# TYPE SCRIPT PROGRAMMING LANGUAGE

**INTRODUCTION:**

* TypeScript is an open-source object-oriented language developed and maintained by Microsoft.
* TypeScript is a superset of JavaScript that compiles to plain JavaScript. Basically, TypeScript is the ES6 version of JavaScript with some additional features. ES6 stands for ECMAScript 6.
* ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript **(European Computer Manufacturers Association Script)**
* Before TypeScript JavaScript was well established in the market.
* Using JS, we can develop both client and server-side applications using different frameworks like Angular or React.js (Client-side application framework), Node.js (Server-side application framework)
* JavaScript is a dynamic programming language with no type of system.
* We need a language with type system which increases the code quality, readability and makes it, an easy to maintain and refactor code base.
* More importantly, errors can be identified at compile time rather than at run time.
* So, without this type of system, it is difficult to scale JavaScript to build complex applications.
* Hence, the reason to use TypeScript is that it supports type system and allows JavaScript to be used at scale to build complex applications.
* A TypeScript code is written in a file with .ts extension and then compiled into JavaScript using theTypeScript compiler.
* A TypeScript file can be written in any code editor.
* A TypeScript compiler needs to be installed on your platform. Once installed, the command tsc <filename>.ts compiles the TypeScript code into a plain JavaScript file.
* JavaScript files can then be included in the HTML and run on any browser.

**Features of TypeScript:**

* **Cross-Platform**: The TypeScript compiler can be installed on any Operating System such as Windows, MacOS and Linux.
* **Object-Oriented Language:** TypeScript follows the principles and practices of OOP’s (features like Classes, Objects, Interfaces, Inheritance and Modules)
* **Static type-checking:** TypeScript uses static typing and helps type checking at compile time. Thus, you can find errors while writing the code without running the script.
* **Optional Static Typing:** TypeScript also allows optional static typing in case you are using the dynamic typing of JavaScript.
* **DOM Manipulation**: You can use TypeScript to manipulate the DOM for adding or removing elements. (Document Object Model)
* **ES6 Features:** TypeScript includes most features of planned ECMAScript 2015 (ES 6, 7) such as class, interface, Arrow functions etc.

**Advantages of TypeScript:**

* TypeScript is an open-source language with continuous development and maintenance by Microsoft.
* TypeScript runs on any browser or JavaScript engine.
* TypeScript is like JavaScript and uses the same syntax and semantics. All of TypeScript's code finally gets converted into JavaScript.
* TypeScript is also closer in syntax to back-end languages like Java. This helps backend developers to write front-end code faster.
* TypeScript code can be called from an existing JavaScript code. TypeScript also works with existing.
* JavaScript frameworks and libraries without any issues.
* TypeScript has support for the latest JavaScript features.

**Differences btw Typescript and JavaScript:**

|  |  |
| --- | --- |
| **JavaScript** | **Typescript** |
| * JavaScript is a scripting language which helps you create interactive web pages. * JavaScript code doesn't need to be compiled. * There is no static typing.   Ex: var num   * It does not have interfaces. * It has no optional parameter feature. * It has no Rest Parameter feature. * Generics not supported. * Modules not supported. * number, string are the objects here.   **------------------------------------------** | * Typescript is Object oriented Programming language & superset of JavaScript. * Typescript code needs to be compiled. * There is static typing. We can declare variables with data types. Ex: var num: number * It has an interface. * It has an optional parameter feature. * It has a Rest Parameter feature. * Generics supported. * Modules supported. * Number, string are the interfaces |

**Environment setup for Typescript:**

The pre-requisites are:

* Nodejs
* Typescript
* Visual Studio code (Environment IDE)

1. First install the NodeJS and along with NodeJS the npm will also get installed where the term npm is known as Node Package Manager.
2. Install Typescript:

