TS: Exercises for Functions

Depicted Candela

September 2, 2024

1 Function Types and Signatures

Exercise 1: Function Type Expressions

1. Define a function type for a function that calculates the distance between two points in a 2D space (each point has x and y coordinates).
2. Create a function that matches the above type and computes the distance.

Exercise 2: Call Signatures

1. Define a callable object that can be used to calculate the perimeter of a rectangle given width and height. The object should also have a method scale(factor: number) to scale the dimensions of the rectangle. Include both call and construct signatures.

Exercise 3: Declaring this in Functions

1. Write a function for a simple geometry object that calculates the area of a triangle. Ensure the function can be used as a method and specify the type of `this` to ensure type safety when invoked.

2 Generics in TypeScript Functions

Exercise 4: Writing Generic Functions

1. Create a generic function that accepts an array of geometric shapes (e.g., rectangles, circles) and returns the shape with the maximum area. Use generics to ensure type safety for different shape types.

Exercise 5: Type Inference and Constraints

1. Define a generic function to merge properties of two objects: one representing a geometric shape and another containing additional metadata (e.g., color, name). Use constraints to ensure the objects have specific properties (e.g., shapes must have an area method).

Exercise 6: Guidelines for Writing Good Generic Functions

1. Refactor the previous exercise's generic function to reduce the number of type parameters and ensure that each type parameter appears at least twice in the function signature.

3 Function Overloads and Usage

Exercise 7: Writing Function Overloads

1. Create function overloads for a function named `translate` that moves a geometric shape by a specified distance. It should handle:
    - Moving by a fixed distance in both x and y directions.
    - Moving by a distance object with dx and dy properties.

Exercise 8: Overload Signatures vs. Implementation Signatures

1. Implement the `translate` function with correct overloads and an implementation signature that handles both overload cases correctly.

Exercise 9: Additional Guidelines for Overloads

1. Modify the `translate` function to use union types instead of overloads where appropriate, and compare the results in terms of simplicity and readability.

4 Optional Parameters and Callback Functions

Exercise 10: Optional Parameters

1. Write a function that calculates the area of a polygon with optional parameters for units (e.g., square meters, square kilometers). The default should be square meters.

Exercise 11: Optional Parameters in Callbacks

1. Define a function that takes a callback to compute the centroid of a set of points. The callback should optionally accept an error parameter if the input set is empty or invalid.

5 Advanced TypeScript Function Topics

Exercise 12: Syntax Differences and Advanced Requirements

1. Compare and contrast two functions: one using a function expression and another using an arrow function. Discuss the implications of this in each case within the context of a geometry application.

6 Construct Signatures

Exercise 13: Writing Construct Signatures

1. Define a construct signature for a class that represents a geometric shape that can be instantiated with `new` to create shapes like circles, rectangles, or polygons.