Software Development

TypeScript and OOP

Module 08

TypeScript

What is TypeScript?

TypeScript

Programming Language

TypeScript is a **typed superset** of JavaScript.

All JavaScript code works in TypeScript.

TypeScript adds new rules to JavaScript that dictate how different values can be used.

Compiler

TypeScript is a **compiler** that converts TypeScript syntax into clean JavaScript that can be consumed anywhere.

Language Server

TypeScript is a **language server** that provides instant error checking in your IDE while you code.

Why TypeScript?

As applications grow and become more complex, JavaScript's flexibility can introduce more bugs than it solves.

TypeScript ensures code operates as intended by introducing **type-checking** to catch errors and unexpected behavior as they happen.

Types

- JavaScript provides built-in types such as `string`, `boolean`, and `number`.
- Vanilla JavaScript allows you to reassign types.
- TypeScript adds a layer of protection that checks that you are consistently assigning these types.
- While this layer of protection is helpful for the code, it also provides more precise information to future developers who use your codebase.

```
// Plain JavaScript file (.js)
let message = "Hello, world!";
message = 42;

// TypeScript file (.ts)
let message = "Hello, world!";
message = 42; // Error: Type
`number` cannot be assigned to
type `string`
```

TypeScript Syntax

- TypeScript leverages its JavaScript foundation to **infer** types without adding additional syntax.
- As a codebase becomes more complex, it becomes more challenging to determine types from context.
- TypeScript is an extension of JavaScript syntax that allows developers to **explicitly** state the expected type.
- TypeScript also provides additional types beyond the built-in JavaScript types for more complex logic and the ability to create your own types.

```
let message: string =
"Hello, world!";
```

TypeScript Compiler (TSC)

- Browsers and servers do not understand TypeScript syntax. Therefore, TypeScript converts TypeScript syntax into useable code through the TSC.

- TSC uses **transpiling**, which takes unsupported syntax and translates it into clean, supported syntax based on the set target environment (ES5, ES6, ES2020, etc.). It then analyzes (or compiles) the new code to run as expected.

```
// Unsupported syntax in
ES5
const message = () => {
  console.log('Hello,
World!');
}

// After transpiling
into ES5
var message = function()
{
  console.log('Hello,
World!');
}
```

Object-Oriented Programming (OOP)

What is OOP?

OOP is a programming paradigm, or pattern, centered around objects.

In OOP, we solve problems by using collections of objects that work together.

OOP

Objects' ability to communicate with each other makes them well-suited to address complex problems. These concepts are the foundations of the OOP paradigm:

- **Inheritance:** New objects are created based on other objects.
- **Composition:** Objects contain other objects.
- **Polymorphism:** Multiple object types implement the same functionality.
- **Encapsulation:** Object data (and functions) are neatly stored.
- **Abstraction:** Create a simple model of something complex.

TypeScript and OOP

While TypeScript will not enforce any object-oriented principles, TypeScript does provide features that allow us to implement OOP **concepts** into our code in ways that JavaScript alone cannot.

OOP is a broad concept that is best learned through real-life examples.

We begin to see the value of OOP when we use objects to model real-world things in code and provide functionality that would otherwise be hard or impossible.

Instructor **Demonstration**

Mini-Project

The End