessment: TypeScript Programming
Completed: June 27, 2026
Prepared for: Sara Maple
Example Company

What's Included

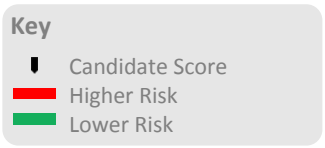
- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The TypeScript Programming 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.

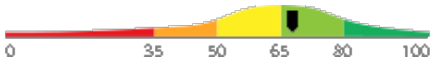
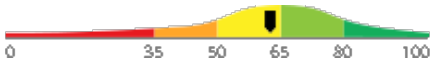
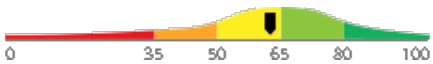
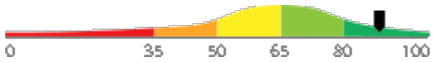
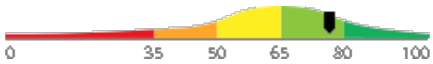
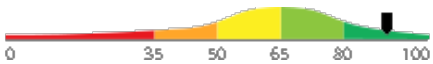
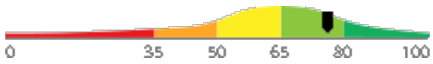
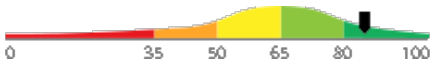
Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com TypeScript Programming June 27, 2026	76	

The candidate exhibits a solid and competent command of TypeScript, reflecting the ability to work effectively with types, interfaces, generics, inheritance, and asynchronous programming patterns in a professional development environment. Minor gaps may exist in advanced areas such as complex type narrowing, decorator usage, or compiler configuration, but the candidate is well-positioned to contribute meaningfully to TypeScript projects with minimal supervision.



Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Async Programming and Error Handling	68	
Functions and Classes (Coding Tasks)	62	
Types, Interfaces, and Type Aliases (Coding Tasks)	62	
Functions and Classes	88	
Generics and Utility Types	76	
Modules, Project Configuration, and Tooling	90	
Types, Interfaces, and Type Aliases	76	
Variable Declarations, Scoping, and Null Handling	85	

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	76th
North America	63rd
United States	63rd
Example Company	70th

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:	TypeScript Programming
Authorized:	June 27, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 27, 2026, 8:10:07PM EDT
Completed:	June 27, 2026, 8:10:07PM EDT
Overall Score:	76

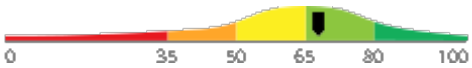
Knowledge and Skills Detail

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

Async Programming and Error Handling

Score: 68



Description:

Covers how to write asynchronous code using async/await and Promises, including chaining and handling resolved and rejected states. Also includes using try/catch blocks for error handling and understanding how TypeScript types errors in catch blocks.

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 asynchronous programming and error handling in TypeScript. They are likely proficient with async/await, Promise chaining, and handling both resolved and rejected states, with a reasonable grasp of how TypeScript types errors within catch blocks.

When you catch an error in a try/catch block in TypeScript with strict mode enabled, what type is the error variable, and how do you safely work with it?

1	2	3	4	5
Assumes the error is typed as Error and does not mention the unknown type or type narrowing.		Knows the error is unknown but is unsure how to narrow it to access message or other properties.		Correctly identifies unknown, uses instanceof Error or a type guard to narrow it, and explains why this is necessary.

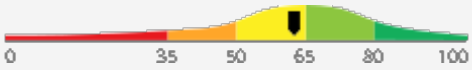
Can you explain what the async and await keywords do in TypeScript and show how you would use them to fetch data and handle an error?

1	2	3	4	5
Cannot explain async/await or writes an example that does not handle errors at all.		Writes a basic async function but forgets try/catch or makes an error in the syntax.		Writes a clean async function with proper await usage, a try/catch block, and correct error handling.

Detail

Functions and Classes (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.

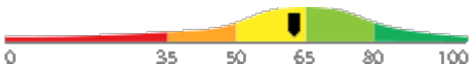
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Types, Interfaces, and Type Aliases (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.

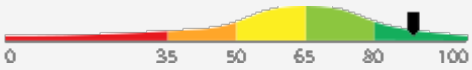
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Functions and Classes

Score: 88



Description:

Covers how to write typed functions including optional and default parameters, return types, and arrow functions. Also includes defining classes with constructors, properties, methods, access modifiers (public, private, protected), and inheritance using extends and implements.

Interpretation:

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

The candidate exhibits an advanced and comprehensive mastery of TypeScript functions and classes. They are highly proficient in writing fully typed functions with optional, default, and inferred parameters, as well as designing robust class hierarchies utilizing constructors, access modifiers, and both inheritance and interface implementation.

Interview Guide

Walk me through how you would design a base class and a subclass in TypeScript, including how the subclass would override a method from the base class.

