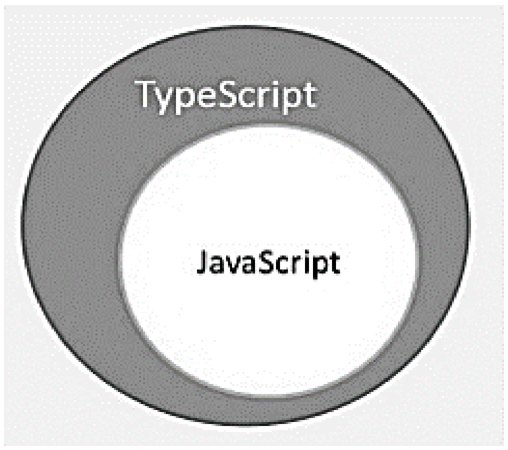
TypeScript

TypeScript Definition:

By definition, "Typescript is JavaScript for application-scale development." TypeScript is a strongly typed, object oriented, compiled language. It was designed by **Anders Hejlsberg** designer of C# at Microsoft. TypeScript is both a language and a set of tools. TypeScript is a typed superset of JavaScript compiled to JavaScript. In other words, TypeScript is JavaScript plus some additional features.



Features of Type Script:

Type is just Java Script:

TypeScript starts with JavaScript and ends with JavaScript. Typescript adopts the basic building blocks of your program from JavaScript. Hence, you only need to know JavaScript to use TypeScript. All TypeScript code is converted into its JavaScript equivalent for the purpose of execution.

Java Script is TypeScript:

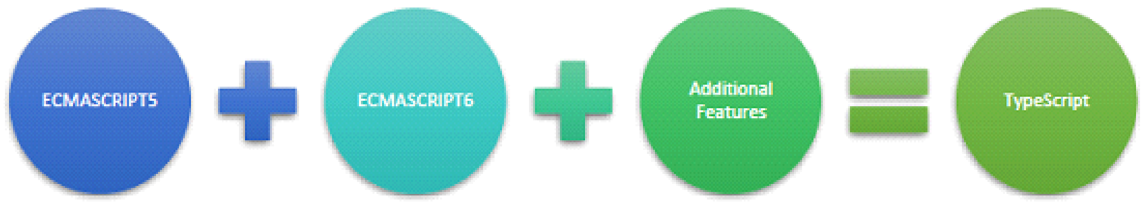
This means that any valid **.js** file can be renamed to **.ts** and compiled with other TypeScript files.

TypeScript is Portable:

TypeScript is portable across browsers, devices, and operating systems. It can run on any environment that JavaScript runs on. Unlike its counterparts, TypeScript doesn't need a dedicated VM or a specific runtime environment to execute.

TypeScript and ECMAScript:

The ECMAScript specification is a standardized specification of a scripting language. There are six editions of ECMA-262 published. Version 6 of the standard is codenamed "Harmony". TypeScript is aligned with the ECMAScript6 specification.



TypeScript adopts its basic language features from the ECMAScript5 specification, i.e., the official specification for JavaScript. TypeScript language features like Modules and class-based orientation are in line with the ECMA Script 6 specification. Additionally, TypeScript also embraces features like generics and type annotations that aren't a part of the EcmaScript6 specification.

Why we use Type Script?

The benefits of TypeScript include:

Compilation:

JavaScript is an interpreted language. Hence, it needs to be run to test that it is valid. It means you write all the codes just to find no output, in case there is an error. Hence, you have to spend hours trying to find bugs in the code. The TypeScript transpiler provides the error-checking feature. TypeScript will compile the code and generate compilation errors, if it finds some sort of syntax errors. This helps to highlight errors before the script is run.

Strong Static Typing:

JavaScript is not strongly typed. TypeScript comes with an optional static typing and type inference system through the TLS (TypeScript Language Service). The type of a variable, declared with no type, may be inferred by the TLS based on its value.

TypeScript **supports type definitions** for existing JavaScript libraries. TypeScript Definition file (with **.d.ts** extension) provides definition for external JavaScript libraries. Hence, TypeScript code can contain these libraries.

TypeScript **supports Object Oriented Programming** concepts like classes, interfaces, inheritance, etc.

Components of Type Script:

TypeScript has the following three components

Language:

It comprises of the syntax, keywords, and type annotations.

The TypeScript Compiler:

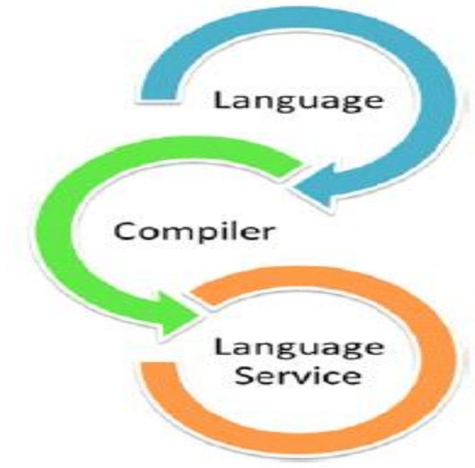
The TypeScript compiler (tsc) converts the instructions written in TypeScript to its JavaScript equivalent.

The TypeScript Language Service:

The "Language Service" exposes an additional layer around the core compiler pipeline that are editor-like applications. The language service supports the common set of typical editor operations like statement completions, signature help, code formatting and outlining, colorization, etc.

Declaration Files:

When a TypeScript script gets compiled, there is an option to generate a **declaration file** (with the extension **.d.ts**) that functions as an interface to the components in the compiled JavaScript. The concept of declaration files is analogous to the concept of header files found in C/C++. The declaration files (files with **.d.ts** extension) provide intelligence for types, function calls, and variable support for JavaScript libraries like jQuery, MooTools, etc.



Every language specification defines its own syntax. A TypeScript program is composed of

- Modules
- Functions
- Variables
- Statements and Expressions
- Comments

Your First Type Script Code

Let us start with the traditional “Hello World” example

var message: string=”Hello World”

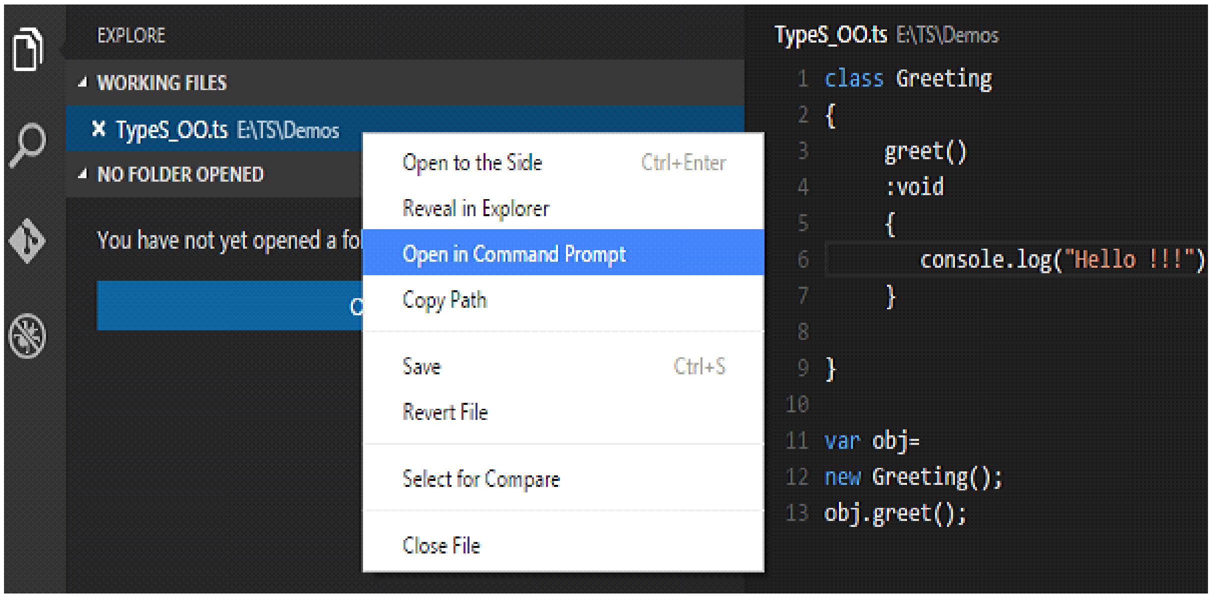
console.log (message)

Compile and Execute a Type Script Programme:

Let us see how to compile and execute a TypeScript program using Visual Studio Code. Follow the steps given below

Step1. Save the file with .ts extension. We shall save the file as Test.ts. The code editor marks errors in the code, if any, while you save it.

Step 2 – Right-click the TypeScript file under the Working Files option in VS Code's Explore Pane. Select Open in Command Prompt option.



```
class Greeting
{
     greet()
     :void
     {
        console.log("Hello !!!")
     }

}

var obj=
new Greeting();
obj.greet();
```

Step 3.To compile the file use the following command on the terminal window.

tsc Test.ts

Step 4.The file is compiled to Test.js. To run the program written, type the following in the terminal.

node Test.js

Compiler Flag:

Compiler flags enable you to change the behavior of the compiler during compilation. Each compiler flag exposes a setting that allows you to change how the compiler behaves.

S.No.	Compiler flag &Description
1.	**--help** Displays the help manual
2.	**--module** Load external modules
3.	**--target** Set the target ECMA version
4.	**--declaration** Generates an additional .d.ts file
5.	**--remove Comments** Removes all comments from the output file
6.	**--out** Compile multiple files into a single output file
7.	**--sourcemap** Generate a source map (.map) files
8.	**--module noImplicitAny** Disallows the compiler from inferring the any type
9.	**--watch** Watch for file changes and recompile them on the fly

Identifiers in Type Script:

Identifiers are names given to elements in a program like variables, functions.

Identifiers Are:

- Identifiers can include both, characters and digits. However, the identifier cannot begin with a digit.

- Identifiers cannot include special symbols except for underscore (_) or a dollar sign ($).
- Identifiers cannot be keywords.
- They must be unique.
- Identifiers are case-sensitive.
- Identifiers cannot contain spaces.

The Following Tables Lists a few of valid and Invalid Identifiers:-

Valid identifiers	**Invalid identifiers**
first Name	Var
first_name	first name
num1	first-name
$result	1number

TypeScript Key:

Keywords have a special meaning in the context of a language.

break	as	any	switch
case	if	throw	else
var	number	string	get
module	type	instanceof	typeof
public	private	enum	export

finally	for	while	void
null	super	this	new
in	return	true	false
any	extends	static	let
package	implements	interface	function
new	try	yield	const
continue	do	catch	

Whitespace and Line Breaks:

TypeScript ignores spaces, tabs, and newlines that appear in programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

TypeScript is Case-sensitive:

TypeScript is case-sensitive. This means that TypeScript differentiates between uppercase and lowercase characters.

Semicolons are Optional

Each line of instruction is called a **statement**. Semicolons are optional in TypeScript.

Example console.log (“hello World”)

console.log(“we are learning TypeScript”)

Comments in TypeScript:

Comments are a way to improve the readability of a program. Comments can be used to include additional information about a program like author of the code, hints about a function/ construct etc. Comments are ignored by the compiler.

TypeScript supports the following types of comments

- **Single-line comments (//)** – Any text between a // and the end of a line is treated as a comment
- **Multi-line comments (/* */)** – These comments may span multiple lines.

Example //this is single line comment

/* This is a Multi-line comment*/

TypeScript and Object Orientation:

```
class Greeting {

   greet ():void {

      console.log ("Hello World!!!")

   }

}

var obj = new Greeting();

obj.greet();
```

The above example defines a class *Greeting*. The class has a method *greet ()*. The method prints the string "Hello World" on the terminal. The **new** keyword creates an object of the class (obj). The object invokes the method *greet ()*.

On compiling, it will generate following JavaScript code.

```
//Generated by typescript 1.8.10

var Greeting = (function () {

   function Greeting () {

   }
   Greeting.prototype.greet = function () {
      console.log ("Hello World!!!");
   };
      return Greeting;
}());

var obj = new Greeting ();
obj.greet()
```

The output of the above program is given below

```
Hello World!!!
```

Q1.What are the Challenges in Modern Web Development?

Ans: 1.Unified UX

2. Fluid UX
3. User and SEO Friendly URL's (Routing)
4. Loosely Coupled and Extensible Architecture.
5. Simplified Deployment and Install

Q2.What is Solution?

Ans: 1.Better I will Build a SPA (AngularJs) For you world popular SPA Twitter. YouTube is a Podcasting.

2. Better I will now build a progressive Web Appication on Angular7.

Q3.How to Build SPA?

