Setup Environment for TypeScript

- **Install Node JS on your PC**
  - Web Development requires a package manager software.
  - Package Manager is used to install various libraries that are required for project development.
  - There are various types of package managers like
    - **Yarn**
    - **NPM**
    - **NuGET**
    - **Bower**
    - **RubyGEMS** etc.
  - **Installing Node JS on your computer gets "NPM"** as Package Manager.
    - Visit https://nodejs.org/en/download/current/
    - Download and Install Node JS for your OS.

Note: If you have Node JS already installed on your PC. Make sure that its version is 10x and higher.

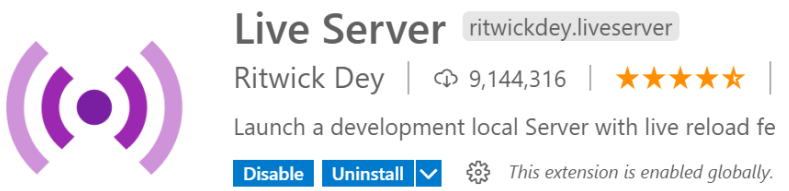
  - Test the version of Node JS and NPM from your command prompt.
    C:\> node -v
    C:\> npm -v

- **Install TypeScript on your PC**
  - Open your windows command prompt
  - Run the following command
    C:\> npm install -g typescript
  - You can check the version of TypeScript by using
    C:\> tsc -v

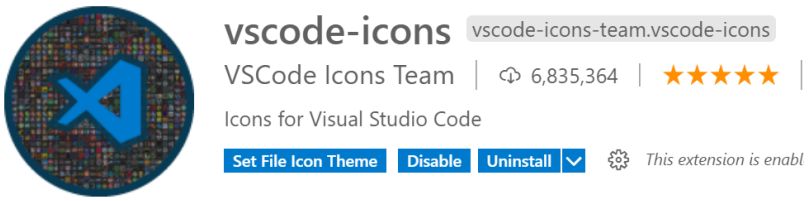
- **Install Editor on your PC**
  - It provides an Integrated Development Environment [IDE] from where you can build, debug, test and deploy application.
  - Developers can use various Editors like
    - Sublime
    - Visual Studio Code
    - Eclipse
    - Web Strom etc.
  - **Download and Install "Visual Studio Code" as Editor**
    https://code.visualstudio.com/

- **Install following plugins on your Visual Studio Code**
  - Open Visual Studio Code

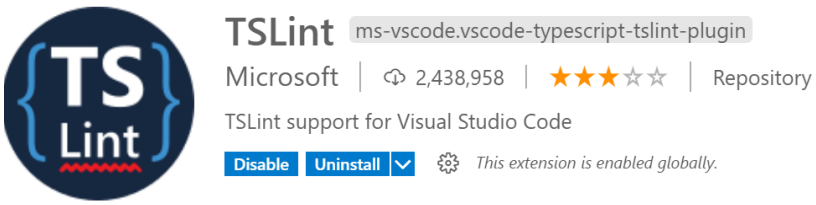
- Go to “Customize” category and install support for “JavaScript”
- Go to “Extensions”
- Search and Download the following extensions
  - **Live Server**



  - **Vscode-icons**



  - **TsLint**



- **Create a new TypeScript Project**
  - Open File Explorer on your PC
  - Create a new folder for project **C:\TypeScript-4**
  - Open the project folder in Visual Studio Code.

        - File Menu -> Open Folder -> C:\TypeScript-4
    - Add a new file by name **hello.ts**

      let username:string = "John";
      document.write("Hello ! " + username);
    - Go to “Terminal Menu -> New Terminal” [Ctrl + `] backtick
    - Run the following command

      C:\TypeScript-4> tsc hello.ts
    - This will generate “hello.js”
    - Create “Index.html” page and link the JS file

```
<!DOCTYPE html>
<html>
   <head>
      <title>Index</title>
      <script src="hello.js"></script>
   </head>
   <body>
      <h2>TypeScript Project</h2>
   </body>
</html>
```

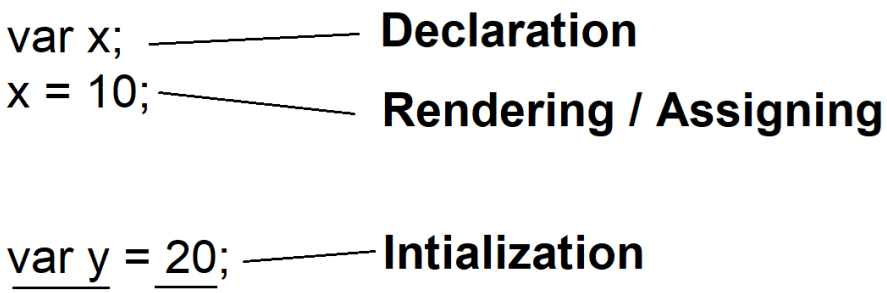
TypeScript Language Basics

- Variables

- Data Types
- Operators
- Statements

Variables

- Variables are storage locations in memory, where you can store a value and use it as a part of any expression.
- Variables configuration comprises of 3 phases
    - Declaration
    - Rendering / Assigning
    - Initialization



