**Typescript (TS) [**https://www.typescriptlang.org/**]**

(Net Ninja’s Typescript tutorial: <https://www.youtube.com/playlist?list=PL4cUxeGkcC9gUgr39Q_yD6v-bSyMwKPUI>)

(Traversy Media Typescript tutorial: <https://www.youtube.com/watch?v=BCg4U1FzODs>)

You have seen that JS is **NOT** a statically typed PL. That is, when you start using a variable, the variable’s data type is NOT statically specified unlike statically-typed PLs such as C, C++, Java, C#, etc. Because the variable’s data type is dynamically assigned at run-time, it can easily change. While this choice leads to flexibility, it also makes your code prone to errors. Because the type of a variable is NOT known at compile-time, there is no way for the compiler/interpreter to make static type checking and find errors. That is why, it is usually the case that most errors will pop up during run-time, which may be undesirable.

The idea behind TypeScript (TS) was to add syntax for types to JS so that the language becomes strongly-typed like the popular strongly-typed PLs, which lets the TS compiler find many common errors at the compile time. So, TS is nothing but JS with data types. It also adds some extra features to the language. So, TS is a superset of JS.

You should however know that the JS engine (the browsers) do not understand TS. They only understand JS. This means that after you write your code in TS, you have to use a TS-to-JS compiler to convert your TS code to vanilla JS before you can run it in a browser. Because all TS code is compiled into vanilla JS, all TS features are optional. That is, even if you prefer NOT to use any of the TS features, the TS compiler will NOT complain while converting your TS code to JS code.

You can download and globally install the TS compiler (tsc) into your node modules as follows:

|  |
| --- |
| % npm install -g typescript |

Now, open up a file named 12-TS/01.ts and write your first TS code:

|  |
| --- |
| let firstname: string = "John";  let lastname: string = "Doe";  let id: number = 4;  console.log(firstname, lastname, id)  // types are optional  const gpa = 3.45;  console.log(`My gpa is ${gpa}`) |

We now need to compile this code to js. Open up a terminal and type:

|  |
| --- |
| % tsc 01.ts |

This converts your TS code to JS and creates a file named 12-TS/01.js. Here is the contents of that file:

|  |
| --- |
| var firstname = "John";  var lastname = "Doe";  var id = 4;  console.log(firstname, lastname, id);  // types are optional  var gpa = 3.45;  console.log("My gpa is ".concat(gpa)); |

As you can see, this is plain JS code (ES5) with no types whatsoever. Let and const were introduced in ES6, that is why tsc changes “let” and “const” with “var”. We can now run this code either inside a browser or inside the node.js by typing % node 01.js.

We can also put “tsc” in continuous “watch” mode by specifying the --watch flag so that it automatically does the conversion as 01.js is changed.

**Static Types can NOT be changed**

After you declare the type of a variable, you cannot change its type. The following code will give an error in TS because we first declared “x” to be a number type, but later tried to assign a string to it.

|  |
| --- |
| let x: number = 2;  x = “abc”; // This will throw an error! |

**TS Data Types**

Some of the TS data types are: 12-TS/02.ts

**TS Function Parameter & Return Types**

Look at 12-TS/03.ts

**TS Tuples**

Look at 12-TS/04.ts

**TS Classes**

Look at 12-TS/05.ts

**TS Generic Types**

Look at 12-TS/06.ts

**Zod (**[**https://github.com/colinhacks/zod**](https://github.com/colinhacks/zod)**), (**[**https://zod.dev/**](https://zod.dev/)**) (**[**https://www.totaltypescript.com/tutorials/zod**](https://www.totaltypescript.com/tutorials/zod)**)**

Zod is a TypeScript-first schema declaration and validation library. The goal is to eliminate duplicative type declarations. With Zod, you declare a validator once and Zod will automatically infer the static TypeScript type. It's easy to compose simpler types into complex data structures.

Look at 12-TS/07-zod-test for an example.

Here are two Zod tutorials by WDS:

1. <https://www.youtube.com/watch?v=9UVPk0Ulm6U>
2. <https://www.youtube.com/watch?v=L6BE-U3oy80>