--- npm install -g typescript

1. To check the version of the typescript: tsc -v
2. Install the IDE Visual studio Code and Create TypeScript project in Visual Studio Code
3. Create the filename under the project with .ts extension.
4. Its mandatory to datatype of the variables in Typescript:
5. Create the function.
6. Invoke the function.
7. Compile time commands:
8. tsc Samp.ts (Convert into JS from TS)
9. node Samp.js (Run JS program in console)

**Variables in Typescript:**

1. Variables are used to store data.
2. TypeScript follows the same rules as JavaScript for variable declarations.
3. JavaScript is not a typed language. It means we cannot specify the type of a variable such as number, string, Boolean etc.
4. However, TypeScript is a typed language, where we can specify the type of variables.
5. In typescript variables can be used and declared as: var, let, and const.
6. We can declare the variables in four different ways.
7. both type and initial value
8. only the type
9. only the initial value
10. without type and initial value
11. **Var Keyword:**
12. The variables declared using var are available within the function.
13. If we declare them outside the function, then they are available everywhere i.e., they are a global variable.
14. If we define them inside a code block, they are still scoped to the enclosing function.
15. It is redefined or updated.
16. **Let Keyword:**
17. The variables declared using **let** can only be used in the code block where we declare them. Outside the code block, they are invisible.
18. If they are outside the code block, but within the function body then they become function scoped.
19. **Const Keyword:**
20. We must declare a const variable with an initial value.
21. The value cannot be reassigned again.
22. The keyword 'const' makes the variable a constant, which means the variable's value cannot be changed.

**OPERATORS:**

* Operator is a symbol which will perform certain operation.
* TypeScript consists of different types of operators that are used to perform different operations. They are:

1. Arithmetic Operators
2. Assignment Operators
3. Relational/Comparison Operators
4. Logical Operator

* **Arithmetic Operators:** The Typescript arithmetic operators take numerical values as their left & right operands, perform the arithmetic operation, and return a numerical value. They are: **+, -, \*, /, %,++,--**
* **Assignment Operators:**  Assignment operators are used to assign values to variables. They are:

|  |  |  |
| --- | --- | --- |
| * **Operator** | **Description** | **Example** |
| = (Simple Assignment) | Assigns values from the right-side operand to the left side operand | C = A + B will assign the value of A + B into C |
| += (Add and Assignment) | It adds the right operand to the left operand and assigns the result to the left operand. | C += A is equivalent to C = C + A |
| -= (Subtract and Assignment) | It subtracts the right operand from the left operand and assigns the result to the left operand. | C -= A is equivalent to C = C - A |
| \*= (Multiply and Assignment) | It multiplies the right operand with the left operand and assigns the result to the left operand. | C \*= A is equivalent to C = C \* A |
| /= (Divide and Assignment) | It divides the left operand with the right operand and assigns the result to the left operand. |  |

* **Relational Operators:** Relational Operators test or define the kind of relationship between two entities. Relational operators return a Boolean value, i.e., true/ false.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| > | Greater than | (A > B) is False |
| < | Lesser than | (A < B) is True |
| >= | Greater than or equal to | (A >= B) is False |
| <= | Lesser than or equal to | (A <= B) is True |
| == | Equality | (A == B) is True |
| != | Not equal | (A != B) is True |

**Logical operators:** Logical Operators are used to combine two or more conditions. Logical operators too return a Boolean value. Assume the value of variable A is 10 and B is 20.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && (And) | The operator returns true only if all the expressions specified return true | (A > 10 && B > 10) is False |
| || (OR) | The operator returns true if at least one of the expressions specified return true | (A > 10 || B >10) is True |
| ! (NOT) | The operator returns the inverse of the expression’s result. For E.g.: !(>5) returns false | !(A >10 ) is True |

* **Strings:** 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.

1. charAt() : charAt() is a method that returns the character from the specified index. Characters in a string are indexed from left to right.
2. Concat(): It adds the strings more than two or two.
3. Replace (): It replaces the values in the string.
4. Split (): This method **splits** a String object into an array of strings by separating the string into substrings.
5. Substrings (): The **substring**() method extracts characters, between two indices (positions), from a string, and returns the **substring**.
6. toLowercase(): Changes the upper case to lowercase letters.
7. toUppercase(): Changes the lower to uppercase letters of a string.