- In JavaScript variable declaration is not mandatory if it is not in strict mode.
- In TypeScript variable declaration is mandatory. As it is strict super set of JavaScript.
- Variables in TypeScript can be declared by using following keywords:
    - var
    - let

- const

var:

- It is used to define a function scope variable.
- You can declare variable in any block of a function and access from any location in function.
- Var allows
  - Declaration — var x;
  - Assigning or Rendering — x = 10;
  - Initialization — var y = 20;

Ex:

```
function f1()
{
  var x;      // declaring
  x = 10;     // rendering
  if(x==10)
  {
    var y = 20; // initialization
  }
  console.log("x=" + x + "\n" + "y=" + y);
}
f1();
```

- Var allows “Shadowing”.

- Shadowing is the process of re-declaring same name identifier in the scope.

  Ex:

```
function f1()
{
  var x = 10;
  if(x==10){
     var y = 20;
     y = 21;      // rendering
     var y = 30;  // shadowing
  }
}
f1();
```

- Var allows “Hoisting”.
- Hoisting is the technique supported by compiler where the declaration of variable can be after using the variable, which is not possible in many programming languages.

  Ex:

```
function f1()
{
  x = 20;
  console.log("x=" + x);
  var x;   // hoisting
}
f1();
```

Let:

- It is used to define a block scoped variable.
- It is accessible only within the declared block.

  Ex:

```
function f1()
{
  let x;
  x = 10;
  if(x==10)
  {
    let y = 20;
  }
  console.log("X=" + x + "\n" + "Y=" + y);        //
Invalid – y not found.
}
f1();
```

- Let allows
  - Declaring
  - Rendering / Assigning
  - Initialization

  Ex:

```
function f1()
{
  let x;         // declaring
  x = 10;        // rendering
  if(x==10)
```

```
    {
      let y = 20; // initialization
      console.log("X=" + x + "\n" + "Y=" + y);
    }
  }
  f1();
```

- Let will not allow “Shadowing”

```
  Ex:
  function f1()
  {
    let x;
    x = 10;
    if(x==10)
    {
      let y = 20;
      let y = 30;        // invalid – can’t re-define
      console.log("X=" + x + "\n" + "Y=" + y);
    }
  }
  f1();
```

- Let will not allow “**Hoisting”**

```
  Ex:
  function f1()
  {
    x = 10;
    console.log("x=" + x);
```

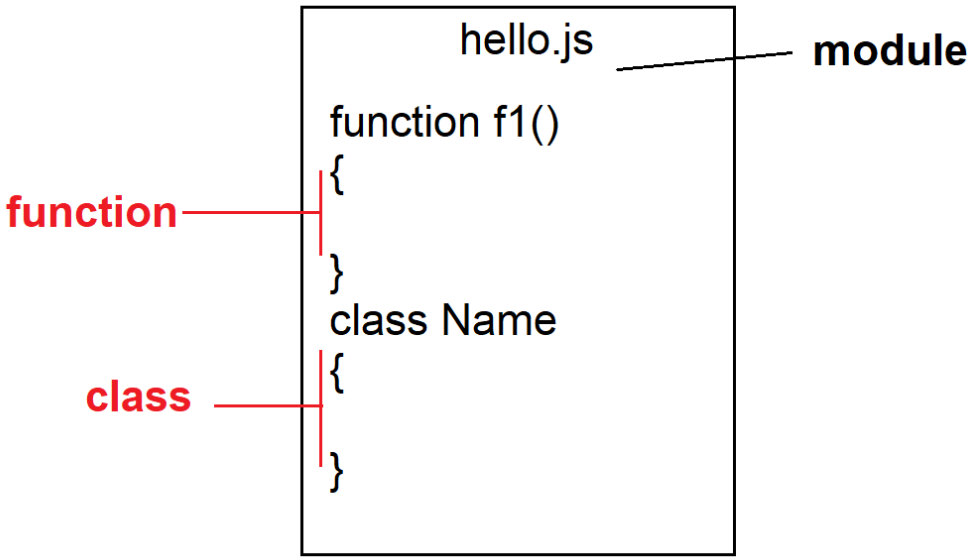
```
    let x;          //invalid- Hoisting
}
f1();
```

const:

- It defined block-scoped variable.
- It will not allow declaring and assigning.
- It will allow only initialization.
  Ex:
  const x; //**invalid:** const declaration must have initialization
  x = 20; //**invalid:** can't assign value to const.
- const will not allow shadowing.
- const will not allow hoisting.

