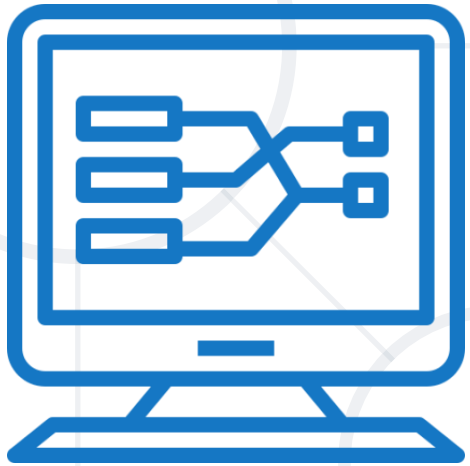
# TypeScript Type System

## Basic and Advanced data types




**SoftUni Team**

**Technical Trainers**

Software University

SoftUni Foundation





**Software University**

http://softuni.bg

# Table of Content



# Have a Question?

SoftUni Foundation

## sli.do

# #typescript


# Introduction to TypeScript

# TypeScript and JavaScript

- TypeScript is a **superset** of JavaScript
- Created by Microsoft Corporation
- All JavaScript code is **valid** in TypeScript too
- TypeScript **compiles to** JavaScript

# TypeScript vs JavaScript

SoftUni Foundation

- TypeScript

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

- JavaScript

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

# Install TypeScript to Visual Studio Code

SoftUni Foundation

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

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

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

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

- Create the **tsconfig.json** file

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

# tsconfig.json

SoftUni Foundation

- 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.
    }
}
```

# Basic Data Types

SoftUni Foundation

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

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

- **Number** - a numeric data type

```
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

SoftUni Foundation

- **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

SoftUni Foundation

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

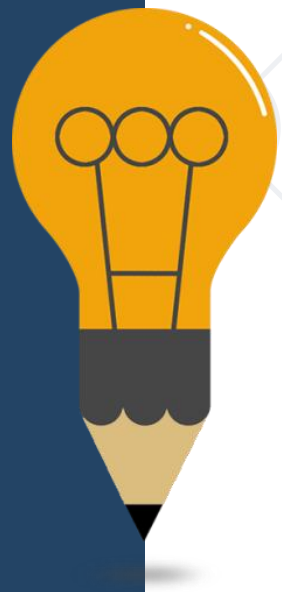
```
let arrayOfStr: string[];
arrayOfStr.push(`Hello`); //valid
arrayOfStr.push(`World`); //valid
arrayOfStr.push(11); //invalid
```

- **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

SoftUni Foundation

- **Any** - takes any and all values. It`s a way to escape the strong types

```
let example: any = `hello`;
example = true; //valid
example = 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**

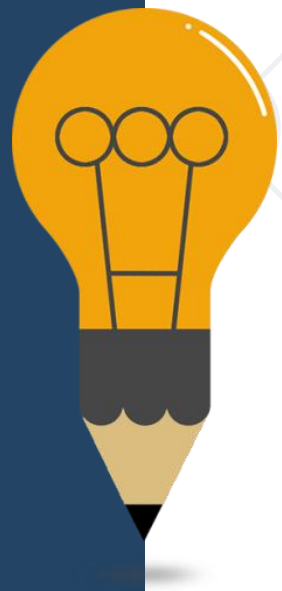
```
function greet (name: string): string {
    return name;
}

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

# Advanced Data Types

SoftUni Foundation

- **Union type** - combine multiple types in one type



```
function greet(message: string | string[]) {
    if (typeof message === "string") {
        return message;
    }
    return message.join(' ');
}
let greeting = 'Hello world';
let greetingArray = ['Dear', 'Sir/Madam'];

console.log(greet(greetingArray)); //Dear Sir/Madam
```

# Advanced Data Types

- **Intersection types** - combine multiple types in one type

```
interface Person { fullName: string | string[]; }
interface Contact { email: string; }
function showContact(contactPerson: Person & Contact) {
    return contactPerson;
}
let contactPerson: Person & Contact = {
    fullName: 'Svetoslav Dimitrov',
    email: 'test@test.com'
}
console.log(showContact(contactPerson));
```

# Problem: Mathematical operations

- Write a **TypeScript function** that makes **simple mathematical operations** over an array of numbers
  - It will receive two parameters: **array of numbers and operation**
    - The operations might be: **addition**, **multiplication** **or finding the largest number**

# Solution: Mathematical operations

SoftUni Foundation

```
function solve(arrOfNums: number[], operation: string): number {
    let result: number = 0;
    const addition = () => result = arrOfNums.reduce((a, b) => a + b, 0);
    const multiplication = () => result = arrOfNums.reduce((a, b) =
> a * b, 1);
    const largestNumber = () => result = Math.max(...arrOfNums);
    const actions = {
        'Addition': addition,
        'Multiplication': multiplication,
        'Largest number': largestNumber
    }
    actions[operation]();
    return result;
}
```

# Live Exercises

# 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

# SoftUni Diamond Partners

SoftUni Foundation













# SoftUni Organizational Partners

SoftUni Foundation




# Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

# License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

# Trainings @ Software University (SoftUni)

SoftUni Foundation

- Software University - High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg