# **Learning TypeScript**

## ****1. Installation & Basic Commands****

**Install TypeScript globally**  
To install TypeScript globally on your system so you can use tsc from any directory:

npm i typescript -g

**Initialize TypeScript configuration**  
To create a tsconfig.json file in your project, which stores compiler options and settings:

tsc --init

**Compile a TypeScript file**  
To compile your app.ts file into app.js:

tsc app.ts

**Enable watch mode**  
To watch TypeScript files for changes and automatically recompile them:

tsc --watch

## ****2. Type Inference****

**Definition:**  
When TypeScript automatically infers the type of a variable or function based on the assigned value without explicitly mentioning the type.

**Example:**  
If you write:

* let a = 10; → TypeScript infers a as a number.
* let str = "harsh"; → TypeScript infers str as a string.

Here, you are not explicitly defining the type, but TypeScript understands it based on the value assigned.

## ****3. Type Annotation****

**Definition:**  
When you explicitly mention the type of a variable, parameter, or function return type.

**Example:**

* let a : number = 10;
* let str : string = "harsh";

Here, you are telling TypeScript the exact type of each variable.

## ****4. Using Type Annotation in Functions****

**Example:**  
If you have a function that adds two numbers:

Function definition:  
You specify the type of parameters as number and also mention the return type as number.

For example:

* function add(a : number, b : number) : number { return a + b; }

Here:

* a : number → parameter type annotation
* b : number → parameter type annotation
* : number after parentheses → function return type annotation

**Void Return Type Example:**  
If a function does not return anything:

* function greet(name : string) : void { console.log("Hello, " + name); }

## ****5. Type Aliases****

**Definition:**  
Type aliases allow you to create a custom name for a type. It improves code readability and makes complex or union types easier to reuse.

**Example:**  
If you want a variable to store either string or number:

* type StringOrNumber = string | number;
* let id : StringOrNumber = 123;
* id = "ABC123"; // also valid

**Example with function using type alias:**  
If you have a User type:

* type User = { name : string; age : number; }

And a function using it:

* function printUser(user : User) : void { console.log("Name: " + user.name + ", Age: " + user.age); }

**6. Interface**

Definition:  
An interface is used to define the structure of an object. It enforces that an object must have specific properties with their types.

Example:  
If you want to define a user object with name and age:

• interface User {  
  name : string;  
  age : number;  
 }

Using interface in a function:

• function printUser(obj : User) {  
  console.log("name is " + obj.name + " and age is " + obj.age);  
 }

Correct usage:

• printUser({ name : "harsh", age : 22 });

Wrong usage:

• printUser({ name : "harsh", age : "22" }); // age should be number, not string.

Day 1 end