Ans: You can use JavaScript and HTML .

jQuery HTML

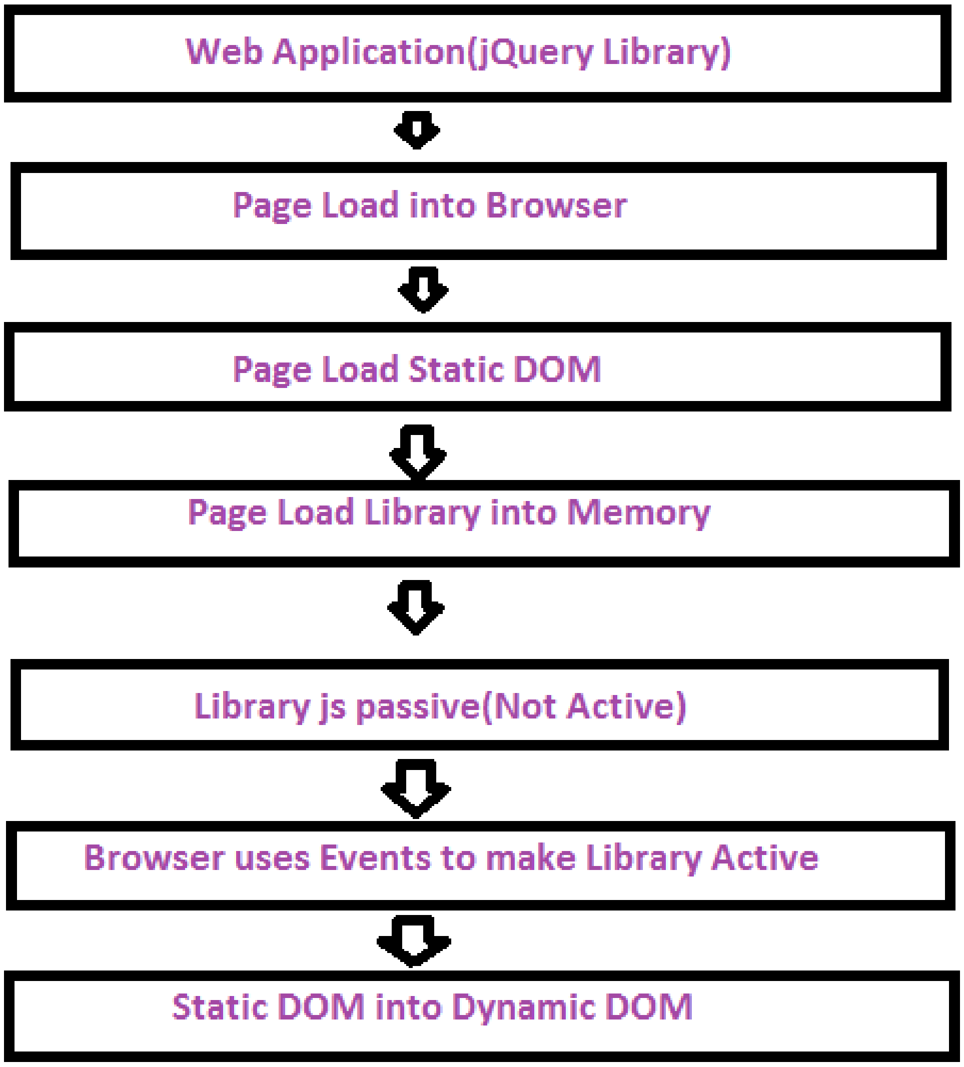
Then why Angular AngularJs.

Q4.What is Problem in JavaScript and jQuery?

Ans: JavaScript is a Language, jQuery is a Library and AngularJs is Framework.Angular7 is a Platform.

Q5.What is problem jQuery?

Ans: jQuery is a Library.


Web Application(jQuery Library)
Page Load into Browser
Page Load Static DOM
Page Load Library into Memory
Library js passive(Not Active)
Browser uses Events to make Library Active
Static DOM into Dynamic DOM

Q6.

Basic Requirement to build End to End Web Application:

Language/Tool	**Description**
HTML	This is a markup Language used for Presentation.
CSS	It makes the presentation more responsive.

Client Side Script:

It reduces the burden on server by handling Interactions and Validations Client Side.

Example: JavaScript, ECMAScript

Server Side Script:

It is a script employed on Server to generate response customized for ever Client request.

Example: PHP, JSP, ASP

Database:

It defines the data, queries, reports and analysis required for application.

Example: RDBMS, Oracle, Non-RDBMS

Middleware:

It is a software framework used in Multitier Application to handle communication

Example: JBOSS, Express

IDE'S:

It provides an Integrated Development Environment to Build, Debug, Test and Deploy Applications.

DTP:

Publishing Tools Required for Publishing Images and Animation for web

Note:

A Software Stack can be used for building End to End Application, Which includes MEAN, MERN etc.

M-MongoDB [Database]

E-Express [Middleware]

A-Angular [Client Side]

N-Nodejs [Server Side]

R-Reactjs[ClientSide]

Challenges in Modern Web Development:

1. Unified UX

A web application must reach broad range of devices ie.from Browser to mobile

- It must Work on low Bandwidth devices.
- It should not optimize its content.
- Mobile users must get access to everything.
- Application must have the same feel and look across devices.

2. Fluid User Experience

It is nothing but User must get access to everything from one Page.

- There must not be any navigation between Pages.
- User will stay on one Page from beginning to the End.
- New details are updated into Page without reloading the Page.
- Loosely coupled and extensible
- Application can get new updates without re-install.
- It must support extensibility without re-construction.
- We can extend application functionality without stopping it.
- User friendly and SEO friendly URL's.
- User can query any content easily
- User browsing will be kept in tracking so that it can suggest relative content.
- User can reach to any content easily.

5. Simplified Deployment and Upgrade

- Without restarting the device application can be installed and used.
- Single Step Installation.

Q7.What is the Solution?

Ans: Better Build

- SPA (Single Page Application)
- Progressive Web Application.

Q8. How to build SPA or Progressive Web Apps?

Ans: JavaScript & HTML

jQuery & HTML

KnockoutJs & HTML

BackboneJs & HTML

AngularJs & HTML

Q9.What are the issues with JavaScript and jQuery?

Ans: Use lot of DOM manipulations:

DOM-> Document Object Model

HTML is a collection of Elements and Attributes.

HTML has normal elements, void elements, RC Data elements, Foreign key elements.

Challenges in JavaScript and jQuery:

- JavaScript and jQuery use DOM Manipulations.
- It requires lot of reference and code.
- Js and jQuery will give access to BOM (Browser object Model), which leads to Navigation issues.
- Explicitly we have to control navigation.

  Ex: What happens when back button is used?
- Data Binding issues. How the changes in view are updated in Model (data)?
- Routing Issues.

  Ex: How new updates are added to the Page without reloading?
- Not secured URL’s

- Js and jQuery are object Based not object oriented

  Ex: what about Extensibility.
- Js and jQuery are not strictly typed

  Ex: How you will handle remote data?

  10. What is solution?

      Better use a Framework ClientSide.

Q10. What is a Framework?

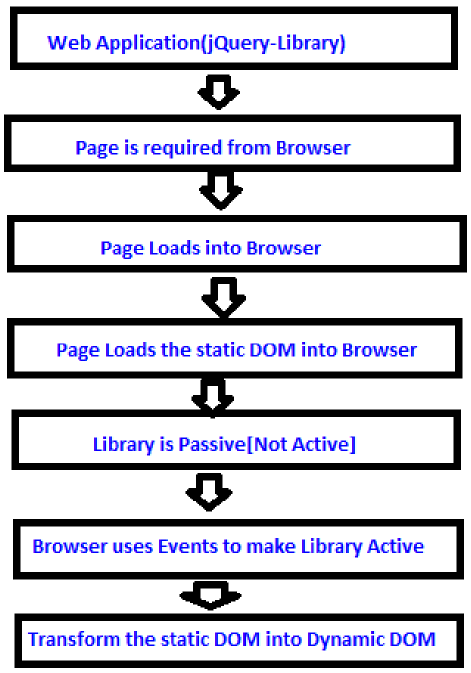
Ans: It is an Software Architecture Pattern.

- Architecture Pattern have the ability to build and control the application flow.
- The various Architecture Pattern in real world are:
  - Blackboard System
  - Broken Pattern
  - Event –driven architecture
  - Implicit Invocation
  - Layer
  - Micro Services
  - Model-view-controller(MVC)
  - Presentation-abstraction-control

- Model-view-presenter(MVP)
- Model-view-view Model(MVVM)
- Multitier Architecture(often 3-tier or n-tier)

Library Vs Framework

Library:


Web Application(jQuery-Library)
Page is required from Browser
Page Loads into Browser
Page Loads the static DOM into Browser
Library is Passive[Not Active]
Browser uses Events to make Library Active
Transform the static DOM into Dynamic DOM

Framework:

- Library is controlled by Browser Events.Hence Application have to control its own flow.
- Framework can control the application flow.
- The process of converting Static DOM into Dynamic DOM is known as

"BootStrapping".

Q11.What is AngularJs?

Ans: AngularJs is an open source cross platform.JavaScript based Framework to build Single Page Application.It uses "Apache Cordova" a cross platform framework for bulding Mobile Apps. It is developed by Google and maintained a community of Developers and organisation.The Latest version is "1.7.8" [2019].

Q12.What are issues with AngularJs?

Ans: It is not Designed for what we use are using.It have lot of Gap's.

[As-Is,To-be,GAP]

As-Is :What your technology supports

To-be:What Client needs?

GAP:What you can't satisfy with your technology?

It uses a legacy of library that makes rendering slow.

It have version compatability issues.

It is maintained by different technolgy developers.Hence to do the same thing there are so many alternatives.It leads to priority issues.

Q13.What is Angular?

Ans:1.Angular is TypeScript based open source front end Web Application Platform.

2.It is lead by Angular Google team at Google and by a community of indiviuals and corporations.

3.Angular is a complete rewrite from the same team that buld AngularJs.

4.Angular uses TypeScript ,which is class based OOP and Strictly Typed.

Angular leverage the benfits of OOP to achieve code reusability,code seperation,code extensibility etc..

5.Angular is component based and Modularis Approach these makes Angular more faster and light weight.

Angular is a Developer platform with Language and tools like Materials,Animation etc...

6.The various versions of Angular are

Angular2-> September14,2016

Angular4 ->March 23rd,2017

Angular5->November 1st,2017

Angular7->October 18th,2018

TypeScript:

1.TypeScript is an open source programming language developed and maintained by Microsoft.

2.It is a strict syntatical superset of JavaScript and adds optional   static typing to language.

3.Typescript is a superset of JavaScript.

Hence existing JavaScript programme are also valid TypeScript programs.

TypeScrippt is Influenced by JavaScript ,Java,C#.

TypeScript Influenced language called "AtScript" designed by Google

The first release of TypeScript is 3.4.3 released on 10th April 2019.

TypeScript is completely built with same TypeScript Language.It comprises of following components.

TypeScript Architecture Diagram:

1.The core TypeScript compiler is used to compile the TypeScript code and identify the compiletime error.

2.The standalone compiler is responsible for translating the TypeScript code into Target JavaScript.

3.Language service uses a Reflection Mechanism that allow to indentify the members of any specific component used in language.

4.Vs Shim is responsible for defining the rules for Language that can be CrossCompiled into JavaScript

5.TS Server defines a platform to load,compile and genereate the output.

6.The Editor tools defines the language rules foe specific Environment.The various Editors are Vscode,Sublime,WebStorm,Brackets,etc...

7.The Vs Managed language service is responsible for   making the TypeScript code cross platform

Features of TypeScript Language:

Type Annotation:

The Type Annotation in Compiler programming refers to the type signature that defines input and output type for a function.

Type Inference:

It determines the datatypes of value implicitly and handles specific type dynamically.

TypeScript:

it is a techinique that resolve the issues dynamically during runtime.Itsave the compiler time.

TypeScript supports a concept called Genereics.

```
function f1(a:number,b:number)

{

}

function f1<T>(a:T,b:T)

{

}

f1<number>(10,10)

f1<string>("A",10)
```

Generics:-

Generics refersto type safe and strongly typed for specific type of value dynamically.It identifies the type and infer the type.ie.Generics

Interface:

Interface defines a contract that is rules for designing a comment.

TypeScript support "Async" and "Await".

Async/Await

Await for first request   wait until response come and wait until response come.

Async will not wait for response all are light weight.Async will not wait it will movecan to next one.

Asynchronous:

It refers to an Asynchronous process where a request is made and will not wait for response.it proceeds to the   next whenever response is ready it will start the process.

1.Download and Install NodeJs

Package Manager->NPM,Composer,Grunt,Growl,yarn,Nuget

First we need to install NPM

"www.NodeJs.org" =>10.15.3 LTS

Recomended for Most Users

Window ->windows installer

C:Users\Administrator>node _v v10.15.0

c:Users\Administrator>

2)Learn Download Visual Studio Code[IDE]

WebStrom,Sublime,Vs,Ecllipse

[IDE]-Build ,Test,Application

Visual Studio Code-official site,windows,MAC OS

Windows

window 7,8 etc

Download and install TypeScript Languagecommand prompt:

>npm install -g TypeScript

npmJs is the officialwebsite.Package is a Collection of Library.you will get Information about that Package there are several other option also.

