Angular 8 : Online Class

17-August-2019

By: Sahosoft Solutions

Presented by : Chandan Kumar

Angular 8 : Online Class

TypeScript

By: Sahosoft Solutions

Presented by : Chandan Kumar

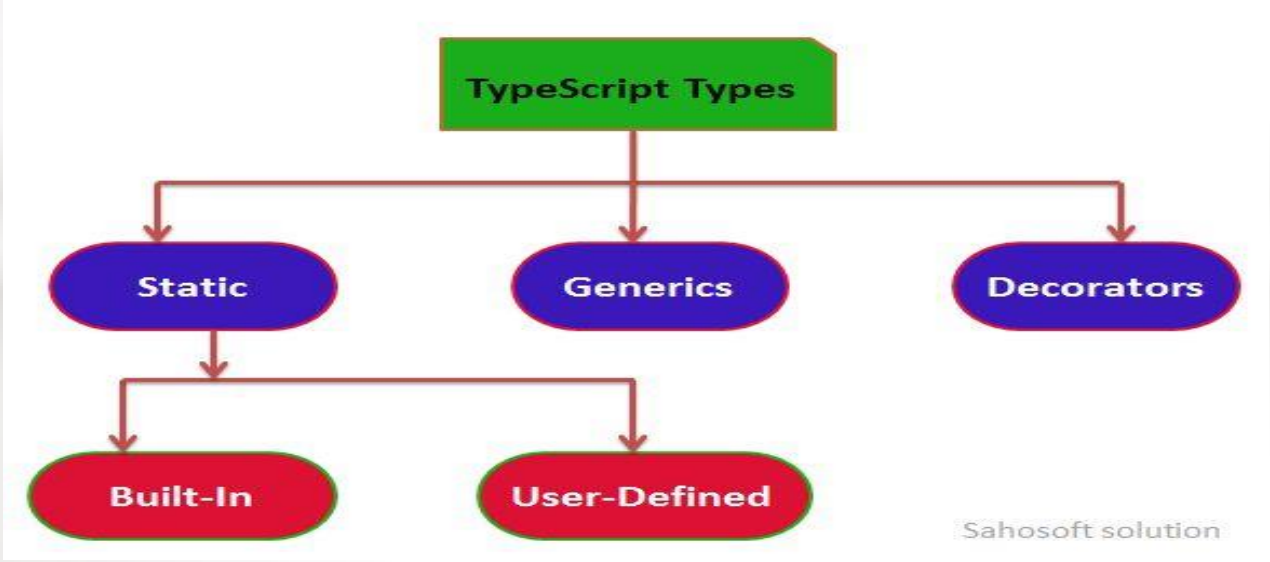
Typescript Data Type

The TypeScript language supports different types of values.

It provides data types for the JavaScript to transform it into a strongly typed programing language.
JavaScript doesn't support data types, but with the help of TypeScript, we can use the data types feature in JavaScript.

TypeScript plays an important role when the object-oriented programmer wants to use the type feature in any scripting language or object-oriented programming language.
The Type System checks the validity of the given values before the program uses them. It ensures that the code behaves as expected.

TypeScript provides data types as an optional Type System. We can classify the TypeScript data type as following.



Typescript Data Type

Static Types

In the context of type systems, static types mean "at compile time" or "without running a program."

In a statically typed language, variables, parameters, and objects have types that the compiler knows at compile time. The compiler used this information to perform the type checking.

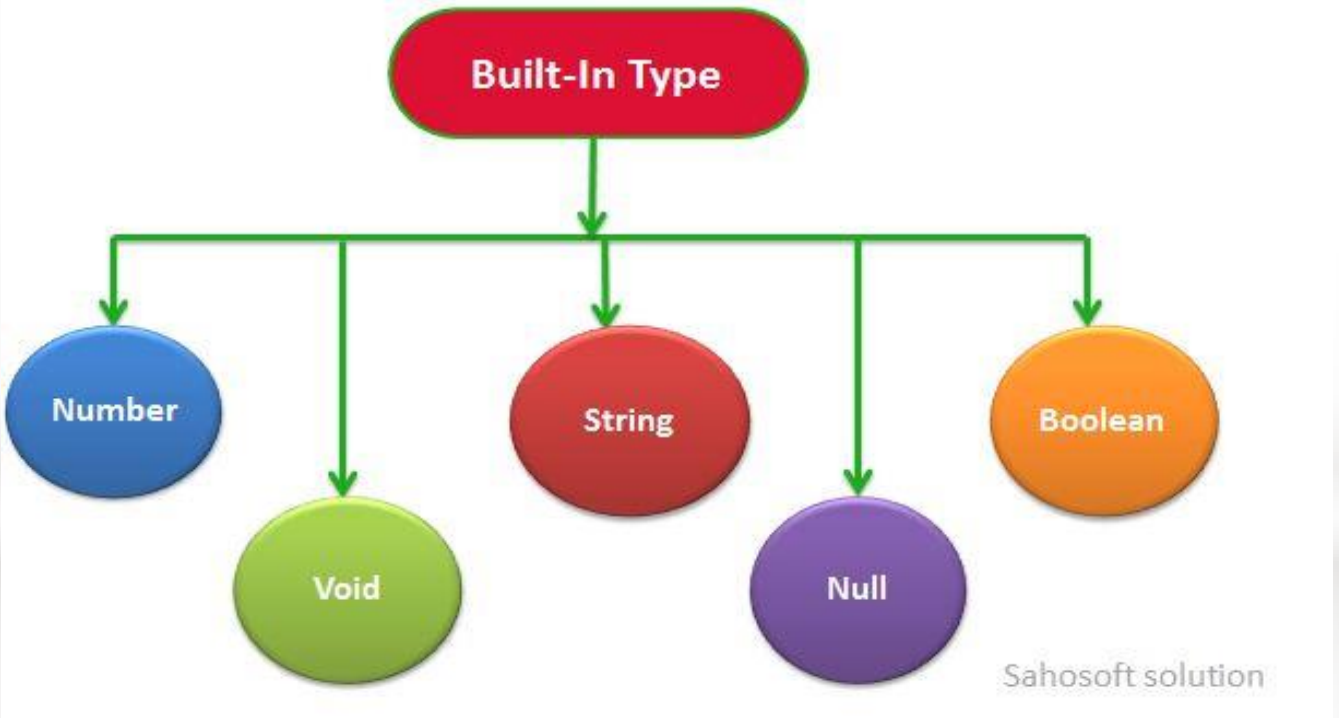
Static types can be further divided into two sub-categories:

1. **Built-in or Primitive Type**
2. **User-Defined DataType**

Typescript Data Type

Built-in or Primitive Type

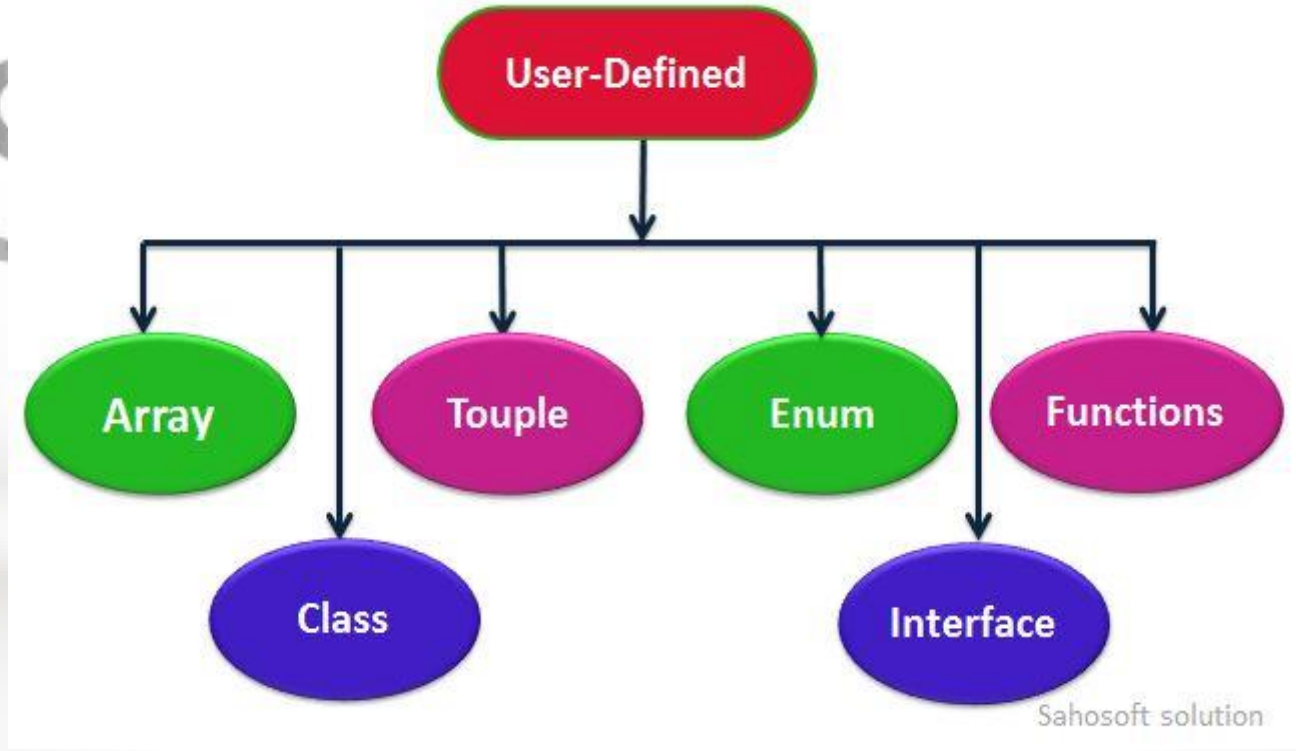
The TypeScript has five built-in data types, which are given below.



Typescript Data Type

User-Defined DataType

TypeScript supports the following user-defined data types:



Built-in or Primitive Type

Number

Like JavaScript, all the numbers in TypeScript are stored as floating-point values.
These numeric values are treated like a number data type. The numeric data type can be used to represents both integers and fractions.

