Introduction to TypeScript

SoftUni Team

Technical Trainers

Software University

https://softuni.bg

Have a Question?

sli.do

#TypeScript

Table of Contents

1. Introduction to TypeScript
2. TypeScript vs JavaScript
3. Environment and Setup
4. Basic Data Types
5. Debugging

Introduction to TypeScript

What is TypeScript?

- TypeScript is an **open-source** programming language developed by Microsoft
- It is a **statically typed** superset of JavaScript that transpires to plain JavaScript
- TypeScript adds optional **static typing**, making it more robust and maintainable

Why Use TypeScript?

- Static Typing: Helps catch errors during development, improving **code quality** and **reliability**
- Better Tooling: Enhanced code editor support with **intelligent auto-completion**, navigation, and refactoring

Why Use TypeScript?

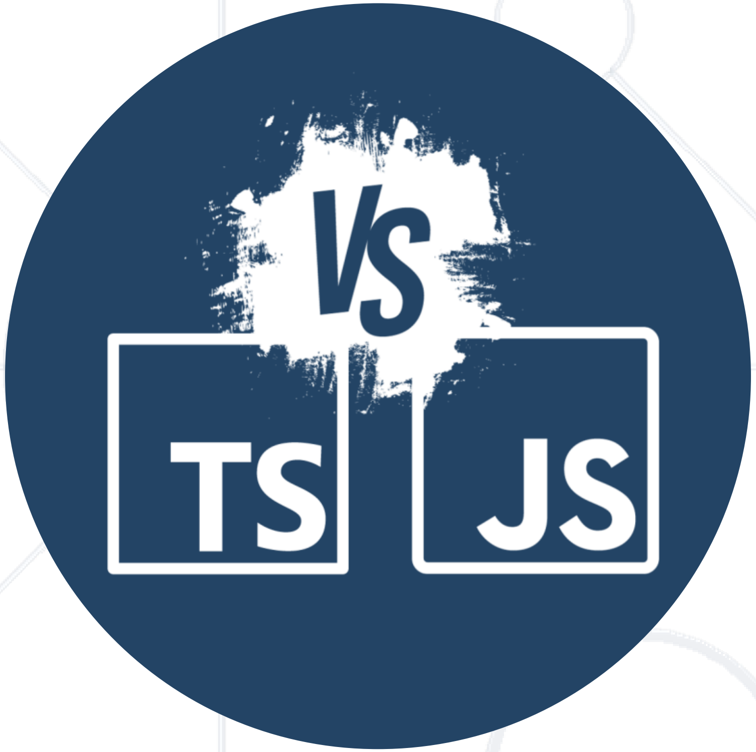
- **Readability and Maintainability**: Type annotations provide **self-documentation**, making code easier to understand and maintain
- **Scalability**: Suitable for **large-scale applications** with a strong type system

Key Features of TypeScript

- Static Typing: Types are **inferred** or **explicitly** declared, catching type-related errors during development
- Interfaces: Define contracts for **object shapes**, **enhancing code** readability and maintainability
- Classes: Follow **object-oriented principles** with support for constructors, properties, and methods

Key Features of TypeScript

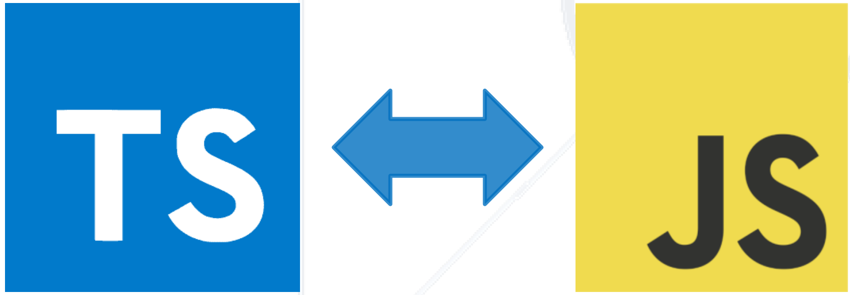
- **Enums**: Define a set of **named constants** for improved code readability
- **Generics**: Write **flexible** and **reusable** code components
- **Modules**: Organize code into **logical** and **reusable** units for better maintainability



TypeScript vs JavaScript

TypeScript vs JavaScript

- JavaScript: A **dynamic, loosely** typed language widely used for **web development**
- TypeScript: A **statically typed** superset of JavaScript that provides additional **features** and **tools** for better development experience



