TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Typescript

EXPERIENTIAL
learning factory

What is Typescript?

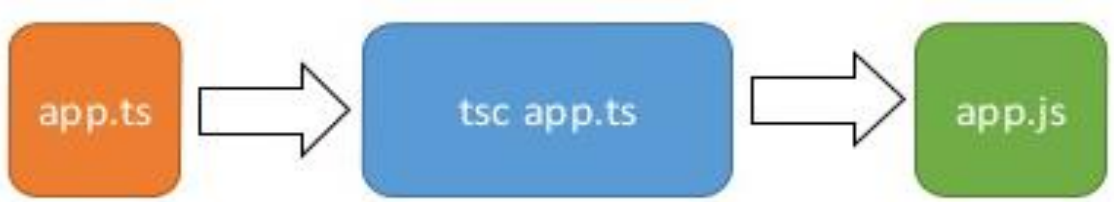
- Typescript is a open source programming language developed and maintained by Microsoft.
- TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
- TypeScript is pure object oriented with classes, interfaces and statically typed like C# or Java.
- The popular JavaScript framework **Angular 2.0** is written in TypeScript.
- TypeScript can help programmers to write object-oriented programs and have them compiled to JavaScript.
- TypeScript is a strongly typed, object oriented, compiled language. It was designed by **Anders Hejlsberg** (designer of C#) at Microsoft.

Why Typescript?

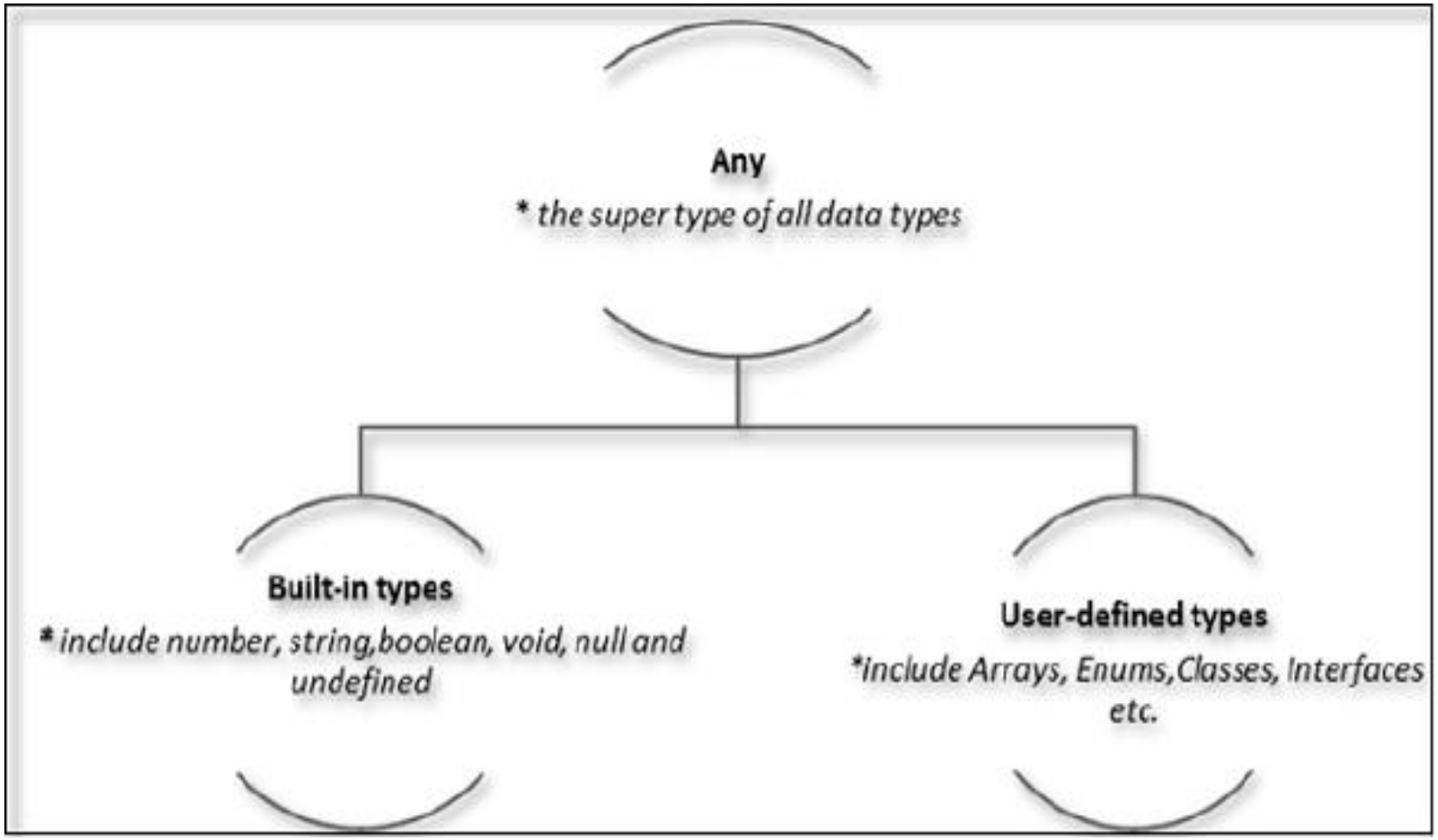
- **Compilation** – JavaScript is an interpreted language. Hence, it needs to be run to test that it is valid. The TypeScript transpiler provides the error-checking feature. TypeScript will compile the code and generate compilation errors, if it finds some sort of syntax errors. This helps to highlight errors before the script is run.
- **Strong Static Typing** – JavaScript is not strongly typed. TypeScript comes with an optional static typing .
- TypeScript **supports Object Oriented Programming** concepts like classes, interfaces, inheritance, etc.