Global Scope for Variable

- Variables declared inside module are considered as Global.
- You can use var, let or const for global variable.



Ex:

```
var x = 10;
let y = 20;
const z = 30;                    // all are Global
function f1(){
  console.log(`Function 1 : \nx=${x}\ny=${y}\nz=${z}`);
}
function f2(){
  console.log(`Function 2 : \nx=${x}\ny=${y}\nz=${z}`);
}
f1();
f2();
```

Variable Naming

- Variable name must start with an alphabet or underscore “_”
- It can be alpha numeric and recommended to use only _ as special chars.

Ex:

```
var _sales;          // valid
var sales2020;       // valid
var 2020Sales;       // invalid
var sales_2020;      // valid
var sales 2020;      // invalid
var sales.2020;      // invalid
```

FAQ: Why developers use “_” for naming?

- “_” indicates that it requires further implementation.

```
var _name;
get Name() {
    return _name;
}
set Name(newName) {
```

```
        _name = newName;
    }
```

- Variable name can’t be more that 255 chars.
- Variable name must be in camel case. [txtName, btnSubmit]

Data Types

- TypeScript is a strongly typed language.
- It specifies the data type for any variable, so that variable can handle only the type specified.
- Data Type defines the data structure.
- In computer programming data structure determines the type and range of value that can be stored in memory.
- All data types in TypeScript are derived from “any”, which is root type.
- “any” type can initially to be declared to handle any type of value like string, number, boolean etc.

  Syntax:

  let variableName: datatype = value;

  Ex:

  let x:any = 10;

  x = “John”; **Valid**

  x = true; **Valid**

- You can restrict the type by using the Primitive or Non-Primitive Data types provided by TypeScript.

Primitive Data Types

- The primitive data types use a memory stack to store value.
- Stack uses the mechanism of “Last-in-First-Out” [LIFO].
- Primitive types are immutable.
- Their structure can’t change dynamically.
- They have a fixed range for value.
- TypeScript Primitive Types are
    - number
    - string
    - boolean
    - null
    - undefined

FAQ: If data type is not defined for variable, can you store a value in TypeScript?

A. Yes, the default data type for variable is determined as “any” type.

Number Type

- It is similar to JavaScript [ES6] number type.
- It is used to handle various numeric values like
  - Signed Integer
    x:number = -10;
  - Unsigned Integer
    x:number = 10;
  - Floating Point
    x:number = 45.50
  - Double
    x:number = 450.420;
  - Decimal
    x:number = 45.96969584838282; // 29 decimal places.
  - Exponent
    x:number = 2e3; [2 x 10^3] = 2000
  - Hexadecimal
    x:number = 0xf00d;
  - Binary
    x:number = 0b1010; [10]

    - Octal
      x:number = 0o744;
    - Bigint
      x:number = 100n; [binary format content]
- The datatype “number” is used to store any of the given value types
  Ex:
  let signed:number = -10;
  let unsigned:number = 10;
  let exponent:number = 2e3;
  let binary:number = 0b1010;
  console.log(`Exponent=${exponent}\nBinary=${binary}`);

FAQ: What is the min and max number range?

- From ES6 the min and max number range in JavaScript and TypeScript is
  Min Number=-9007199254740991
  Max Number=9007199254740991

Ex:

```
console.log(`Min
Number=${Number.MIN_SAFE_INTEGER}\nMax
Number=${Number.MAX_SAFE_INTEGER}`);
```

String Type

- String is a literal with group of characters enclosed in
  - Double Quotes “string”
  - Single Quotes ‘string’
  - Back Tick `backtick`
- String type is defined by using “string” keyword.
- You can use **single and double** quotes to swap between inner and outer string.

  Syntax:

  let link:string = "Home";
  document.write(link);

  Syntax:

  let link:string = 'Home';

  document.write(link);

- Backtick is available from ES6 version.
- It allows to configure a string with embedded expression.
- Expression can be embedded by using “${ }”

  Ex:

  let uname:string = "John";
  let age:number = 22;
  let msg1:string = "Hello !" + " " + uname + " " + "You will be" + " " + (age + 1) + " " + "Next Year";

let msg2:string = `Hello ! ${uname} You will be ${age+1} Next Year`;
console.log(msg1);
console.log(msg2);

- A string literal may contain special characters, but few special characters escape printing.
- You have to print the non-printable characters by using “\”
  Syntax:
  let path:string = "\"D:\\Images\\Pics\\car.jpg\"";
  console.log(path);
- Your string can use HTML elements if it is presenting on web page.
  “Home”
  **“Hello !
Welcome to Angular”.**
- If your string is targeted towards console then you have to use escape sequence characters.

\n	New Line
\t	Horizontal Tab
\v	vertical space.

