

# Test Results and Interview Guide

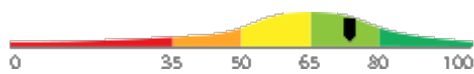
Candidate: **Elizabeth Wantsajob**  
Assessment: TypeScript Programming (Short)  
Completed: June 28, 2026  
Prepared for: Sara Maple  
Example Company

## What's Included

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

**Important Note:** The TypeScript Programming (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

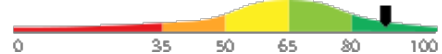
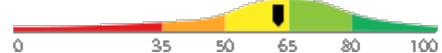
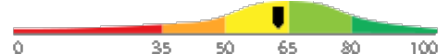
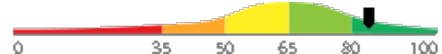
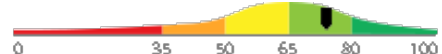
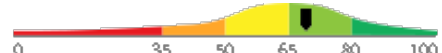
Candidate	Score	Interpretation
<b>Elizabeth Wantsajob</b> beth.wantsajob@gmail.com TypeScript Programming (Short) June 28, 2026	<div style="background-color: #28a745; border-radius: 50%; width: 40px; height: 40px; display: flex; align-items: center; justify-content: center; margin: 0 auto;">73</div>	

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.

**Key**


- Candidate Score
- Higher Risk
- Lower Risk

## Competency Summary

Competency	Score	Interpretation
<i>Skills/Knowledge (relates to immediate readiness)</i>		
Async Programming and Error Handling	88	
Functions and Classes (Coding Tasks)	62	
Types, Interfaces, and Type Aliases (Coding Tasks)	62	
Functions and Classes	84	
Generics and Utility Types	74	
Types, Interfaces, and Type Aliases	69	

## 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	73rd												
North America	61st												
United States	61st												
Example Company	67th												

## 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	-	-	<p>Summary Points (AI):</p> <ul style="list-style-type: none"> <li>(Generic Text for Sample Report) Strong performer in Drag and Drop Files tasks, indicating comfort with file management and basic computer interactions.</li> <li>Demonstrates solid numerical accuracy in Recognizing and Confirming Numbers, a valuable asset in detail-oriented roles.</li> <li>Moderate overall performance in Analytical Thinking and Attention to Detail, with adequate grammar skills but room for improvement.</li> <li>Struggles with Reading and Analyzing Problems, which may limit effectiveness in roles requiring critical reading and complex problem-solving.</li> <li>Lowest performance in Navigating Between Screens, suggesting difficulty with multi-screen software workflows that could impact productivity in computer-intensive roles.</li> </ul> <p>Narrative (AI): Elizabeth Wantsajob demonstrates a mixed profile of knowledge, skills, and abilities across the assessed competencies.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>Computed on: April 2, 2026, 11:09:49PM EDT</p>

## Detail

Candidate: Elizabeth Wantsajob, beth.wantsajob@gmail.com  
 Assessment: TypeScript Programming (Short)  
 Authorized: June 28, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com  
 Started: June 28, 2026, 9:13:24AM EDT  
 Completed: June 28, 2026, 9:13:24AM EDT  
 Overall Score: 73

## 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: 88



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

The candidate exhibits an advanced and comprehensive mastery of asynchronous programming and error handling in TypeScript. They are highly proficient with async/await, Promise chaining, resolved and rejected state management, try/catch error handling, and TypeScript's approach to typing errors in catch blocks, and can be expected to apply these skills effectively in complex, production-level scenarios.

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

Assumes the error is typed as Error and does not mention the unknown type or type narrowing.



2

Knows the error is unknown but is unsure how to narrow it to access message or other properties.



3



4

Correctly identifies unknown, uses instanceof Error or a type guard to narrow it, and explains why this is necessary.



5

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

Cannot explain async/await or writes an example that does not handle errors at all.



2

Writes a basic async function but forgets try/catch or makes an error in the syntax.



3



4

Writes a clean async function with proper await usage, a try/catch block, and correct error handling.