2.Dowload Visual Studio Code[IDE]

3.Download and Install TypeScript

SetUp Environment for TypeScript:

Download and Install NodeJs onyour Computer

https://nodejs.org/en/download/[.msi] file

2.Download and Install TypeScript.

>open windows command   prompt

>Type the following Command

>npm install _g TYPESCRIPT

Test:

>node _v ->nodejs version

>npm -v ->npm version

Note:

you can   get the Package names their various and bugs information and goto site

Test:>node -v->nodejs version

>npm -v->"https://www.npmjs.com"

Angular App

_TypeScript

ctrl+backTick[`]~

PS D:\ TypeScript>tsc demo.ts

    D:\ TypeScript>tsc demo.js

Search

Open in Browser=>Install this Extension.

Creating a new Project for TypeScript:

1.Open your pc location in Explorer

2.Create a new Folder

D:\TypeScript

3.Open Visual Studio Code IDE

4.Go to "File->Open Folder"

5.Select "D:\TypeScript" and open

Add a new TypeScript File:-

1.Open Project Explorer

2.Click on "[+]" new file button

3.Name it as "demo.ts"

4.Write the following code

```
console.log("welcome to TypeScript");
```

Transcompile into JavaScript

1.Open Terminal

ctrl+`[backtick]

2.Use TypeScript compiler command

```
D:/  TypeScript>tsc demo.ts
```

3.This will generate a new file "demo.js"

Run JavaScript File:

1.open Terminal

ctrl +

2.Type the following command

```
>node demo.js
```

You canlink to HTML File:

1.Add a new HTML file into project "Home.html"

2.Link js file

```
<head>

<script src="demo.js"></script>

</head>
```

3.Install a Plug -in "open inBrowser"

->Go to "Extentions"in Vs Code

->Search for "Open in Browser"

->Install for Vs Code

->Now you can right click on HTML files and Openin Browser.

```
D:\TypeScript>tscdemo.ts

D:\TypeScript>node demo.js
```

Variables in TypeScript:

Variables are named storage locations in memory,where you can store a value and use it as a part of any expressions

HomePage:

Example:

```
                              <html>

                                <head>

             <script>
```

```
                                   "use strict":
                                   function f1( ){
                                   x=10;
                                   document.write("x=" + x);
                                     }
                                  f1( );
                                 </script>
```

o/p: x=10

Declaring Variables is important when we use JavaScript in strict mode.

In TypeScript you have to declare and use a variable.

you can declare variable in TypeScript by using var,let,const.

a)var

b)let

c)const

```
Example: var a=10;
                  let    b=20
                 const
                 console.log("a="+a+"\n"+"b="+b+"\n"+"c="+c);
```

```
>tsc demo.ts
>node demo.js
```

var is function scoped Variable.

let and const are blockscoped Variables.

Example: function f1()

```
                    {
                      var    x=10;
                        if(x==10){
                        let    y=20;
                      {
                      log(x=+x,y=+y)
                      }
                      }
```

let y is not assessible as it work within the block.

var use can use anywhere let is only block scope

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

y is not accessable outside block.

const cannot be used without Initializing a value

var,let allows shadowing.Shadowing same name identified by same value.

const y=20; //not avalible //error message cannot be declared for block scoped variables.

var will allow shadowing.

const z; value initialized but not valid

value must be Initalized while declaring itself we need to declare value for it.Same name Identifier must be there with   different values.

Var and let can render value const must be initialized.

```
function Outer( ){

var x=10;

function inner( ){

var y=20;

console.log("x ="+ x );

}

console.log(" y= "+ y);

}
```

let also works in the function it will not allow to Shadow.

TypeScript Language:

Varables in TypeScript:

Variables are storage locations in memory.

Where you can store a value and use it as a part of any expression.

TypeScript is in strict mode of JavaScript.Hence it is mandatory to declare Variables before rendering.

```
  Var x;    //Code Declaration
```

X=10; //Code Rendering

Initialize mean Declaring.Code Rendering means Rendering after Declaring.

x=10; //Invalid -x not Declared.

console.log(x);

The Variables can be declared in TypeScript by using Keyword

- var
- let
- const

Keyword	**Description**
var	Scoped Variable so that it is accessable from any location in the function.It allows Shadowing ie.Same name Identifier with different value.
let	It defines a block scope Variable that cannot be Shadowed.However you can define and later Render a Value.
const	It is also a block Scoped Variable that will not support Shadowing a const must be initialized with a value.

Syntax:
```
function f1(){

        var x=10;

        if(x==10){

          let    y=20;
```

```
    }
  console.log(" x= " +    x +    "\n" +    "y=" + y);
    }
```

Error: y   is not accessible outside block

Syntax: Shadowing

```
function f1()
{
var x=10;
if(x==10){
var x=20; //valid
let y=30;
let y=50; //InValid
}
}
```

Syntax: const reference

```
  function f1(){
  const x; //Invalid
    x=10;        //x-must be Initialized
    }
    "use strict"
function f1( ){
const x;
```

```
                x=10;

                console.log(" x= "+ x);

                let x;

                }
```

Syntax: const reference

```
                function f1(){

                const x;

                x=10;

               }
```

protoType follow   some sequence.

Interpreter will go whenever it required.

hoisting however it precides the order of hoisting.

TypeScript Variables:

TypeScript   Variables will not support hoisting.It is a process supported in JavaScript where the translator Executes the code declaration followed by rendering.

TypeScript compiler will sequentially executes the statements with a specific presiding order.Hence it will not support hoisting.

Syntax: function f1(){

```
                x=10;

               console.log("x="+ x);

                let x;

                }
```

Window related to Browser

- The Global Scoped for variable in TypeScript is defined by declaring it outside the function.
- JavaScript allows a Global scope for variable by declaring inside the function using window keyword.

Syntax:
```
let x=10; //global Ts Variable

        function f1( ){

        window.y=20;// global Js Variable.

        console.log("x in f1=" + x);

        console.log("y in f1="+y);

        }

      function f2( ){

      console.log("x is f2="+ x);

        console.log("y in f1="+y);

      }

    f1( );

    f2( );

    var length=255 chars

    variables are CaseSensitive

    function f1( ){
```

```
                    var name="John";
                    var name="Dhoni";
                   console.log(name);
                   }
             f1( );
  D:\TypeScript>ts demo.ts
```

Variable name cannot start with number or alphabet.

Variable name can be alpha numeric

Can't periods "."

we can use special characters but not recomonded

ECMA 255 chars long.

TypeScript variable must follow the given naming conventions

They are CaseSensitive

```
let Name="John";
let Name="Jonson"; //both are different
```

b)Variable name must start with an alphabet or _and can contain alpha numeric with_.

```
  var _Sales=12000; //valid
var 2019 sales=34000; //Invalid
var sales 2019=45000; // valid
```

var sales.2019=40000 // Invalid

var sales _2019=45000;// valid

c)ECMA Standards define Variable name length to max 255 chars.

However it can   vary according to Browser.

d) Variable name must speak what it is.

var id=10;

Variables in TypeScript:

Data Type:

It determines the type of value that can be stored in a given memory reference.

1.Primitive Types/Value Types

2.Non Primitive Types/Reference Types.

Primitive Type =>Stored in a memory Stack.

- LIFO
- Have a Fixed Value.
- There will be minimum and maximum value.
- number/integer -32767 32767
- accessed directly by their values.
- value types.

a. number
b. boolean
c. string

d. undefined

e. null

TypeScript DataTypes:

I. The Data Type determines Type of value that can be stored in a reference.

II. The TypeScript Data Types are categorized into two Types:

a. Primitive Types

b. Non-Primitive Types (or) Reference Types.

Primitive Types:

1.The Primitive Types are Stored in a Memory Stack.

2.They use the Mechanism LIFO.

3.They have a fixed value.

4.They are stored and accessed by value.

5.Hence They are Known as Value Types.

The TypeScript value Types are:

a. number

b. boolean

c. string

d. undefined

e. null

Number Type:

Signed Integer= -10

Unsigned Integer=10

Floating Point=3.5

Double=340.560

Decimal=4.556979794 -29

let Signed:number=-10;

let Unsigned:number=10;

let float:number=4.5;

let double:number=450.560;

let decimal:number=4.55697979795932;

let exponent:number=2e3;

console.log="Signed="+Signed+"\n"+"Unsigned="+Unsigned+"\n"+"Float="+Float +"\n"+"decimal="+decimal+"\n"

The number Type in TypeScript is used to handle numerical values which includes the following number Type:-

Signed Integer	-10
UnSigned Integer	10
Floating Point	3.5
Double	340.560
Decimal	4.556979794 -29
Exponent	2e^3 2*10''3=2000
Hex	oxfood
Binary	0b1010
Octal	0o174

Example:- "DataType.ts"

```
  let Signed:number= -10
let UnSigned:number=10;
let float:number=4.5;
let double:number=450.560;
let decimal:number=4.556979795932;
let exponent:number=2e3
console.log("Signed="+Signed+"\n"+"UnSigned"+UnSigned+"\n"+"Float="+float+"
\n"+"Double="+double+"\n"+"Decimal="+decimal+"\n"+"Exponent"+exponent)
```

```
output: Signed=-10
          UnSigned=10
          Float=4.5
         Double=450.560
         Decimal=4.556979795932
         Exponent=2000
```

Boolean Type:-

The Boolean type is True/False 1/0

1. The Boolean type allocate a binary type of memory that holds the address for a number but stores the values True (or) False.
2. The Boolean Type over Mathematical operation will define 1 for "True" and 0 for "False".
3. However Boolean Type Reference can store only two keywords True or False.

4. The Boolean Condition are also satisfied by using the keywords "True" or "False"   or "0" or "1".

Example:- let   iSValid:boolean=true;

```
                    let iSinvalid:boolean=false;

                    console.log(isValid+isInValid);//1

                    console.log(isValid) //true

                   if(isValid==1)//inValid Expression
```

string type

"   "

'   '

`    `   outer code must be "   "

inner code must be '   ' for string.

back tick use along with expressions.

let year:number=2019

let msg:string="Next year"+(year+1)

you have to bind a string with expressions that how we need to bind

>tsc String type.ts

>node Stringtype.js

escape sequence to print those in sequence.

Non printing chars has to print these

D:\\Images\Cars\\pic.js\

ps D:\TypeScript

3.String Type:

A string represents group of characters enclosed in codes.a string can enclose alphabet,number or special characters.

To enclose a string you can use following.

Closure	Description
""(double quote)	It is used to define the outer string
' '(single quote)	It is used to define the inner string
``(back tick)	It is used to define a string with expressions.

The TypeScript expressions are defined by using ${ }

Syntax1: string

```
let msg:string="welcome";
```

Syntax2:   with Inner String

```
let msg:string="alert('hello')";
```

Syntax3: string with expression

```
let year:number=2019;
let msg:string='Next year ${year+1}welcomes you';
```

Syntax4:WITHOUT bACKTICK EXPRESSION

```
let msg:string="Next Year"+(year+1)+"Welcome you";
```

Note: Special Characters in a string representable can escape printing.To print the non-Printable characters use "\"

Example: let path:string="D:\Images\car.jpg";

o/p: D:Imagescar.jpg

```
let path:string="D:\\Images\\car.jpg";
```

o/p: D:\Images\car.jpg

```
let msg:string="\" welcome "\";
```

o/p: "welcome"

Undefined:

The undefined type in TypeScript specifies that a value is not defined for a given reference.It can be verified by using keyword Undefined.

Syntax:

```
let x:number;
      console.log(x)
```

Example:

```
let price:number;
      if(price==undefined)
      {
  console.log("price is Not defined    with value");
      }
     else {
     console.log("price="+price);
     }
```

Null Type: Null and Empty

```
function f1( ){
console.log(" x= "+ x );
}
function GetName( ){
```

```
                                        let uname:string;

                                        uname=prompt("Your Name:");

                                        if(name==" "){

                                        document.write("Name can't be Empty")

                                          }

                                        else if(name==null){

                                        document.write("you cancelled")

                                          }

                                        else{

                                          }

                                          >tsc nulltype.ts

                                          >node nulltype.js
```

"nulltype.html"

```
  <DOCTYPE html>

<html>

<head>

<script src="nulltype.js"></script>

<script>

function bodyload( ){

GetName( );

}

</script>
```

prompt box:

- prompt reference is in the "uname".
- The Page if he use cancel prompt not created at all.
- During runtime reference unable to create if null is coming it is not empty value not there.
- The null type is TypeScript is used to represent a reference that doesnt contain any value (or) not supplied with a value at runtime.
- A refernce is created but value is not defined then TypeScript designates it as a "nul type".
- The keyword null in lowercase is used to verify the null   type.