TypeScript vs JavaScript

- TypeScript

```
class Person {
  private firstName: string;
  constructor(fName: string) {
    this.firstName = fName;
  }
  greeting() {
    return `${this.firstName}`
  }
}
```

- JavaScript

```
"use strict";
class Person {
  constructor(fName) {
    this.firstName = fName;
  }
  greeting() {
    return  `${this.firstName}`;
  }
}
```

Environment and Setup

Install Visual Studio Code

- In this course we will use and demonstrate on:
  - Visual Studio Code
  - Installation Guidelines
- Alternatives:
  - WebStorm
  - JS Fiddle

Install TypeScript to Visual Studio Code

- Install **TypeScript** with **npm**

```
npm install -g typescript (latest stable build)
```

- Test if **TypeScript** is **installed properly**

```
tsc --version //Should return a message 'Version 5.x.x'.
```

- Create the **tsconfig.json** file

```
tsc --init – This command will create a new tsconfig.json file
```

Configuration of "tsconfig.json"

- In the tsconfig.json file, please **remove the comments** from the following:

```
{
  "compilerOptions" : {
       "target": "esnext",   // ECMAScript target version
      "module": "esnext",   // module code generation
      "sourceMap": true,    // Generates corresponding .map file
      "strict": true,       // strict type-checking options
      "outDir": "out",      // redirect output to the directory.
    }
}
```

Transpilation vs Compilation

- Transpilation
  - Source code is **translated** to a similar-level language.
  - Output is in a **similar abstraction** level
  - **Example:** TypeScript to JavaScript
- Compilation
  - Source code is translated to a **lower-level language**
  - Output is in a form suitable for **direct execution** by the machine

Basic Data Types

Basic Data Types

- **String** - used to represent **textual** data

```
let str: string = 'hello';
str = 'singleQuotes' ; // valid
str = "doubleQuotes" ; // valid
str = 11; // invalid
```

- **Number** - used to represent **numeric** data

```
let decimal: number = 11; // valid
let hex: number = 7E3; // valid
let binary: number = 11111100011 // valid
let float: number = 3.14 // valid
decimal = 'hello'; // invalid
```

Basic Data Types

- **Boolean** - only **true** and **false** values
  - Functions or expressions that return **true** or **false** values may also be assigned to **Boolean** data type

```
let isBool: boolean = true;
isBool = 5 < 2; // valid
let numbers = [1, 2, 3, 4];
isBool = numbers.includes(100) // valid
isBool = 11; // invalid
```

Basic Data Types

- **Symbol** - used to represent **unique** data

```
let uniqueSymbol: symbol = Symbol('mySymbol');
let anotherSymbol: symbol = Symbol('mySymbol');
console.log(uniqueSymbol === anotherSymbol); // false
```

- **null and undefined** - special types used to represent absence of a value in variables and functions

```
let undefinedValue1; // undefined
let undefinedValue2: undefined = undefined;
let person: null = null
```

Basic Data Types

- **Array** - use any valid data type (String, Boolean, Number) and postfix []

```
let arrayOfStr: string[];
arrayOfStr.push('Hello'); // valid
arrayOfStr.push('World'); // valid
arrayOfStr.push(11); // invalid
```

- **Tuple** - array with fixed number of elements whose types are known

```
let tuple:[string, number];
tuple = ['Hello', 11]; // valid
tuple = [11, 'Hello']; // invalid
```

Basic Data Types

- **Enum** - gives sets of numeric values more readable names
  - By default each enum starts at 0

```
enum DaysOfTheWeek {
    Monday,  // 0
    Tuesday, // 1
    …
};
let day: DaysOfTheWeek;
day = DaysOfTheWeek.Monday;
console.log(day); // 0
if (day === DaysOfTheWeek.Monday) {
    console.log('I hope you all had a great weekend!');
} // It will print the message
```

Basic Data Types

- **Any** and **Unknown** - takes any and all values. It's a way to escape the strong types. Unknown is safer

```
let a: any = 'hello'; // let a: unknown = 'hello';
a = true; // valid
a = 11 ; // valid
```

- **Void** - mainly used in functions that return no value

```
function greet(message: string): void {
    console.log(message);
}
```

Optional Data Types

- The **optional** data types are marked with **?**
  - Required parameters **cannot** follow optional ones

```
function optionalParams(name: string, mail?: string) {
     // some logic
} // valid

function optionalParams(name?: string, mail: string) {
     // some logic
} // invalid
```

Return Data Types

- The **return data types** are marked with **:** after the braces in function declaration
  - The **return value type** should match the **return type**

```
function greet (name: string): string {
    return name;
}

console.log(greet('Hello'));
```