Syntax:

```
let identifier: number = value;
```


String

We will use the string data type to represents the text in TypeScript. String type work with textual data. We include string literals in our scripts by enclosing them in single or double quotation marks. It also represents a sequence of Unicode characters. It embedded the expressions in the form of **$ {expr}**.

Syntax

```
let identifier: string = " ";
                Or
let identifier: string = ' ';
```

Built-in or Primitive Type

Boolean

The string and numeric data types can have an unlimited number of different values, whereas the Boolean data type can have only two values. They are "true" and "false." A Boolean value is a truth value which specifies whether the condition is true or not.

Syntax

```
let identifier: boolean = Boolean value;
```

Void

A void is a return type of the functions which do not return any type of value. It is used where no data type is available.

A variable of type void is not useful because we can only assign undefined or null to them.

Syntax

```
let unusable: void = undefined;
```

Built-in or Primitive Type

Null

Null represents a variable whose value is undefined.
Much like the void, it is not extremely useful on its own.
The Null accepts the only one value, which is null. The Null keyword is used to define the Null type in TypeScript, but it is not useful because we can only assign a null value to it.

Undefined

The Undefined primitive type denotes all uninitialized variables in TypeScript and JavaScript. It has only one value, which is undefined. The undefined keyword defines the undefined type in TypeScript, but it is not useful because we can only assign an undefined value to it.

Built-in or Primitive Type

Any Type

It is the "super type" of all data type in TypeScript. It is used to represents any JavaScript value. It allows us to opt-in and opt-out of type-checking during compilation. If a variable cannot be represented in any of the basic data types, then it can be declared using "**Any**" data type. Any type is useful when we do not know about the type of value (which might come from an API or 3rd party library), and we want to skip the type-checking on compile time.

Syntax

```
let identifier: any = value;
```

User-Defined DataType

Array

An array is a collection of elements of the same data type. Like JavaScript, TypeScript also allows us to work with arrays of values. An array can be written in two ways:

1. Use the type of the elements followed by [] to denote an array of that element type:

```
var list : number[] = [1, 3, 5];
```

2. The second way uses a generic array type:

```
var list : Array<number> = [1, 3, 5];
```


Touple

The Tuple is a data type which includes sets of values of different data types.
It allows us to express an array where the type of a fixed number of elements is known, but they are not the same.

For example, if we want to represent a value as a pair of a number and a string.

User-Defined DataType

Interface

An Interface is a structure which acts as a contract in our application. It defines the syntax for classes to follow, means a class which implements an interface is bound to implement all its members. It cannot be instantiated but can be referenced by the class which implements it.

Class

Classes are used to create reusable components and acts as a template for creating objects.
It is a logical entity which store variables and functions to perform operations.
TypeScript gets support for classes from ES6. It is different from the interface which has an implementation inside it, whereas an interface does not have any implementation inside it.

Enums

Enums define a set of named constant. TypeScript provides both string-based and numeric-based enums.
By default, enums begin numbering their elements starting from 0, but we can also change this by manually setting the value to one of its elements.
TypeScript gets support for enums from ES6.

User-Defined DataType

Functions

A function is the logical blocks of code to organize the program. Like JavaScript, TypeScript can also be used to create functions either as a **named function** or as an **anonymous function**. Functions ensure that our program is readable, maintainable, and reusable. A function declaration has a function's name, return type, and parameters.

Generic DataType

Generic

Generic is used to create a component which can work with a variety of data type rather than a single one. It allows a way to create reusable components. It ensures that the program is flexible as well as scalable in the long term.

TypeScript uses generics with the type variable <T> that denotes types. The type of generic functions is just like non-generic functions, with the type parameters listed first, similarly to function declarations.

Decorators

A decorator is a special of data type which can be attached to a class declaration, method, property, accessor, and parameter. It provides a way to add both annotations and a meta-programing syntax for classes and functions. It is used with "@" symbol.

Difference between Null and Undefined

Null

Null is used to represent an intentional absence of value. It represents a variable whose value is undefined. It accepts only one value, which is null. The Null keyword is used to define the Null type in TypeScript, but it is not useful because we can only assign a null value to it.

Undefined

It represents uninitialized variables in TypeScript and JavaScript. It has only one value, which is undefined. The undefined keyword defines the undefined type in TypeScript, but it is not useful because we can only assign an undefined value to it.

Null vs. Undefined

The important difference between Null and Undefined are

Difference between Null and Undefined

Null vs. Undefined

The important difference between Null and Undefined are

SN	Null	Undefined
1	It is an assignment value. It can be assigned to a variable which indicates that a variable does not point any object.	It is not an assignment value. It means a variable has been declared but has not yet been assigned a value.
2	It is an object.	It is a type itself.
3	The null value is a primitive value which represents the null, empty, or non-existent reference.	The undefined value is a primitive value, which is used when a variable has not been assigned a value.
4	Null indicates the absence of a value for a variable.	Undefined indicates the absence of the variable itself.