Example: Add   a new TypeScript file by name "nulltype.ts"

```
      function GetName( ){
      let uname : string;
      uname=prompt("Your Name");
      if(uname==" "){
      document.write("Name can't be Empty");
      }
      else if(uname==null){
        document.write("you cancelled");
    }
else{
  document.write("you cancelled");
}
else{
document.write("Hello!"+ uname);
}
}
```

2.compile Ts File

```
>tsc nulltype.ts
>node nulltype.js
```

3.Create a new HTML file by name "nulldemo.html"

```
<!DOCTYPE html>
<html>
<head>
<script src="nulltype.js">
<script>
function bodyload( ){
GetName( );
}
</script>
</head>
<body OnLoad="bodyload( )">
</body>
</html>
```

output: This Page says

your name

Jonson

Hello!Jonson

Non-primitive Types:

1. The Non-primitve Types are stored in memory heap.
2. The heap memory refers to Browser Cache memory
3. The heap memory doesn't have any fixed size.Hence,the value range varies according to the browser.
4. The Non-primitive types are accessed by a memory reference.Hence they are also known as Reference Types.
5. The TypeScript reference types are

    1.Array Type

    2.Object Type

    3.Regular Expression Type.

Array Type:

Q.What is the use of an array in programming?

Ans:Array are used in programming

a. Reduce OverHead.
b. Reduce Complexity.

I. It reduce OverHead by storing values in Sequential Order.

II. It reduce the Complexity by storing multiple values under one Reference.

III. Array naturally can store various types of values.

IV. But TypeScript can restrict the values to similar Types.

Q.Your technology restricts according to your nature [ ]?

Ans: MetaCharacters * ? + [ ]

* Astrik=Many

+=One or More

?=one

[ ]=Range

dir p*.*

dir t p*.*

Array Type in TypeScript:

1.Array in Programming are used to reduce overhead and complexity

2.Array reduce overhead by storing values in sequential order

3.Array reduce complexity by storing multiple values under one reference name.

4.TypeScript Array can handle various types of values.

5.TypeScript Array Size can be changed dynamically.

6.TypeScript Array can also restrict an Array to handle specific type of Values.

7.The Array Types are defined by using the meta character the open close brase "[ ]"

Syntax: Array Constructor

let values :string[ ]=new Array("A","B");

let values:any[ ]=["A",20,true];

Syntax: Rendering Values into Array

let Values:string[ ];

Values[0]="A";

Values[1]="B";

In Client Side Scripting array is an object

Example: let cities:string[ ]=new Array( );

let products:string[ ]=[ ];

Values are dynamically assigned values into array.If you are not resolving error

Q.What is the difference between the Values for Square and Array Constructor?

Ans: Initialize with [   ] is resolved during compile time

Initialize with Array[ ] is resolved during runtime.

Array Manipulations:

i. Reading Values from an Array.

- toString( )
- join( )
- Slice( )

toString( )->Reads the values from Array and return Values from Array

CSV->Comma Seperated Values.

Join( )->It allows you to use your own delimeter

Slice( )->Allow you to read values from specific location.

Example: let products : string[ ]=["Tv","Mobile"];

console.log(products.join());

or Slice(1,3)

Starting will include with 1 ending with -1 .

If you want to extract we need to extract.

Note:

Reading values from Array:-

Function	**Description**
toString( )	It reads all Values of an Array and returns them as string Seperated with "," as delimeter.
join( )	It reads all values of an Array and return them as string seperated with any specified delimeter. "|","-->"
Slice( )	It reads the Values from an Array based on Specified index and return a new Array.

Example:

let products :string[ ]=["Tv","Mobile","Sunglasses","Watch"];

console.log(products.toString());

//output:Tv,Mobile...

console.log(products.join("|"))

//output: Tv/Mobile/...

console.log(products.slice(1,3));

//output['Mobile','Sunglasses']

2.Adding Element /Values into Array

- push( )
- unshift( )
- splice()

Example:

let products :string [ ]=["Tv","Mobile"];

console.log(products.toString( ));

products.push("watch","Mouse");

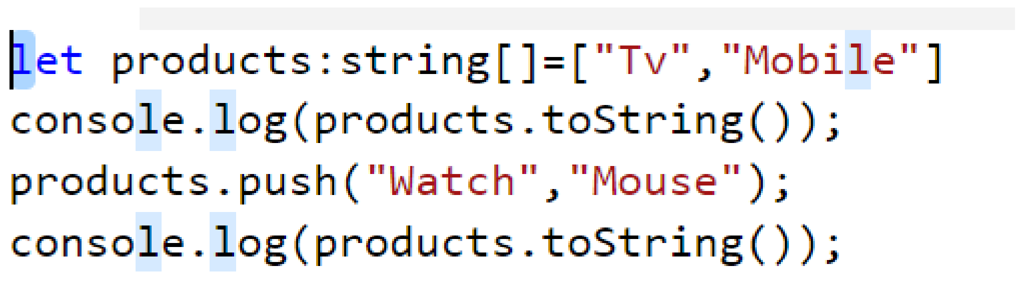
console.log(products.toString( ));

command prompt

>tsc arraytype.ts

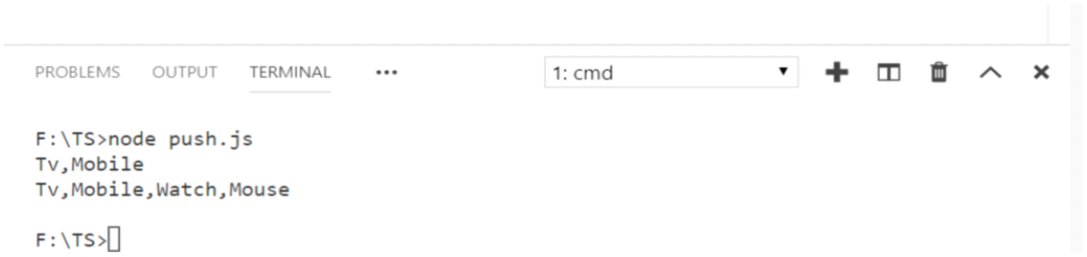
>node arraytype.js

push.ts



```
let products:string[]=["Tv","Mobile"]
console.log(products.toString());
products.push("Watch","Mouse");
console.log(products.toString());
```

Output:



```
F:\TS>node push.js
Tv,Mobile
Tv,Mobile,Watch,Mouse

F:\TS>
```

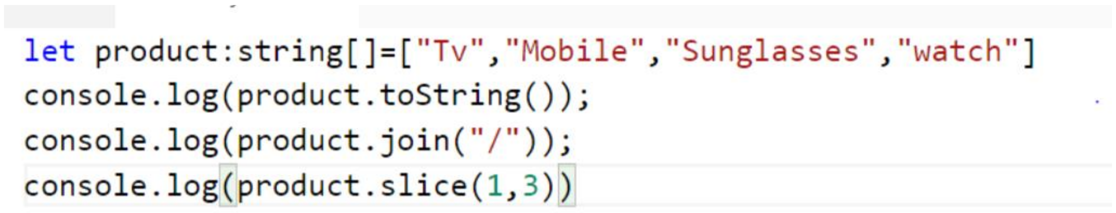
Rest operator :

It is used to pass many values.

```
...
```

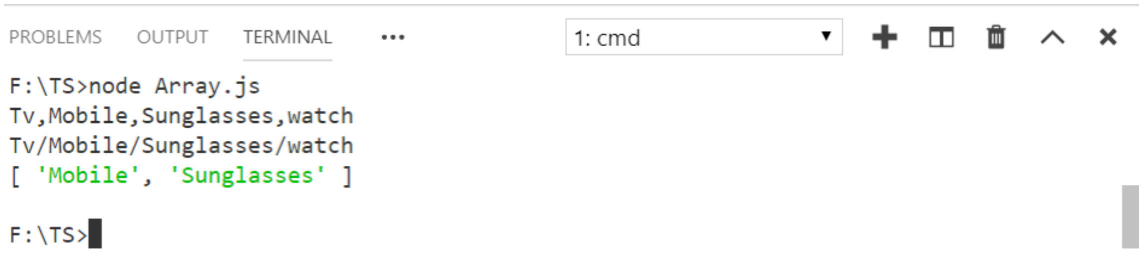
Function	Description
push( )	Append it is used to append a new element as last element.
unshift( )	It is used to append a new element into Array as first element.
splice( )	It is used to add new element into Array at any specified index.

Array.ts



```
let product:string[]=["Tv","Mobile","Sunglasses","watch"]
console.log(product.toString());
console.log(product.join("/"));
console.log(product.slice(1,3))
```

output:



```
F:\TS>node Array.js
Tv,Mobile,Sunglasses,watch
Tv/Mobile/Sunglasses/watch
[ 'Mobile', 'Sunglasses' ]

F:\TS>
```

5)Removing Elements from Array:

Function

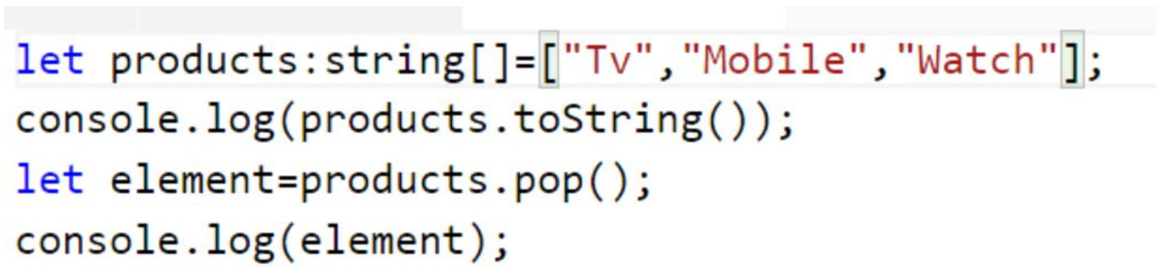
Description

1.pop( ) It remove & returns the last element in an Array.

2.Shift( ) It removes and returns the first Element in an Array.

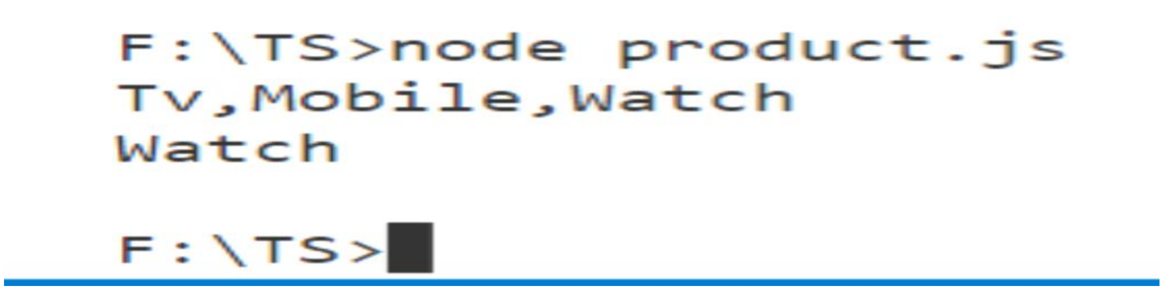
3.Spilce It remove a specified element from a Array based on the given index no.It return an array of remove elements.

pop



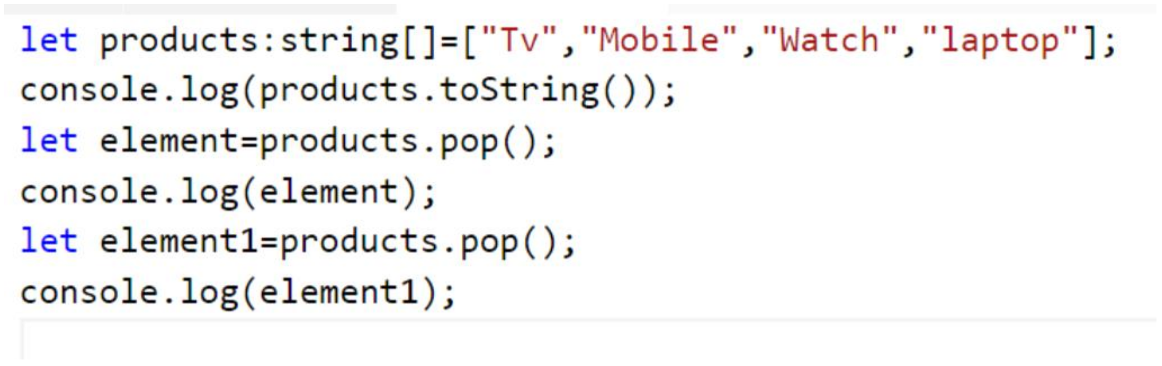
```
let products:string[]=["Tv","Mobile","Watch"];
console.log(products.toString());
let element=products.pop();
console.log(element);
```

output



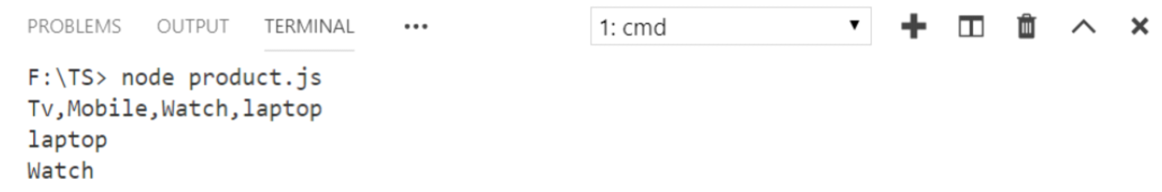
```
F:\TS>node product.js
Tv,Mobile,Watch
Watch

F:\TS>
```

pop



```
let products:string[]=["Tv","Mobile","Watch","laptop"];
console.log(products.toString());
let element=products.pop();
console.log(element);
let element1=products.pop();
console.log(element1);
```