1	2	3	4	5
Cannot demonstrate inheritance or method overriding with correct TypeScript syntax.		Creates a working example but omits type annotations or makes minor syntax errors.		Produces a clean example with proper typing, uses super correctly, and explains the override keyword or pattern.

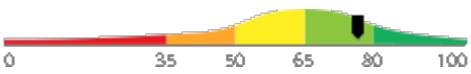
How do you declare a class in TypeScript with a constructor that sets a private property, and how would you access that property from outside the class?

1	2	3	4	5
Cannot write a basic class or does not understand why a private property cannot be accessed directly.		Writes a class with some errors or misunderstands one aspect of access modifiers.		Correctly writes the class, explains private access restriction, and suggests a getter method as a solution.

Detail

Generics and Utility Types

Score: 76



Description:

Covers how to write reusable, flexible code using generic functions, classes, and interfaces. Also includes practical use of built-in TypeScript utility types such as Partial, Required, Pick, Omit, and Readonly to transform and work with existing types.

Interpretation:

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

The candidate shows a solid and competent understanding of TypeScript Generics and Utility Types, capable of writing reusable and flexible code using generic constructs. They are proficient with most built-in utility types and can apply them reliably to transform and work with existing types in practical development scenarios.

Interview Guide

If you have an interface with several required properties and you want to create a version where all properties are optional, how would you do that in TypeScript?

1	2	3	4	5
Manually rewrites the interface with optional properties, unaware of the Partial utility type.		Knows Partial exists but cannot fully explain how it works or when to use it.		Immediately identifies Partial, explains how it works, and may mention other utility types like Required or Pick.

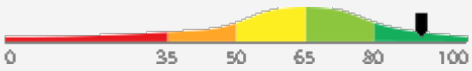
What is a generic in TypeScript, and can you give a simple example of a generic function that works with any type?

1	2	3	4	5
Cannot explain generics or provides an example that does not actually use a type parameter.		Explains the concept in general terms but writes an incomplete or slightly incorrect example.		Clearly explains generics and writes a correct, simple generic function with a type parameter.

Detail

Modules, Project Configuration, and Tooling

Score: 90



Description:

Covers how to organize code across multiple files using ES module syntax (import/export), configure a TypeScript project using tsconfig.json, and manage packages with npm. Also includes using the TypeScript compiler to catch errors and understanding common compiler options like strict, target, and module.

Interpretation:

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

The candidate exhibits an advanced and comprehensive mastery of TypeScript modules, project configuration, and tooling. They are highly proficient in organizing codebases with ES module syntax, fine-tuning tsconfig.json compiler options, managing dependencies with npm, and leveraging the TypeScript compiler to enforce type safety and catch errors effectively.

Interview Guide

What is the purpose of the tsconfig.json file, and what are two or three compiler options you commonly set in a TypeScript project and why?

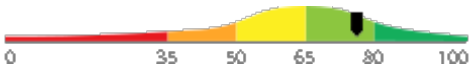
1	2	3	4	5
Has little knowledge of tsconfig.json or can only name one option without explaining its purpose.		Names a few options like strict or target but gives vague or partially correct explanations.		Explains tsconfig.json clearly, names relevant options like strict, target, and moduleResolution, and explains their practical impact.

How do you export a function from one TypeScript file and import it into another file?

1	2	3	4	5
Cannot recall the correct import/export syntax or confuses named and default exports.		Knows the basic syntax but is unsure about the difference between named and default exports.		Correctly demonstrates both named and default export/import syntax and explains when to use each.

Types, Interfaces, and Type Aliases

Score: 76



Description:

Covers how to define and use basic types (string, number, boolean, etc.), create interfaces and type aliases, and apply union types, intersection types, and type assertions. This is the foundation of TypeScript and is used in nearly every line of TypeScript code written in a business application.

Interpretation:

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

The candidate exhibits a solid working knowledge of TypeScript, including competence with types, interfaces, generics, modules, and asynchronous programming patterns. They are likely capable of writing, debugging, and maintaining TypeScript applications with moderate complexity, though some advanced topics such as advanced utility types, decorators, or nuanced compiler configuration may require further refinement. This candidate is well-suited for entry-level to mid-level TypeScript development roles.

How would you use a union type and a type guard together to safely handle a value that could be either a string or a number in a function?

1	2	3	4	5
Cannot explain union types or type guards, or provides an incorrect implementation.		Describes the concept correctly but produces an incomplete or partially incorrect code example.		Writes a clean, correct function using a union type and a typeof type guard with no errors.

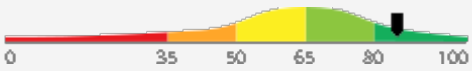
Can you explain the difference between a type alias and an interface in TypeScript, and give an example of when you might use each one?

1	2	3	4	5
Confuses the two or cannot provide a meaningful distinction or example.		Explains the basic difference but struggles to give a clear, practical example.		Clearly distinguishes both, notes key differences like declaration merging, and gives relevant examples.