Ex:

```
let path:string = "Hello ! \n Welcome";
console.log(path);
```

String Manipulations

- All string manipulations are same as JavaScript string.
- The “string” object provides a set of properties and methods that are used to format and manipulate string.
- The commonly used JavaScript string functions
    - charAt()
    - charCodeAt()
    - concat()
    - endsWith()
    - startsWith()
    - slice()
    - substring()
    - substr()
    - indexOf()
    - lastIndexOf()
    - localeCompare()
    - replace()
    - search()
    - split()
    - toString()
    - toUpperCase()
    - toLowerCase()
    - bold()
    - italic()
    - fontcolor()

- fontsize()
- small()
- big()
- sub()
- sup()

Boolean Type

- Boolean types are used in decision making.
- Boolean type is defined by using “boolean” keyword.
- Boolean type in TypeScript can handle only “true or false”.
- Boolean conditions can use only “true or false”.
- In JavaScript boolean conditions can uses 0 – for false and 1 – for true.

Ex:

```
let Name:string = "Samsung TV";

let Price:number  = 45000.55;

let InStock:boolean = true;

let Status:string;

if(InStock==true) {

   Status = "Available";

} else {

   Status = "Out of Stock";
```

```
}
console.log(`Name=${Name}\nPrice=${Price}\nStock=${Status}`);
```

Note: TypeScript supports “Union of Types”, it allows to configure a variable that can handle different types of values.

Syntax:

```
let variableName: DataType | DataType | DataType;
```

Ex:

```
let value:number|string;
value = "A";
value = 30;
value = true;
```

Undefined Type

- It is a type returned by variable when value is not initialized or rendered.
- The “undefined” keyword is used to verify the undefined value.

Ex:

```
let Name:string = "Samsung TV";
let Price:number|undefined;
```

```
Price = 34000.44;
if(Price==undefined)
{
   console.log(`Name=${Name}`);
} else {
console.log(`Name=${Name}\nPrice=${Price}`);
}
```

Null Type

- Null Type is used for variable if it is not supplied with a value dynamically during run time.
- It a variable is expecting value during run time and it is not supplied then dynamically it is configured as “null”.
- The keyword “null” is used to verify the null type.

Ex:

Hello.ts

```
let ProductName:string|null = prompt("Enter Product Name");
if(ProductName=="") {
   document.write("Name can't be Empty");
} else if(ProductName==null) {
   document.write("You canceled..");
```

```
} else {
   document.write(`Name=${ProductName}`);
}
```

> tsc hello.ts

Index.html

```
<head>
  <script src="hello.js"></script>
</head>
```

Summary: Primitive Types

- Number
- String
- Null
- Undefined
- Boolean

Non-Primitive Types

- Non-Primitive types are stored in memory heap. [Cache Memory]
- Don’t have any fixed range.
- The range varies according to the memory available.
- Mutable type.
- The structure of type can change according to state and situation.

- TypeScript Non-Primitive types are
    - Array
    - Object
    - Regular Expression

Array Type

- Arrays are used in programming to reduce overhead and complexity.
- Arrays can reduce overhead by storing values in sequential order.
- Array can reduce complexity by storing multiple values under one name.
- TypeScript array can store similar type of values or different types of values.
- If array allows different types of values then it is called as “Tuple”.

Declare Array:

```
let variableName:datatype[];
let variableName:number[];      // Array of Numbers
let variableName:string[]       // Array of Strings
let variableName:any[]          // Tuple – Various types of values
```

Note: Declaring array will not allow to store values. You must initialize or assign memory for array.

Initializing memory for Array:

- You can initialize memory for array by using
  - “[]” array meta character
  - “new Array()” array constructor.

Syntax: Initializing

```
let variableName:number[] = [ ];

let variableName:number[] = new Array();
```

Syntax: Assigning

```
let variableName:number[];

variableName = [];
```

What is difference between Array meta character [] and Array() constructor?

- Meta character allows a “Tuple” to initialize into memory, when data type is defined as “any”.
- Array constructor will not allow to initialize different types of value even when the data type is defined as “any”.
- Array constructor allows the value to similar type, based of first value type.

Ex:

```
let values:any[] = [10, "A", true];
console.log(values[0] + "\n" + values[1]);

let values:any[] = new Array(10, "A");     //Invalid
console.log(values[0] + "\n" + values[1]);
```

Note: Array constructor will allow to assign or render different types of value. It will not allow only initialization of value.

Ex:

```
let values:any[] = new Array();
values[0] = 10;
values[1] = "A";
values[2] = true;
```

Store Values into Array:

- Array values are stored by using property reference.
- Array properties are string type and they have the same name as index name.

Ex:

```
let products:string[] = [];
products["0"] = "Samsung TV";
products["1"] = "LG Mobile";
for(var property in products)
{
   console.log(`[${property}]-${typeof property}
${products[property]}\n`);
}
```