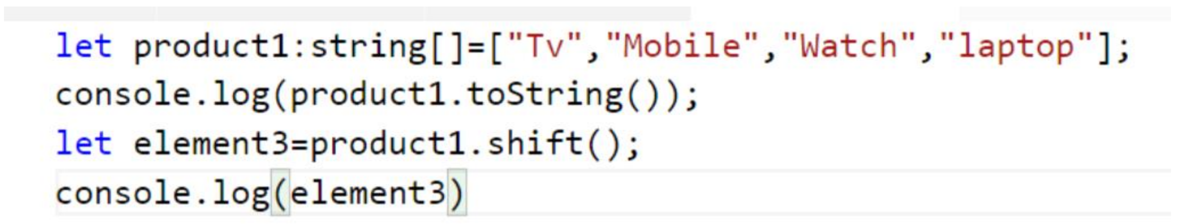
output



```
F:\TS> node product.js
Tv,Mobile,Watch,laptop
laptop
Watch
```

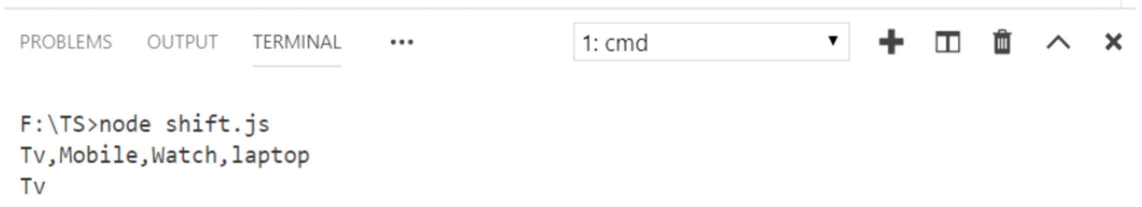
//Watch Removed

shift



```
let product1:string[]=["Tv","Mobile","Watch","laptop"];
console.log(product1.toString());
let element3=product1.shift();
console.log(element3)
```

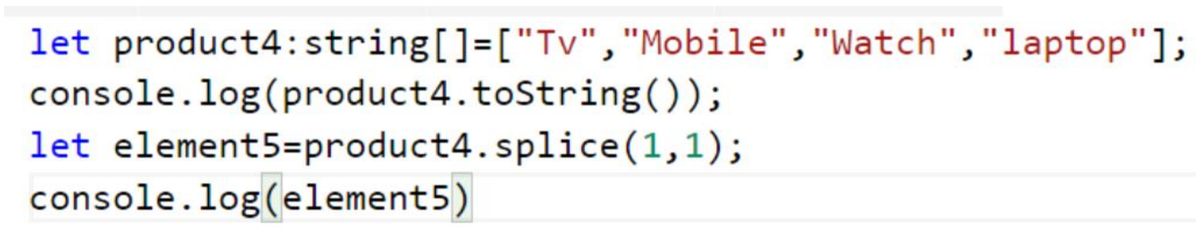
output



```
F:\TS>node shift.js
Tv,Mobile,Watch,laptop
Tv
```

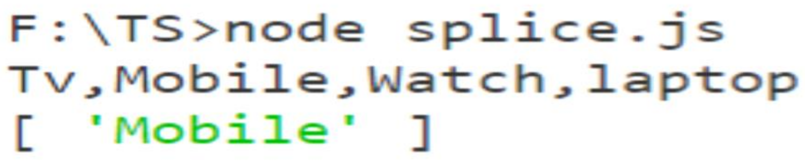
//Tv removed

splice



```
let product4:string[]=["Tv","Mobile","Watch","laptop"];
console.log(product4.toString());
let element5=product4.splice(1,1);
console.log(element5)
```

output:



```
F:\TS>node splice.js
Tv,Mobile,Watch,laptop
[ 'Mobile' ]
```

//Mobile Removed

Validating an Email

name.123@gmail.com

dotposInDomain=Str.IndexOf(".");

str.Slice(dotpos);

Slice(4,1);

Slice(4)

Search for Element in an Array

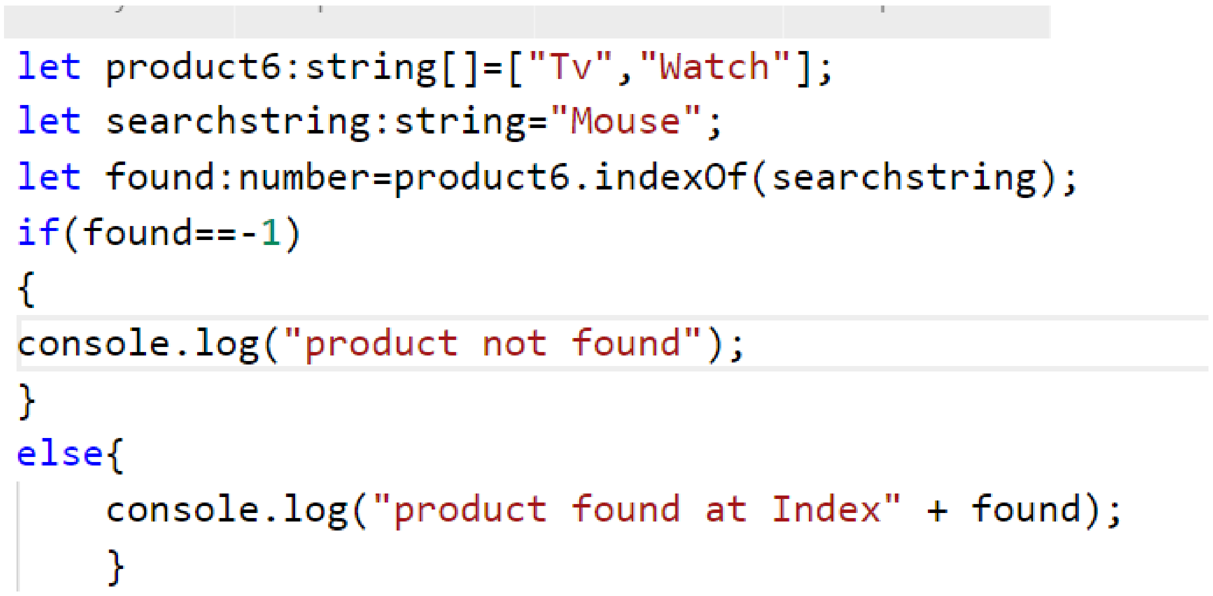
TypeScript provides the following Array functions that are used to Search for any Element in an Array.

Function	Description
indexOf( )	It returns the index number of the Element specified in search.If not found then it returns "-1".
lastIndexOf( )	It is similar to indexOf but finds the last occurance

of element and returns index.

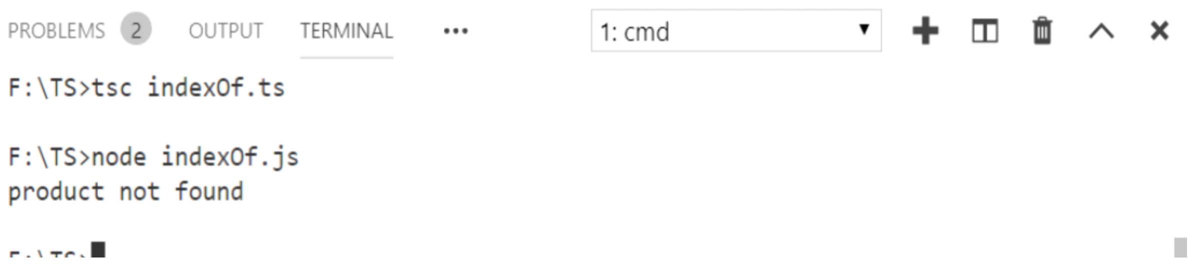
filter( ) It uses a function or an expression that can filter elements based on specified criteria

Example: indexOf



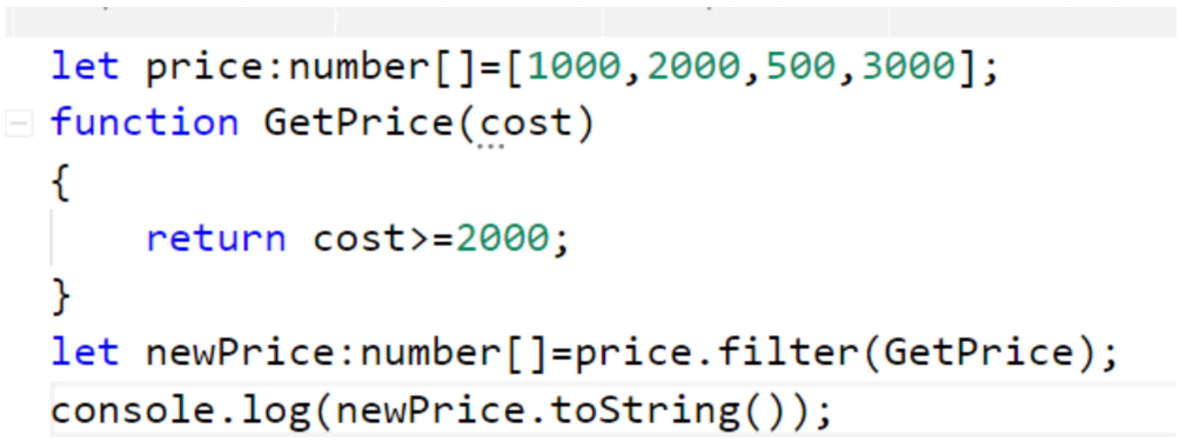
```
let product6:string[]=["Tv","Watch"];
let searchstring:string="Mouse";
let found:number=product6.indexOf(searchstring);
if(found==-1)
{
console.log("product not found");
}
else{
    console.log("product found at Index" + found);
    }
```

output



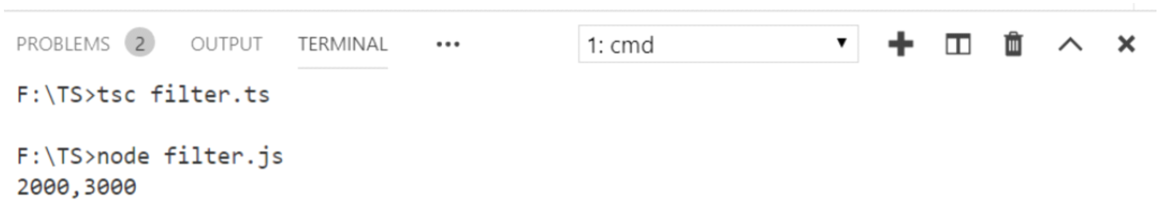
```
F:\TS>tsc indexOf.ts

F:\TS>node indexOf.js
product not found
```

Example: filter( )



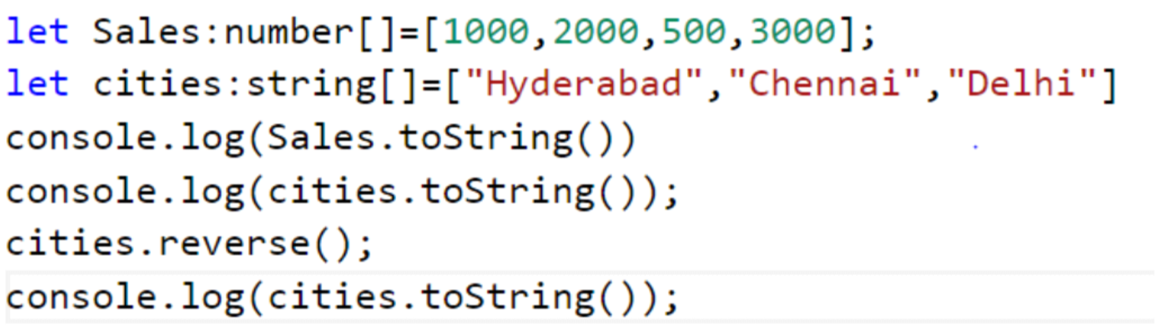
```
let price:number[]=[1000,2000,500,3000];
function GetPrice(cost)
{
    return cost>=2000;
}
let newPrice:number[]=price.filter(GetPrice);
console.log(newPrice.toString());
```

output:



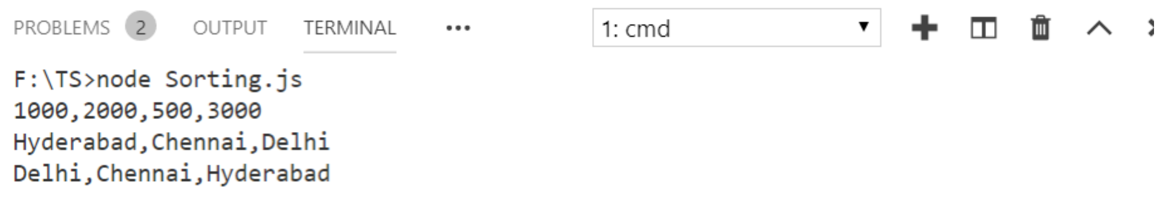
```
F:\TS>tsc filter.ts

F:\TS>node filter.js
2000,3000
```

Example: Sorting



```
let Sales:number[]=[1000,2000,500,3000];
let cities:string[]=["Hyderabad","Chennai","Delhi"]
console.log(Sales.toString())
console.log(cities.toString());
cities.reverse();
console.log(cities.toString());
```

output:



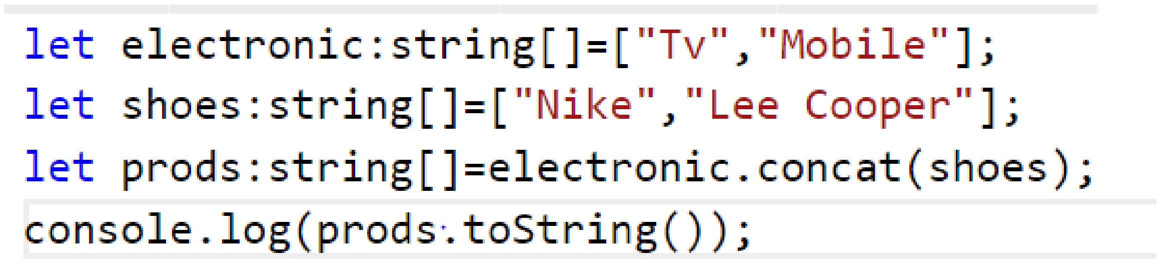
```
F:\TS>node Sorting.js
1000,2000,500,3000
Hyderabad,Chennai,Delhi
Delhi,Chennai,Hyderabad
```

Sorting Array Element:

Function	Description
Sort( )	A Function is sort Arranges array elements in ascending order
reverse( )	Arranges array element in the reverse order ie from bottom to top.

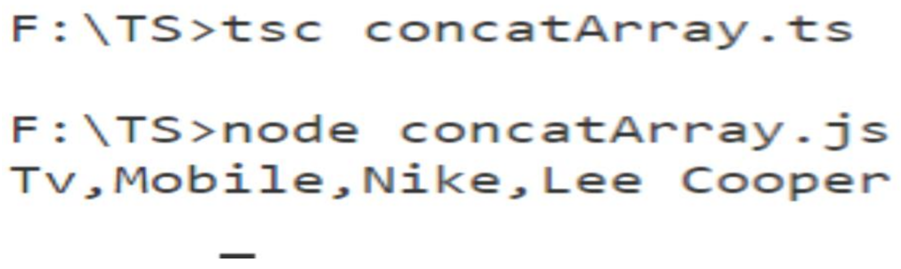
6.concat Array

you can combine Array elements from Multiple Arrays and Store in a Single Array.The function "concat( )" is used for combining Array elements.



```
let electronic:string[]=["Tv","Mobile"];
let shoes:string[]=["Nike","Lee Cooper"];
let prods:string[]=electronic.concat(shoes);
console.log(prods.toString());
```

output:



```
F:\TS>tsc concatArray.ts

F:\TS>node concatArray.js
Tv,Mobile,Nike,Lee Cooper
```

Q.How do we declare to handle any Type of elements?

Ans:Object encapsulation a set of members logical Entity that hold the code.Object reference to class and reference.Object is a collection of members,It contain Members,Properties and Methods.These properties and Methods are related to object Events are related to Browser.Events are associated with browser not object Events are used by object.The behaviour of object is exihibited building an object is a set of properties and Methods.

Object Type:

1. JavaScript is an object based Programming.
2. TypeScript is an object oriented programming
3. In both Programming approaches object is an logical Entity that encapsulates a set of Properties and Methods.
4. An object allows to reuse the components without redefing.
5. An object encapsulates the members in "{   }".
6. Object is by default static type hence its members are accessed directly by refering to Object name.

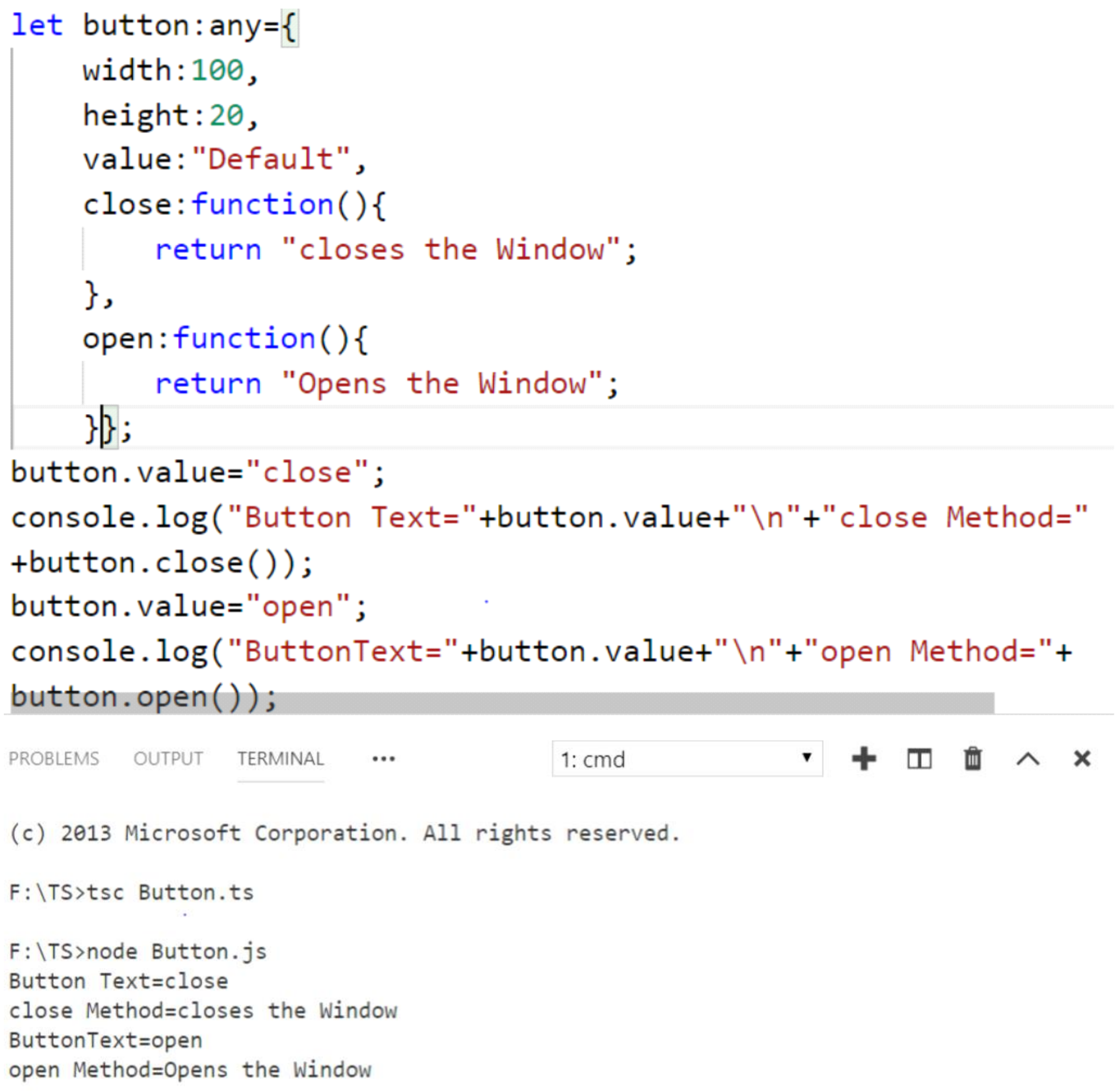
```
let button:any={
                                width:100,
           height:20,
                                value="Default"
         close:function( ){
                               return "closes the window";
                               },
                               open:function( ){
                               return "Opens the Window";
                              },
```

button.value="close";

console.log("Button Text="+button.Value+"\n",+"Close Method="+button.close( ));

button.Value="open";

console.log("Button Text="+button.Value+"\n"+"open Method"+button.open( ))



```
let button:any={
    width:100,
    height:20,
    value:"Default",
    close:function(){
        return "closes the Window";
    },
    open:function(){
        return "Opens the Window";
    }};
button.value="close";
console.log("Button Text="+button.value+"\n"+"close Method="
+button.close());
button.value="open";
console.log("ButtonText="+button.value+"\n"+"open Method="+
button.open());
```

PROBLEMS OUTPUT TERMINAL ··· 1: cmd

```
(c) 2013 Microsoft Corporation. All rights reserved.

F:\TS>tsc Button.ts

F:\TS>node Button.js
Button Text=close
close Method=closes the Window
ButtonText=open
open Method=Opens the Window
```

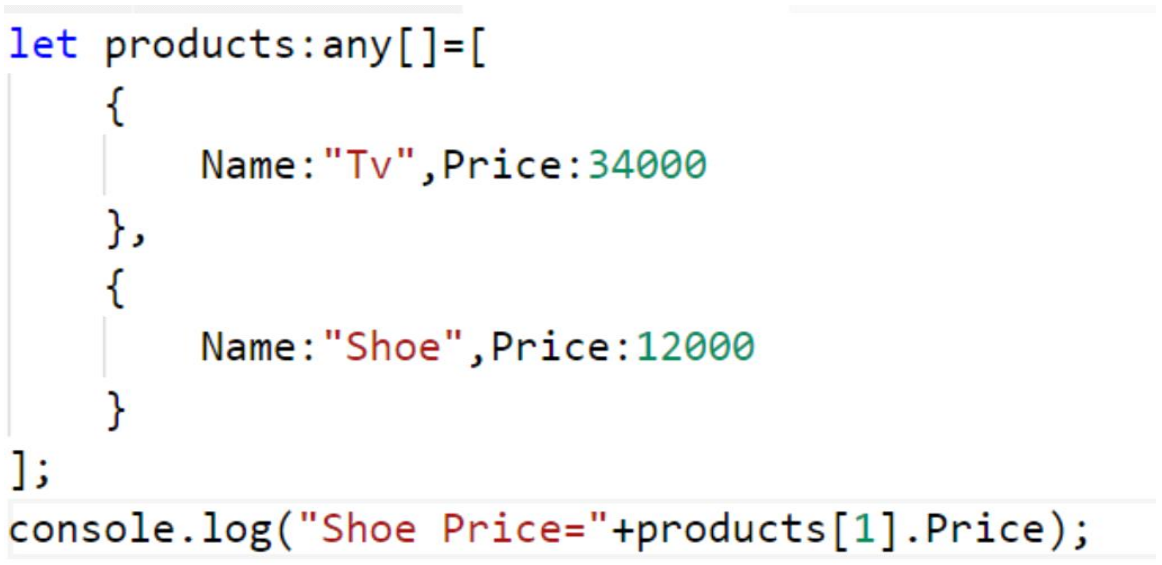
Json Type:

Json type will be Array of object.

