TypeScript will infer the data-type:

var aString = "a traditional var";

let bString = "using let keyword";

const birthday = "12-12-1960";

Explicitly specified data types:

var aString : string = "a traditional var";

let bString : string = "using let keyword";

const birthday : string = "12-12-1960";

The let and const keywords have been available since the release of TypeScript 1.4.

Constant

A data item is constant when its value cannot be changed while a program is running. So, If a variable has been marked as const, then its value can only be set when the variable is defined, and cannot be changed afterwards.

const birthdate : string = "12-12-1960";

//compile error

birthdate = "11-11-1960";

We’ve defined a constant birthdate and it cannot be changed as it has const keyword. Attempting to compile this code will result compile error.

Variable

A variable can hold only one value at a time, but the value it holds can change. For example, if you create a variable named speed, it might hold 0 when the application starts, later be altered to hold 120, and still later be altered to hold 150.

Variables in JavaScript are defined by using the keyword var. TypeScript introduces the letkeyword, which can be used in the place of the var keyword when defining a variable.

var vs. let

var is scoped to the nearest function block and let is scoped to the nearest enclosing block. But both are global if outside any block.

function testForLet() {

// i is not visible here

/\*

Variables declared with let are not accessible before

they are declared in their enclosing block.

\*/

for (let i = 0; i < 10; i++) {

// i is only visible here

}

// i is not visible here

}

function testForVar() {

// i is visible here

for (var i = 0; i < 10; i++) {

// i visible here

}

// i visible here

}

var will let you re-declare the same variable in the same scope. On the other hand, let will not:

let abc : string = "test";

let abc : string = "abc"; //Syntax Error

var abc = "test";

var abc = "abcd"; //no problem

TypeScript’s data types

Whether a data item is variable or constant it always has a data type. The data type describes the type of data that can be stored there and what types of operations can be performed on the data.

The basic primitive data types are the any, Boolean, number, string, array, void, and Enumtypes.

Any

All types in TypeScript are subtypes of Any type. This type is used to represent any JavaScript value.

//string

let hasValue : any = "yes";

//boolean

hasValue = true;

//now a number

hasValue = 1;

Boolean

Boolean logic is based on true or false comparisons. A boolean variable can hold only one of two values—true or false.

let isOverspeed : boolean = false;

isOverspeed = true;

Number

All numbers in TypeScript are floating point values.

let speed : number = 100;

let distance : number = 10.50;

String

String is a data type that can hold text data, to include string literals we use single or double quotation marks in our scripts.

let url : string = 'http://brainbell.com/typescript';

//or using double quotes

url = "http://brainbell.com/typescript";

Array

An array is simply marked with the [] notation, similar to JavaScript, and each array can be strongly typed to hold a specific type, as seen in the code below:

let numbers : number [] = [1, 2, 3, 4];

Or using generic array type (will cover later):

let numbers : Array<number> = [1, 2, 3, 4, 5];

Enum

You can create your own data types with a fixed set of values. To create your own data type, use the keyword enum.

enum Month {Jan, Feb, Mar, Apr};

let january : Month = Month.Jan;

Void

It is the type for the result of a function that returns normally, but does not provide a result value to its caller.

function alertUser() : void {

alert ('example of void');