Day 1: Introduction to TypeScript and Basic Types

Objective: To understand the fundamentals of TypeScript, set up the development environment, and learn how to use basic types.

Introduction to TypeScript (15 minutes)

1. What is TypeScript and Why Use It?

TypeScript is a programming language developed by Microsoft that builds on top of JavaScript. It's a "superset" of JavaScript, meaning any valid JavaScript code is also valid TypeScript code. The primary feature TypeScript adds to JavaScript is **static typing**.

Why is this important?

- **Error Detection:** In regular JavaScript, you might accidentally try to perform an operation on the wrong type of data, which you would only discover when you run the code (a runtime error). TypeScript checks for these types of errors as you code, catching them before you even run the program.
- **Improved Readability and Maintainability:** Explicitly defining the types of data your code expects makes it easier for you and other developers to understand what the code is doing, especially in large and complex applications.
- **Better Tooling:** TypeScript's static typing allows for more intelligent code completion, refactoring, and error checking in your code editor.

2. The Relationship Between TypeScript and JavaScript

Think of TypeScript as a more advanced version of JavaScript. Browsers and other JavaScript environments cannot execute TypeScript directly. Therefore, you need a special program called the TypeScript compiler to "transpile" or convert your TypeScript code into standard JavaScript code that can run anywhere.

3. Overview of the TypeScript Compiler (tsc)

The TypeScript compiler, known as `tsc`, is a command-line tool that reads your TypeScript files (with a `.ts` extension) and generates corresponding JavaScript files (with a `.js` extension). During this process, it checks your code for any type-related errors.

The compiler is highly configurable through a file called `tsconfig.json`, which allows you to specify various options for how your code should be compiled.

4. Activity: Setting Up a Basic TypeScript Project

Let's get our hands dirty and set up a simple TypeScript project.

1. **Install Node.js:** If you don't have it already, download and install Node.js from the official website. This will also install the Node Package Manager (npm).

2. **Install TypeScript:** Open your terminal or command prompt and run the following command to install TypeScript globally on your system:

   ```
   npm install -g typescript
   ```

3. **Create a Project Folder:** Make a new folder for your project and navigate into it in your terminal.

   ```
   mkdir typescript-day1
   cd typescript-day1
   ```

4. **Initialize a TypeScript Configuration File:** Run the following command to create a `tsconfig.json` file. This file tells the TypeScript compiler how to handle your project.

   ```
   tsc --init
   ```

5. **Create Your First TypeScript File:** Create a new file named `index.ts` and add the following code:

   ```
   let message: string = "Hello, TypeScript!";
   console.log(message);
   ```

6. **Compile Your TypeScript Code:** In your terminal, run the compiler:

   ```
   tsc
   ```

   This will generate a new `index.js` file.

7. **Run Your JavaScript Code:** You can now run the compiled JavaScript file using Node.js:

   ```
   node index.js
   ```

   You should see “Hello, TypeScript!” printed in your console.

Basic Types and Type Annotations (30 minutes)

In TypeScript, you can explicitly define the type of a variable. This is called **type annotation**.

1. Number, String, Boolean, Null, Undefined

These are the fundamental data types in JavaScript, and they work the same way in TypeScript.

- **`number`**: For all numeric values, including integers and floating-point numbers.
- **`string`**: For textual data.
- **`boolean`**: For `true` or `false` values.
- **`null`**: Represents the intentional absence of any object value.
- **`undefined`**: Represents a variable that has been declared but not yet assigned a value.

2. The `any` and `unknown` Types

- **`any`**: The any type is a powerful way to opt-out of type-checking for a variable. You can assign any type of value to a variable of type any. Use this sparingly as it undermines the benefits of TypeScript.
- **`unknown`**: The unknown type is a safer alternative to any. You can assign any value to a variable of type unknown, but you cannot perform any operations on it until you have performed a type check.

3. Type Inference: Letting TypeScript Figure Out the Types

TypeScript is smart! If you declare and initialize a variable in the same statement, TypeScript will infer the type for you.

4. Working with Arrays and Tuples

- **Arrays**: You can define an array of a specific type.
- **Tuples**: Tuples are a special kind of array with a fixed number of elements of known types. The order of the types matters.

5. Hands-on: Declaring Variables with Different Basic Types and Creating Typed Arrays

Let's practice what we've learned. In your `index.ts` file, try the following:

```
// Basic Types
let age: number = 30;
let firstName: string = "John";
let isStudent: boolean = false;

// Type Inference
let lastName = "Doe"; // TypeScript infers this as a string

// any and unknown
let anything: any = "This could be anything";
anything = 42;

let maybe: unknown = "This is a string";
// The following line would cause an error until we check the type
// console.log(maybe.length);

if (typeof maybe === "string") {
  console.log(maybe.length);
}

// Arrays
let numbers: number[] = [1, 2, 3, 4, 5];
let names: string[] = ["Alice", "Bob", "Charlie"];

// Tuples
let person: [string, number] = ["David", 25];

// Let's log these to the console
console.log(age, firstName, isStudent);
console.log(lastName);
console.log(numbers);
console.log(person);
```

After adding this code, re-compile your `index.ts` file with `tsc` and run the resulting `index.js` file with `node index.js`.

Q&A and Wrap-up (15 minutes)

- **Review of Key Concepts:**
  - TypeScript is a superset of JavaScript that adds static typing.
  - The TypeScript compiler (`tsc`) converts TypeScript code to JavaScript.
  - We've learned about basic types: `number`, `string`, `boolean`, `null`, `undefined`, `any`, and `unknown`.
  - TypeScript can infer types.
  - We can create typed arrays and tuples.
- **Preview of Day 2:**
  - Tomorrow, we will dive into how to apply types to functions and how to structure our data more effectively using objects and interfaces.