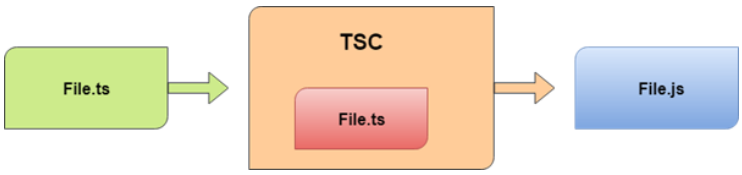
Typescript:

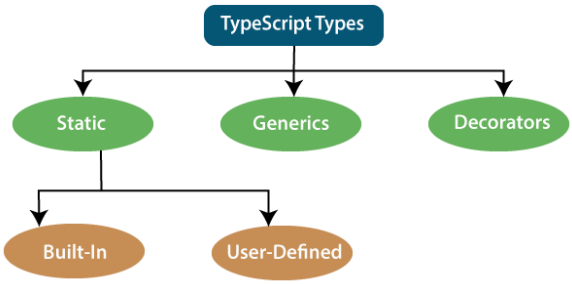
TypeScript is an open-source pure object-oriented programing language. It is a strongly typed superset of JavaScript which compiles to plain JavaScript. It contains all elements of the JavaScript. It is a language designed for large-scale JavaScript application development.

→TypeScript cannot run directly on the browser. It needs a compiler to compile the file and generate it in JavaScript file, which can run directly on the browser. The TypeScript source file is in ".ts" extension. We can use any valid ".js" file by renaming it to ".ts" file.

→TypeScript uses TSC (TypeScript Compiler) compiler, which convert Typescript code (.ts file) to JavaScript (.js file).



Typescript types:

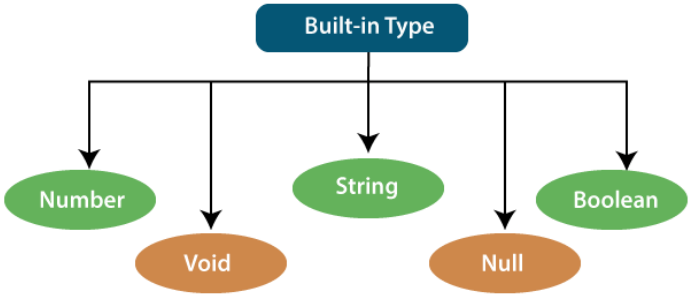


Points to know:

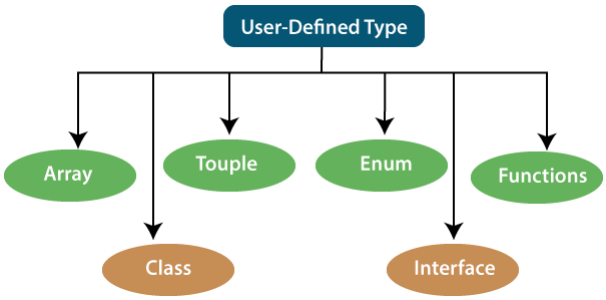
1) What is static type?

Ans) 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.

Built in type:



User defined type



Null VS Undefined:

	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.
5.	Null is converted to zero (0) while performing primitive operations.	Undefined is converted to NaN while performing primitive operations.

Typescript Variable:

Using var Keyword: var i=40;
var i:number=30;

Using let Keyword: let i=40;
let i:number=30;

→ A variable declared with **let** keyword is not hoisted. If we try to use a let variable before it is declared, then it will result in a **ReferenceError**.

Using const Keyword: const i=40;
const i:number=30;

Typescript operator:

1) Arithmetic operators (+ , - , / , % , ++ , --)
2) Comparison (Relational) operators(== , === , != ,>= ,<= , < , >)
3) Logical operators (&&, || , !)
4) Bitwise operators (& , | ,^ , > ,< , >> , << , >>>)
5) Assignment operators (= , += , -= , *= , %= , /=)
6) Ternary/conditional operator (expression ? expression-1 : expression-2;)
7) Type Operator (instance of , in , delete , instance)

Type Aseertion:

In TypeScript, type assertion is a mechanism which tells the compiler about the type of a variable. When TypeScript determines that the assignment is invalid, then we have an option to override the type using a type assertion. If we use a type assertion, the assignment is always valid, so we need to be sure that we are right. Otherwise, our program may not work correctly.