Debugging

Debugging in VS Code

- Utilizing VS Code's powerful **integrated debugger** to find and fix issues in your TypeScript code
- Setting **breakpoints**, inspecting **variables**, and stepping through code

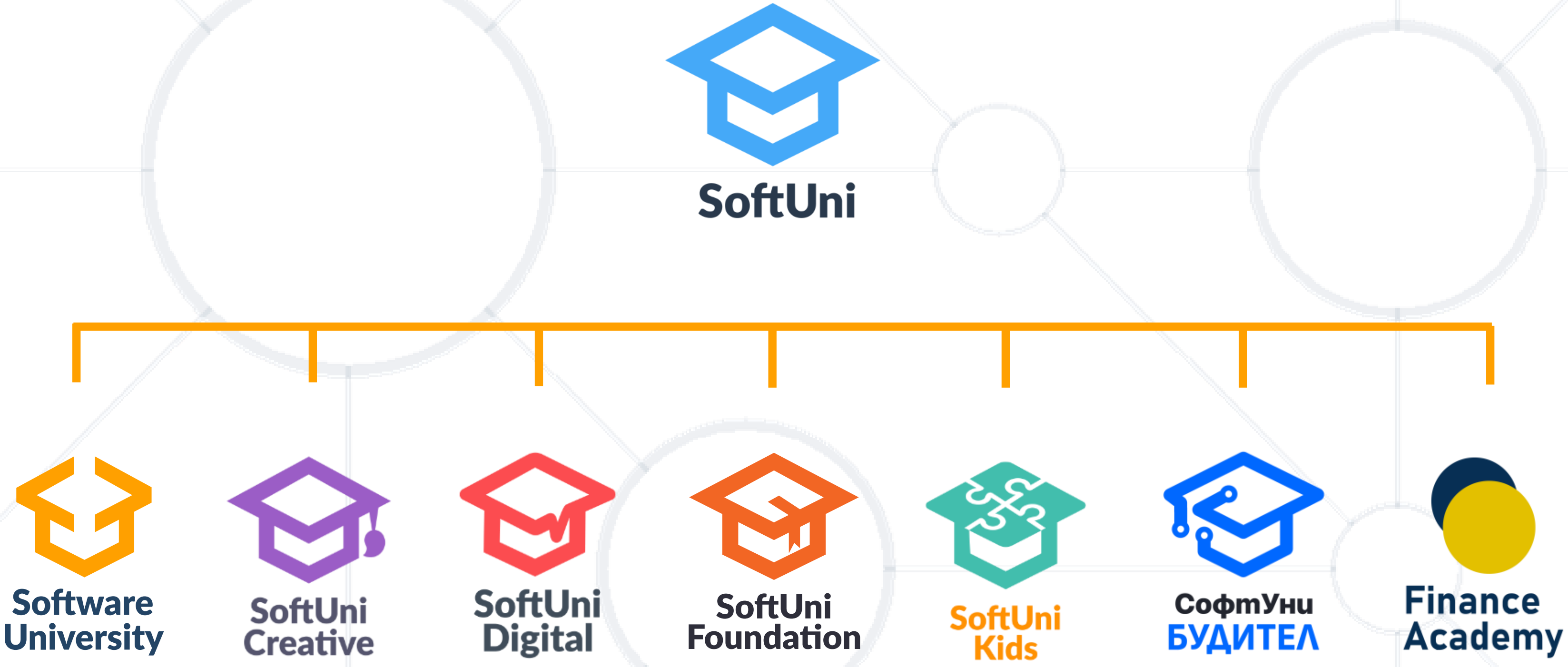
Debugging in VS Code

- Initialize a **TypeScript Project**:
  - Create a **tsconfig.json** file to configure TypeScript settings for the project
- **Launch Configurations**:
  - Configure a **launch.json** file to define how VS Code launches the debugging process
  - Set up configurations for **different scenarios**

Summary

- TypeScript presents strong typing to your JavaScript code
  - let, const and var are used to declare variables
  - There are basic (Number, String, Boolean, etc.) and more advanced data types like union or intersection
- Functions can:
  - Take optional and required parameters and return result

Questions?



SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

SmartIT

createX

VIVACOM

INDEAVR
Serving the high achievers

Coca-Cola HBC
Bulgaria

Postbank
Решения за твоето утре

AMBITIONED

SUPER
HOSTING
.BG

DRAFT
KINGS
THE CROWN IS YOURS

PHAR
VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

Software
University

License