**Conditional Statements:**

* **If**
* **If…...else**
* **else if**
* **Ternary operator**
* **Switch**

**If:** If the Boolean expression evaluates to true, then the block of code inside the if statement will be executed. If the Boolean expression evaluates to false, then the first set of code after the end of the if statement (after the closing curly brace) will be executed.

Syntax:

**If……else:** An ‘if’ statement can be followed by an optional ‘else’ statement, which executes when the Boolean expression is false.

Syntax: if (*condition*) {  
  } else {  
  //  block of code to be executed if the condition is false}

**Else if:** You can use one ‘if’ or ‘else if’ statement inside another ‘if’ or ‘else if’ statement(s).

**Switch:** A ‘switch’ statement allows a variable to be tested against a list of values.

**Looping Statements:**

A loop statement allows us to execute a statement or group of statements multiple times.

* **For**
* **While**
* **Do…...while**
* **Jumping statements: Switch and continue.**

**For:** The **for** loop executes the code block for a specified number of times. It can be used to iterate over a fixed set of values, such as an array.

1. If condition is true then the for loop executes
2. Syntax: **for (Initialization; condition; inc/dec)**

**{**

**Statements ;**

**{**

1. Firstly, the initialization starts and then the if the condition is true then it executes the statements then it will go to increment part.

**While:** The while loop is another type of loop that checks for a specified condition before beginning to execute the block of statements. The loop runs until the condition value is met.

1. In while loop initialization say where to start and the condition says where to stop.
2. **Syntax: while(condition)**

**{**

**Block of code to be executed.**

**}**

**Do-While:** The do..while loop is similar to the while loop, except that the condition is given at the end of the loop. The do..while loop runs the block of code at least once before checking for the specified condition.

do {

// code block to be executed

}

while (condition expression);

**Arrays:** An array is a special type of data type which can store multiple values of different data types sequentially using a special syntax. There are 2 types of arrays:

1. Single dimensional
2. Two Dimensional

**Tuple:** Tuple is a new data type which includes multiple set of values of different data types.

For ex: var employee = [1427, Pranathi]

Tuple is used to create only one variable at a time instead of multiple variables.

* For adding of elements into tuple we can use push() method
* For removing the elements from tuple we can use pop() method

**Functions:** A functions are the primary blocks of any program.

**Advantages of function:**

•**Code reusability**: We can call a function several times without writing the same block of code again. The code reusability saves time and reduces the program size.

**Less coding:** Functions makes our program compact. So, we don't need to write many lines of code each time to perform a common task.

**Easy to debug:** It makes the programmer easy to locate and isolate faulty information.

There are two types of functions are there:

1. Named Function
2. Anonymous Function

**Named Function:**  A named fun

**Class and Objects:** A class can include the following:

- Constructor

- Properties

- Methods

Once a class is created, we can create number of objects, where as Object is a physical entity.

* We can initialize the variables by method.

Basic Program for classes and objects:

class Person {  
  name: string;  
}  
  
const person = new Person ();  
person.name = "Pranathi ";

Class members also be given special modifiers which affect visibility:

There are three main visibility modifiers in TypeScript.

* public - (default) allows access to the class member from anywhere.
* private - only allows access to the class member from within the class.
* protected - allows access to the class member from itself and any classes that inherit it, which is covered in the inheritance section below.

We can consider these as a Members Visibility in a class.

**Inheritance**: Inheritance is acquiring all the variables and methods from one class to another class.

• It helps to reuse the code and establish a relationship between different classes.

1. Parent class (Super or Base class)
2. Child class (Subclass or Derived class)

• A class which inherits the properties is known as Child Class whereas a class whose properties are inherited is known as Parent class.

• TypeScript classes can be extended to create new classes with inheritance, using the keyword extends.