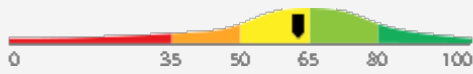
5

Detail

Interview Guide

**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.

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.

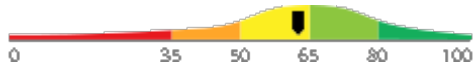


5

Detail Interview Guide

**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.

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.



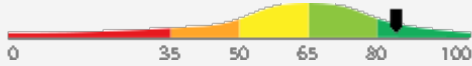
5

Detail

Interview Guide

**Functions and Classes**

Score: 84



*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.

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

Cannot demonstrate inheritance or method overriding with correct TypeScript syntax.



2

Creates a working example but omits type annotations or makes minor syntax errors.



3



4

Produces a clean example with proper typing, uses super correctly, and explains the override keyword or pattern.



5

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

Cannot write a basic class or does not understand why a private property cannot be accessed directly.



2

Writes a class with some errors or misunderstands one aspect of access modifiers.



3



4

Correctly writes the class, explains private access restriction, and suggests a getter method as a solution.

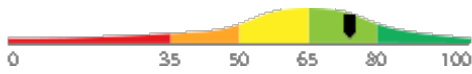


5

Detail Interview Guide

**Generics and Utility Types**

Score: 74



*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.

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

Manually rewrites the interface with optional properties, unaware of the Partial utility type.



2

Knows Partial exists but cannot fully explain how it works or when to use it.



3



4

Immediately identifies Partial, explains how it works, and may mention other utility types like Required or Pick.



5

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



1

Cannot explain generics or provides an example that does not actually use a type parameter.



2

Explains the concept in general terms but writes an incomplete or slightly incorrect example.



3



4

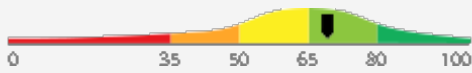
Clearly explains generics and writes a correct, simple generic function with a type parameter.



5

**Types, Interfaces, and Type Aliases**

Score: 69



*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

Cannot explain union types or type guards, or provides an incorrect implementation.



2

Describes the concept correctly but produces an incomplete or partially incorrect code example.



3



4

Writes a clean, correct function using a union type and a typeof type guard with no errors.



5

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

Confuses the two or cannot provide a meaningful distinction or example.



2

Explains the basic difference but struggles to give a clear, practical example.



3



4

Clearly distinguishes both, notes key differences like declaration merging, and gives relevant examples.



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>

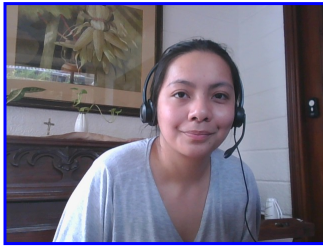
**Comments (AI):** The code segment has several syntax errors and incomplete implementation. The `duplicate_array` function does not correctly allocate memory or copy elements. The main function has syntax errors and does not properly call the `duplicate_array` function. However, the structure and intent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

- 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



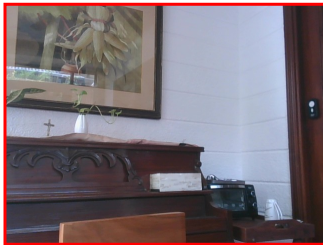
ID Photo



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](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: 20804-1, Key: 0-0, Rpt: 68, Prd: 9626, Created: 2026-06-28 09:13 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	88.4906	Numeric Score	88.4906	16.6667
Functions and Classes	84.5025	Numeric Score	84.5025	16.6667
Functions and Classes (Coding Tasks)	62.9784	Numeric Score	62.9784	16.6667
Generics and Utility Types	74.4715	Numeric Score	74.4715	16.6667
Types, Interfaces, and Type Aliases	69.6763	Numeric Score	69.6763	16.6667
Types, Interfaces, and Type Aliases (Coding Tasks)	62.9784	Numeric Score	62.9784	16.6667
Weighted Average:				73.8496
Final Overall Score:				73

## Notes

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