Environment setup

- A Text Editor
  - Visual Studio Code – by Microsoft.
  - WebStrome – by Jetbrains.
  - Brackets.
- Node.JS
  - 10.16.0 LTS
  - Install from https://nodejs.org/en/
- TypeScript Compiler
  - npm install –g typescript (install globally)



Types


Any
* the super type of all data types
Built-in types
* include number, string,boolean, void, null and undefined
User-defined types
*include Arrays, Enums,Classes, Interfaces etc.

Built-in Types

Data type	Keyword	Description
Number	number	It can be used to represent both, integers and fractions.
String	string	Represents a sequence of Unicode characters
Boolean	boolean	Represents logical values, true and false
Void	void	Used on function return types to represent non-returning functions
Null	null	Represents an intentional absence of an object value.
Undefined	undefined	Denotes value given to all uninitialized variables

Null and undefined

- The null and the undefined data types are often a source of confusion. The null and undefined cannot be used to reference the data type of a variable. They can only be assigned as values to a variable.
- However, null and undefined are not the same. A variable initialized with undefined means that the variable has no value or object assigned to it while null means that the variable has been set to an object whose value is undefined.

Variables

- A variable, by definition, is “a named space in the memory” that stores values.
- In other words, it acts as a container for values in a program.
- TypeScript variables must follow the JavaScript naming rules.
- Variable names can contain alphabets and numeric digits.
- They cannot contain spaces and special characters, except the underscore (_) and the dollar ($) sign.
- Variable names cannot begin with a digit.
- A variable must be declared before it is used.

Variable Declaration in TypeScript

When you declare a variable, you have four options −

Declare its type and value in one statement.

Declare its type but no value. In this case, the variable will be set to undefined.

Declare its value but no type. The variable type will be set to the data type of the assigned value.

Declare neither value not type. In this case, the data type of the variable will be any and will be initialized to undefined.

Variable declaration and syntax

S.No.	Variable Declaration Syntax & Description
1.	**var name:string = ”mary”** The variable stores a value of type string
2.	**var name:string;** The variable is a string variable. The variable’s value is set to undefined by default
3.	**var name = ”mary”** The variable’s type is inferred from the data type of the value. Here, the variable is of the type string
4.	**var name;** The variable’s data type is any. Its value is set to undefined by default.

User-defined Types

- **Enumerations (enums)**

  Enums allow us to declare a set of named constants i.e. a collection of related values that can be numeric or string values.

- **Classes**

  A class in terms of OOP is a blueprint for creating objects. A class encapsulates data for the object.

- **Interfaces**

  Interfaces contain only the declaration of the members. It is the responsibility of the deriving class to define the members.

- **Arrays**

  An array is a special type of data type which can store multiple values of different data types

- **Tuple**

  A tuple type variable can include multiple data types

Generics

Generics offer a way to create reusable components. Generics provide a way to make components work with any data type and not restrict to one data type. So, components can be called or used with a variety of data types.

Example:

```
function getArray<T>(items : T[] ) : T[] {
        return new Array<T>().concat(items);
}
let myNumArr = getArray<number>([100, 200, 300]);
let myStrArr = getArray<string>(["Hello", "World"]);
myNumArr.push(400);      // OK
myStrArr.push("Hello TypeScript");          // OK
myNumArr.push("Hi");      // Compiler Error
myStrArr.push(500);       // Compiler Error
```

Namespace

- The namespace is used for logical grouping of functionalities.
- A namespace can include interfaces, classes, functions and variables to support a single or a group of related functionalities.
- A namespace can be created using the namespace keyword followed by the namespace name.
- All the interfaces, classes etc. can be defined in the curly brackets { }.

Syntax:

```
namespace <name> {

}
```

Modules

- TypeScript provides modules and namespaces in order to prevent the default global scope of the code and also to organize and maintain a large code base.
- Modules are a way to create a local scope in the file. So, all variables, classes, functions, etc. that are declared in a module are not accessible outside the module.
- A module can be created using the keyword export and a module can be used in another module using the keyword import.
- In TypeScript, files containing a top-level export or import are considered modules.

Decorators

- A Decorator is a special kind of declaration that can be applied to classes, methods, accessor, property, or parameter. Decorators are simply functions that are prefixed **@expression** symbol, where expression must evaluate to a function that will be called at runtime with information about the decorated declaration.
- In tsconfig.json

```
{
        "compilerOptions": {
        "target": "ES5",
        "experimentalDecorators": true
        }
}
```

TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Thank You !!!

No.01, 3rd Cross Basappa Layout, Gavipuram Extension,
Kempegowda Nagar, Bengaluru, Karnataka 560019

praveen.d@testyantra.com

www.testyantra.com

EXPERIENTIAL
learning factory