TypeScript provides two ways to do Type Assertion. They are

→Using Angular Bracket <>

```
let empCode: any = 111;
let employeeCode = <number> code;
```

→Using as keyword

```
let empCode: any = 111;
let employeeCode = code as number;
```

Typescript array:

→An array is a homogenous collection of similar type of elements which have a contiguous memory location.
→An array is a user-defined data type.
→An array is a type of data structure where we store the elements of a similar data type. In an array, we can store only a fixed set of elements. We can also use it as an object.

There are two types of an array:

→Single-Dimensional Array

```
array_name = [val1,val2,valn..]
```

→Multi-Dimensional Array

```
let arr_name:datatype[initial_array_index][referenced_array_index]
                                        = [ [val1,val2,val 3], [v1,v2,v3]];
```

TypeScript Tuples

We know that an array holds multiple values of the same data type. But sometimes, we may need to store a collection of values of different data types in a single variable. Arrays will not provide this feature, but TypeScript has a data type called Tuple to achieve this purpose.

Syntax:

```
let tuple_name = [val1,val2,val3, ...val n];
```

A tuple has two operations:

→Push()

```
let empTuple = ["Rohit Sharma", 25, "JavaTpoint"];
empTuple.push(10001);
console.log("Items: "+empTuple);
```

→Pop()

```
let empTuple = ["Rohit Sharma", 25, "JavaTpoint", 10001];
empTuple.pop();
console.log("Items: "+empTuple);
```

TypeScript Union:

In TypeScript, we can define a variable which can have multiple types of values. In other words, TypeScript can combine one or two different types of data (i.e., number, string, etc.) in a single type, which is called a union type. Union types are a powerful way to express a variable with multiple types. Two or more data types can be combined by using the pipe ('|') symbol between the types.

Syntax:

```
(type1 | type2 | type3 | ..... | type-n)
```

Example

```
let value: number|string;
value = 120;
console.log("The Numeric value of a value is: "+value);
value = "Welcome to JavaTpoint";
console.log("The String value of a value is: "+value);
```

TypeScript String:

In TypeScript, the string is an object which represents the sequence of character values. It is a primitive data type which is used to store text data. The string values are surrounded by single quotation mark or double quotation mark. An array of characters works the same as a string.

Syntax:

```
let var_name = new String(string);
var studentName: String = 'Peter';  // single quoted
var studentName: String = “Peter”;//double quoted
```

String methods:

	Method	Description
1.	charAt()	It returns the character of the given index.
2.	concat()	It returns the combined result of two or more string.
3.	endsWith()	It is used to check whether a string ends with another string.
4.	includes()	It checks whether the string contains another string or not.
5.	indexOf()	It returns the index of the first occurrence of the specified substring
6.	lastIndexOf()	It returns the index of the last occurrence of a value in the string.
7.	match()	It is used to match a regular expression against the given string.
8.	replace()	It replaces the matched substring with the new substring.
9.	search()	It searches for a match between a regular expression and string.
10.	slice()	It returns a section of a string.
11.	split()	It splits the string into substrings and returns an array.
12.	substring()	It returns a string between the two given indexes.
13.	toLowerCase()	It converts the all characters of a string into lower case.
14.	toUpperCase()	It converts the all characters of a string into upper case.
15.	trim()	It is used to trims the white space from the beginning and end of the string.
16.	trimLeft()	It is used to trims the white space from the left side of the string.
17.	trimRight()	It is used to trims the white space from the right side of the string.
18.	valueOf()	It returns a primitive value of the specified object.