Detail

Variable Declarations, Scoping, and Null Handling

Score: 85



Description:

Covers the correct use of let and const for variable declarations and their block-scoping rules. Also includes handling null and undefined safely using strictNullChecks, optional chaining (?.), nullish coalescing (??), and non-null assertion operators.

Interpretation:

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

The candidate exhibits an advanced and comprehensive understanding of TypeScript variable declarations, block-scoping semantics, and all major null and undefined handling techniques, including strictNullChecks, optional chaining, nullish coalescing, and non-null assertion operators. They are highly capable of writing safe, well-scoped TypeScript code and can confidently navigate complex edge cases in these areas. This candidate demonstrates expert-level readiness to work with or mentor others on these TypeScript fundamentals.

Interview Guide

How would you safely access a deeply nested property on an object that might be null or undefined at any level, without throwing a runtime error?

1	2	3	4	5
Uses nested if checks or is unaware of optional chaining syntax.		Knows optional chaining but cannot combine it with nullish coalescing to provide a fallback value.		Uses optional chaining (?.) fluently, combines it with nullish coalescing (??) for a fallback, and explains why this is safer.

What is the difference between let and const in TypeScript, and can you give an example of a scoping issue that could occur if you misuse them?

1	2	3	4	5
Cannot explain the difference or provides an incorrect or irrelevant scoping example.		Correctly explains the difference but gives a vague or incomplete scoping example.		Clearly explains mutability and block scoping, and gives a concrete example such as a loop or block scope issue.

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.

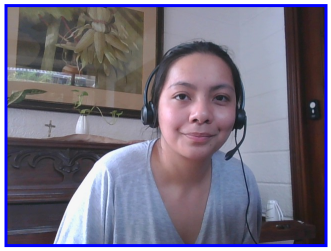
Question or Task	Response

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

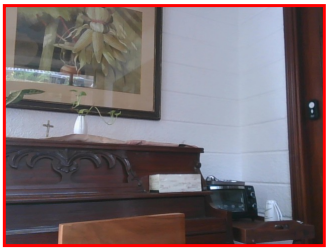
The following photos of the candidate and any identification were uploaded during the assessment session.

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%)

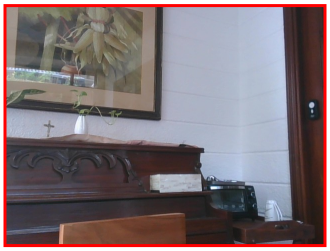


Pre/Post-Test Photo

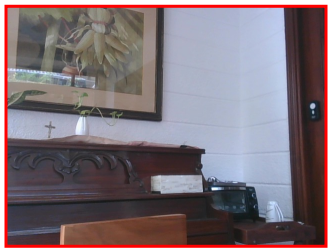
ID Photo



In-Test Error Detected (No Face Detected)

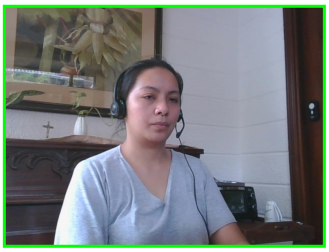


In-Test Error Detected (No Face Detected)

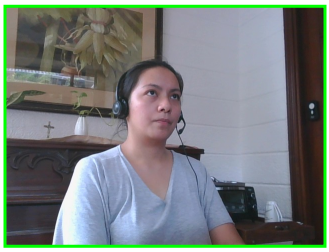


In-Test Error Detected (No Face Detected)

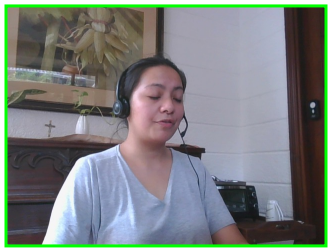
In-Test Photo



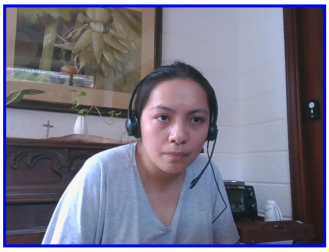
In-Test Photo



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: 20800-1, Key: 0-0, Rpt: 68, Prd: 9622, Created: 2026-06-27 20:10 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 (%)
Async Programming and Error Handling	68.0032	Numeric Score	68.0032	12.5000
Functions and Classes	88.8763	Numeric Score	88.8763	12.5000
Functions and Classes (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Generics and Utility Types	76.7556	Numeric Score	76.7556	12.5000
Modules, Project Configuration, and Tooling	90.3850	Numeric Score	90.3850	12.5000
Types, Interfaces, and Type Aliases	76.5058	Numeric Score	76.5058	12.5000
Types, Interfaces, and Type Aliases (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Variable Declarations, Scoping, and Null Handling	85.1214	Numeric Score	85.1214	12.5000
Weighted Average:				76.4505
Final Overall Score:				76

Notes

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