**Basic Syntax:**

Class A

{

* - ---

}

Class B extends A

{

* - - -

}

**Method Overriding:**

* Method Overriding is the process in which a method belonging to the base (or parent) class is overridden by the same method (same method and signature) of the derived (child) class.
* In this process, a child (derived) class method may or may not use the logic defined in the parent (base) class method.
* In order to invoke the methods or properties of the base class, we may use the **super** keyword which would help us to invoke that particular method or property of the base class into the child class.
* Method Overriding is useful whenever we want to alter the behaviour of any method of the parent class in child class.

**Interface:**

* A Interface is a structure that defines the contract in your application. It defines the syntax that classes to be follow.
* Interface contains variables and methods.
* It also follows the same syntax of the class.
* We don’t have any implementations in the interfaces.
* If we use a class, we use class keyword and if we use, we use Interface keyword.
* The TypeScript compiler does not convert interface to JavaScript. It uses interface for type checking. This is also known as "duck typing" or "structural subtyping".
* An interface is defined with the keyword interface, and it can include properties and method declarations using a function or an arrow function.
* A class implements interface where a class can be implemented. Also interface alone cannot be used but it is used along with the class that what we have created the methods in that class.

**Modules:**

A module is a way to create a group of related variables, functions, classes, and interfaces, etc. It executes in the **local scope**, not in the **global scope**. In other words, the variables, functions, classes, and interfaces declared in a module cannot be accessible outside the module directly. We can create a module by using the **export** keyword and can use in other modules by using the **import** keyword.

There are 2 types of modules are divided:

1. **Internal Module:** Internal modules were in the **earlier version** of Typescript. It was used for **logical grouping** of the classes, interfaces, functions, variables into a single unit and can be exported in another module. The modules are named as a **namespace** in the latest version of TypeScript.

**Namespace** is used identify and refer to objects of various kinds. A namespace ensures that all of a given set of objects have unique names so that they can be easily identified.External Module

1. **External Module:** External modules are also known as a **module**. When the applications consisting of hundreds of files, then it is almost impossible to handle these files without a modular approach. External Module is used to specify the **load dependencies** between the multiple external js files.

We declare the module by using “**Export**” keyword. We can use the declare module in other files by using an “**import”** keyword.

**Typescript Enum:**

1. Enums are a new data type supported in TypeScript.
2. Enums stands for **Enumerations**.
3. It is used to define the set of **named constants**, i.e., a collection of related values.
4. Enumerations are declared by keyword “ENUM”.
5. Typescript supports both numeric and string based enums.
6. It makes it easy to change values in the future.
7. It reduces errors that is caused by mistyping a number.
8. It exists only during compilation time, so it does not allocate memory.
9. It saves runtime and compile-time with inline code in JavaScript.
10. It allows us to create constants that we can easily relate to the program.
11. Enums are of three types. They are:
12. Numeric enum: That accepts only numeric values.
13. String based enums: That accepts the string-based values.
14. Heterogenous enums: It is a combination of both Strings and Numeric values.

**TYPE CASTING:** As we know typecasting is nothing but the conversion of one type to another type.

* Here JS doesn’t support this type casting concept because here variables are having dynamic values.
* But TS supports Type casting concept as it is having types.
* Here Typecasting allows us to convert on variable type into another.
* Casting can be declared with “as” keyword or “< >” operators.

**For Each () Method:** The forEach() method is an array method which is used to execute a function on **each item in an array**. We can use it with the JavaScript data types like Arrays, Maps, Sets, etc. It is a useful method for displaying elements in an array.

### **What are anonymous functions? Provide their syntax in TypeScript.**

An anonymous function is a function without a name. Anonymous functions are typically used as callback functions, i.e., they are passed around to other functions, only to be invoked by the other function at a later point in time. For example,

setTimeout(function () {

console.log('Run after 2 seconds')

}, 2000);

You can invoke an anonymous function as soon as it’s created. It’s called ‘immediately invoked function execution (IIFE)’, For example:

(function() {

console.log('Invoked immediately after creation');

})()