Array of object[JSON-JavaScript object Notation]

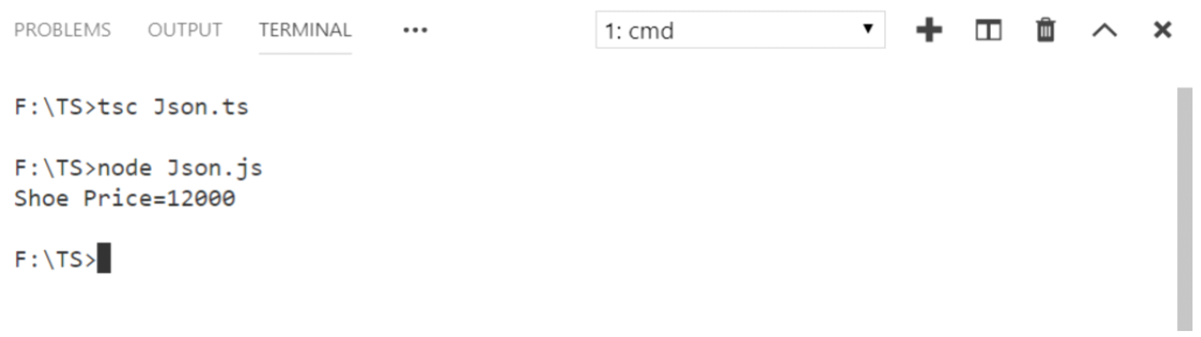
"json.ts"

```
    let products:any[ ]=[
    {
Name:"Tv",Price:34000
Name:"Shoe",Price:12000
}
];
console.log("Shoe Price="+products[1].Price);


},
```



```
let products:any[]=[
    {
        Name:"Tv",Price:34000
    },
    {
        Name:"Shoe",Price:12000
    }
];
console.log("Shoe Price="+products[1].Price);
```

output:



```
F:\TS>tsc Json.ts

F:\TS>node Json.js
Shoe Price=12000

F:\TS>
```

Regular Expression:

isNaN( )->is not a Number

var x=10;

even or odd

var mobile ="+91 10 digits"

It uses a pattern

pattern referes to a format.patterns are built using Meta characters and Quantifiers.patterns are enclosed in '//'.pattern refers to a format.* represents zero or many.Regular Expression are always verifief with match( ) function.

3.Regular Expression Type

1.A Regular Expression is used to verify the format of input value.It uses a pattern that is enclosed in "/ /"

2.The pattern are built by using Meta Characters and Quantifiers.

3.The commonly used Meta characters ara shown below:

Meta Characters	Description
^	It matches the starting position within a string
.	It matches any single character or Exactly specified character.

\	It refer any Entity within the Expression.
$	It matches the ending position within a string.
*	It matches the string preciding with zero or more Elements
?	It matches the preciding element zero or one time
+	It matches the preciding element one or more times
w	It matches a word that is alpha numeric with(under score) _
W	It represents a Non-word
d	It Specifies a Number with decimal places
D	It is to refer UnSigned Number.
s	allows leading spaces.
S	no space allowed.
[ ]	It defines range or Specific Value.
[A-Z]	only uppercase letters allowed.

[a-z]	only lowercase allowed.
[a-zA-Z] or [a-Z]	Both Upper & Lower case .
[0-9]	Only numeric.
[a-zA-Z0-9]	or \w alpha numeric
[a,d,f]	only specified characters
[^a,d,f]	exclude specified chars.
[a-mA-M4-9]	chars in specified range.
\+\-\-\$\@	special chars group
[!@#$%&]	special char group
(?=.*[A-Z])	Atlest one uppercase letter
(?=.*[0-9])	Atleast one number
(?=.*[!@#$%&]	Atlest one special character.

Qunatifiers	**Description**
{n}	Exactly n-number of chars
{n,m}	minimum-n,maximum-m
{n, }	minimum-n,maximum any
{4}	
{4,10}	
{ 4 }	

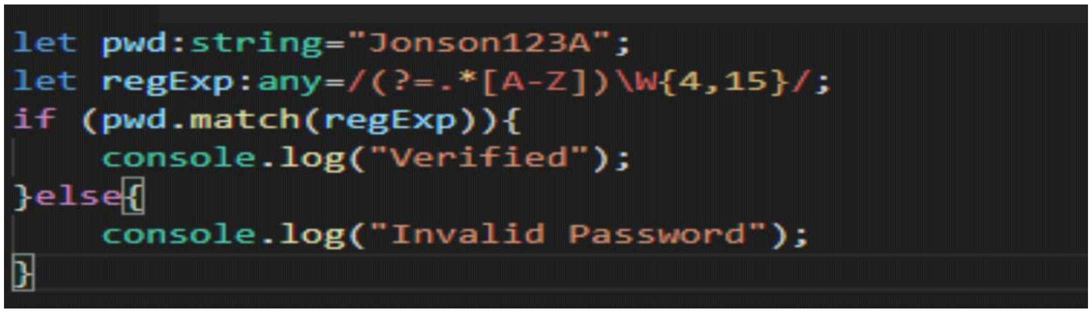
Syntax:

let regExp:any=/[A-Z]{4,10}|;

Note: A regular Expression is verified with the value using "match( )" function.It is a boolean function that   returns true,when the expression matched.

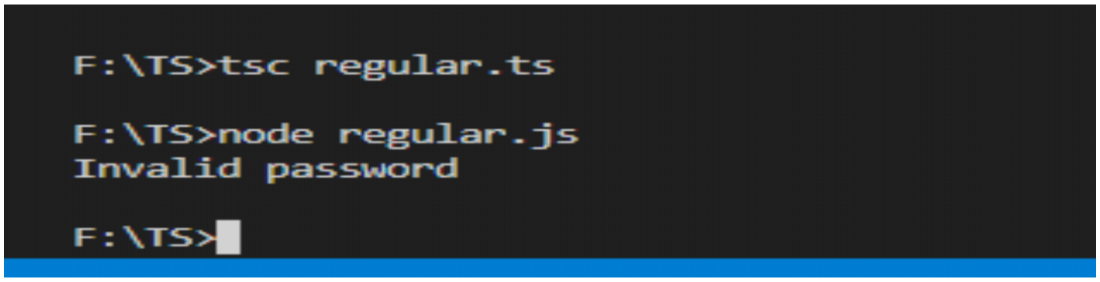
Example:- Regular Expression

```
let pwd:string="Jonson123";

let regExp:any=/(?=.*[A-Z])\w{4,15};

  if(pwd.match(regExp)){

  console.log("Verified");

}

else{

console.log("Invalid Password")

}
```

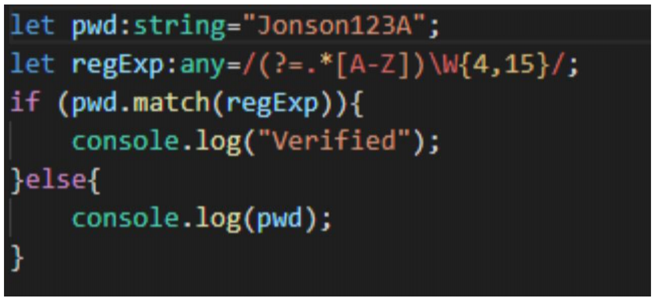


```
let pwd:string="Jonson123A";
let regExp:any=/(?=.*[A-Z])\W{4,15}/;
if (pwd.match(regExp)){
    console.log("Verified");
}else{
    console.log("Invalid Password");
}
```

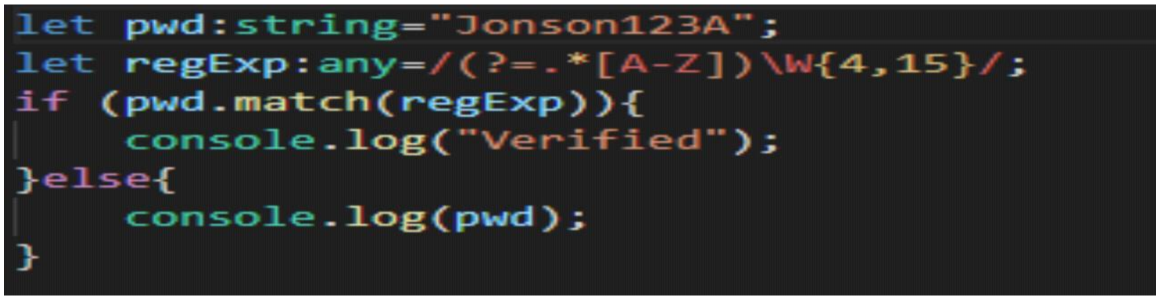
output:



```
F:\TS>tsc regular.ts

F:\TS>node regular.js
Invalid password

F:\TS>
```

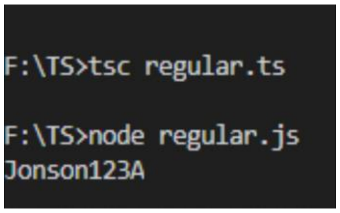


```
let pwd:string="Jonson123A";
let regExp:any=/(?=.*[A-Z])\W{4,15}/;
if (pwd.match(regExp)){
    console.log("Verified");
}else{
    console.log(pwd);
}
```



```
let pwd:string="Jonson123A";
let regExp:any=/(?=.*[A-Z])\W{4,15}/;
if (pwd.match(regExp)){
    console.log("Verified");
}else{
    console.log(pwd);
}
```

output:



```
F:\TS>tsc regular.ts

F:\TS>node regular.js
Jonson123A
```

TypeScript DateTime:

I. A DateTime can hold a date and Time value.TypeScript uses any Type to store date value.The Date value is stored in in memory as string allocated by date constructor "Date( )"

II. Syntax: let mfd:any=new Date("2019/05/06")

III. The date values are accessed and used by using Date( ).Date function which return a date and time number.

Date Function	**Description**
getHour( )	Returns the hour number in 24hr format.
getMinutes( )	Returns the Minutes number

fromTime.

getSeconds( ) Returns the Second Number.

getDate( ) Returns the Current date number ie "7"

[05/07/2019]

getDay( ) Return the weekday number 0=Sunday

getMonth( ) Return Month number 0=Jan

getFullYear( ) Return full year 2019.

getYear( ) Return full year as per y2k/19.

toLocale DateString( ) Return shortdate

toLocale TimeString( ) Return shorttime

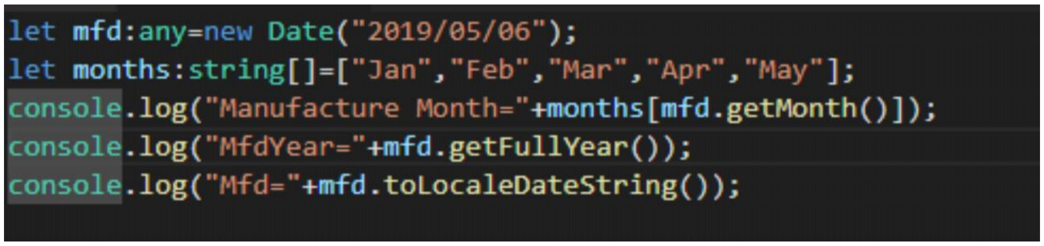
let mfd:any=new Date("2019/05/06");

let month:string[ ]=["Jan","Feb","Mar","Apr","May"];

console.log("Manufacture Month="+months[mfd.getMonth()]);

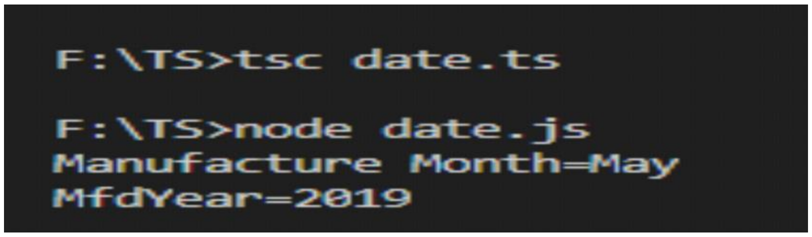
console.log("Mfd Year="+mfd.getFullYear( ));

console.log("Mfd="+mfd.toLocaleDateString( ));

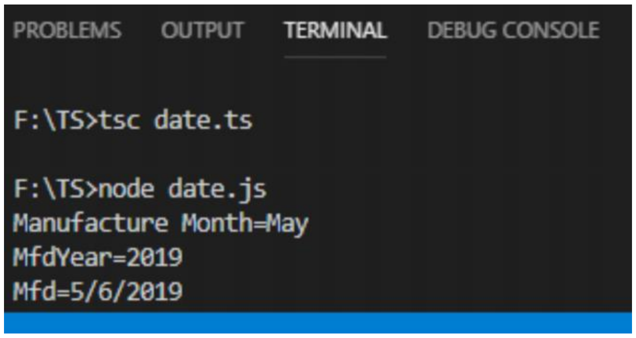


```
let mfd:any=new Date("2019/05/06");
let months:string[]=["Jan","Feb","Mar","Apr","May"];
console.log("Manufacture Month="+months[mfd.getMonth()]);
console.log("MfdYear="+mfd.getFullYear());
console.log("Mfd="+mfd.toLocaleDateString());
```

output:

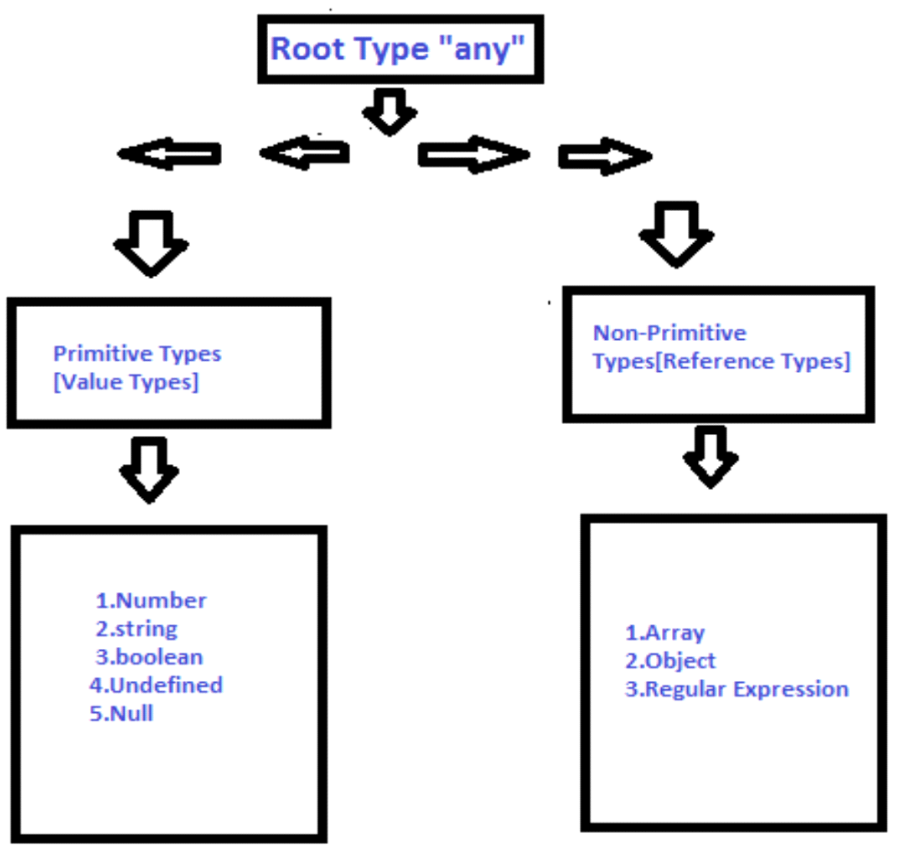


```
F:\TS>tsc date.ts

F:\TS>node date.js
Manufacture Month=May
MfdYear=2019
```



```
F:\TS>tsc date.ts

F:\TS>node date.js
Manufacture Month=May
MfdYear=2019
Mfd=5/6/2019
```

Summary of Data Types:



1.Variables

2.DataTypes

3.Operators-In computer programming operator is an object that evaluates an value by using operands.

4.Statements

Subtraction find difference value:

string-string

number-string

boolean-boolean=number

boolean-number=number

boolean+string=string

TypeScript Operators:

An operator in computer programming is an object that evaluates any value by using operands.

The Operator is defined as unary,binary and ternary based on the number of operands it uses

Single operand-Unary

Two operands -Binary

Three-Ternary

The operators are categorised into several groups based on the type of value they return.

1.Arthematic Operator

2.Comparision (Relational) Operator.

3.Bitwise Operator

4.Logical operator

5.Assignment Operator.

6.Special Operator.

Arthematic Operator:

character	**Description**	**Example**
+	Addition	10+20=30
-	Substraction	20-10=10
*	Multiplication	10*20=200
/	Division	20/10=2
%	Modulus	21%10=1

(Remainder)

++ Increment var a=10; a++;Now a=11;

-- Decrement var a=10; a--;Now a=9;

** Exponent 2**3=8

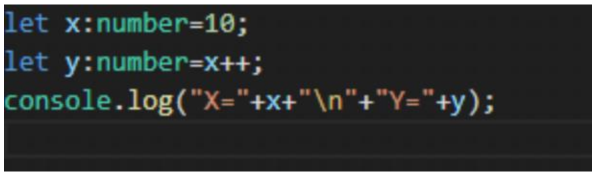
Post Increment:-

It assigns the initialised value and increment thecurrent value

let x:number=10;

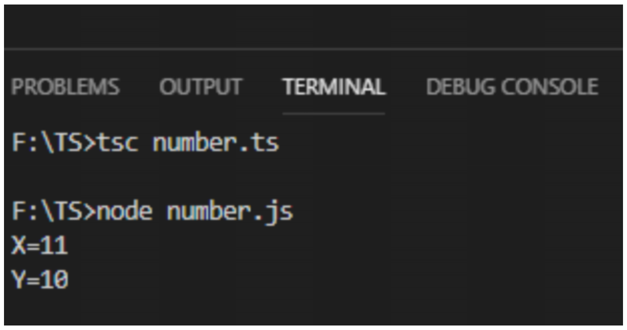
let y:number=x++;

console.log("X="+x+"\n"+"Y="+y);



```
let x:number=10;
let y:number=x++;
console.log("X="+x+"\n"+"Y="+y);
```

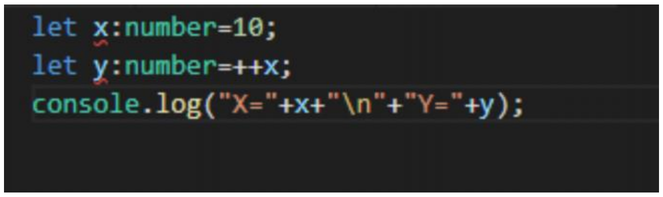
output:



```
F:\TS>tsc number.ts

F:\TS>node number.js
X=11
Y=10
```

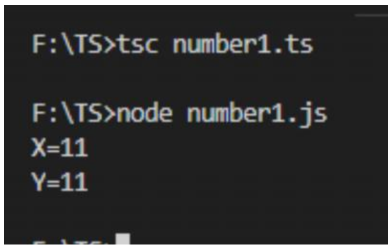
Pre-Increment:-

It Increments the intialized value by 1 and then assigns



```
let x:number=10;
let y:number=++x;
console.log("X="+x+"\n"+"Y="+y);
```

output



```
F:\TS>tsc number1.ts

F:\TS>node number1.js
X=11
Y=11
```

Exponent **   Math.pow(2,3)

Exponent: It is suppoerted in TypeScript not in JavaScript.Js will use "Math.pow( )".js will use "Math.pow( )" function for finding the base raised to power

Syntax: let   x:number=2**3; Ts-8

var x=Math.pow(2,3);Js-8

Comparision Operator:

Operator	Description	Examples
==	Isequalto	10==20=false
===	Identical(equal and of Same Type)	10===20=false
!=	NotEqualto	10!=20=true
!==	Not Identical	20!==20=false
>	Greater than	20>10=true
>=	GreaterThan or Equalto	

		20>=10=true
<	Lessthan	20<10=false
<=	Lessthan or Equalto	20<=10=false
Id===Id	Field name must be id and Type	
Id==Id	It will not compare type.it will only Compare Value.	

Identical Equal and not Equal Operator:

The Identical Equal "===" is used to compare values of same type.If they are different type then the condition always return false.

Example:
```
let x:number=10;
let y:number="10";
if(x==y)//Always false
```

The Identical not equal "!==" is to verify that both values are same type and they are not equal.

Example:
```
let x:number=10;
let y:number=10;
let y:string="10";
```

Bitwise Operators:

Operator	**Description**	**Example**
&	Bitwise AND	(10==20) &(20==33)=False
|	Bitwise OR	(10==20)|(20==33)=False
^	Bitwise XOR	(10==20)^(20==33)=True

~	Bitwise NOT	(~10)=-11
<<	Bitwise Left Shift	(10<<2)=40
>>	Bitwise Right Shift	(10>>2)=2
>>>	Bitwise Right Shift with zero	(10>>>2)=2

Logical Operators

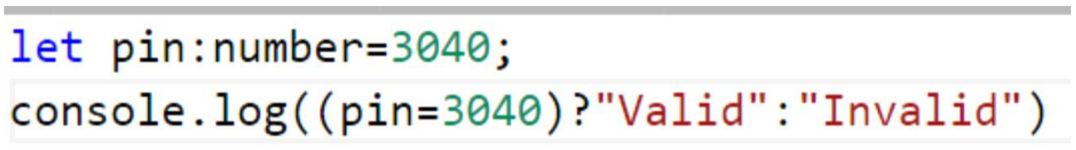
Operator	**Description**	**Example**
=	Assign	10+10=20
+=	Add and Assign	var a=10;a+=20; Now a=30;
-=	Subtract and Assign	var a=20;a=10; Now a=10
*=	Mutiply and Assign	Var a=10;a*=20; Now a=200
/=	Divide and Assign	Var a=10;a/=2; Now a=5;
%=	Modulus and Assign	Var a=10;a%=2; Now a=0

Special Operators:

ternary.ts

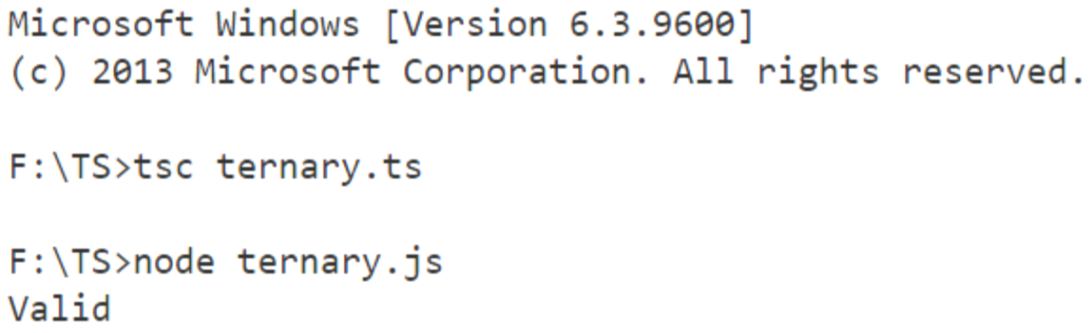
```
let pin:number=3040;
```

console.log((pin=3040)?"Valid":"Invalid")



```
let pin:number=3040;
console.log((pin=3040)?"Valid":"Invalid")
```

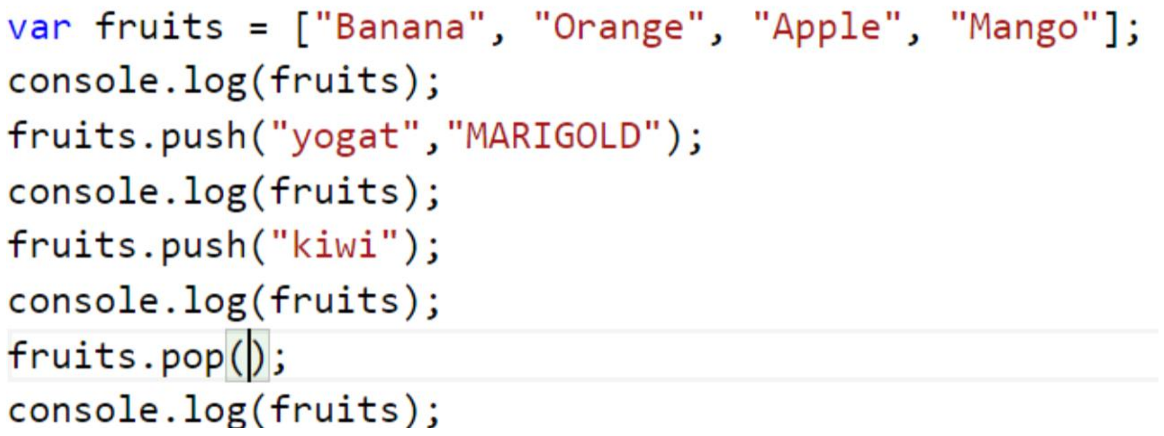
output:



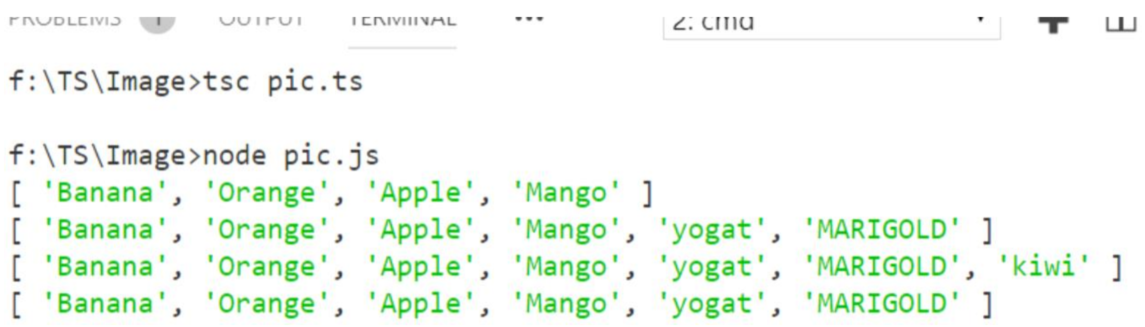
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

F:\TS>tsc ternary.ts

F:\TS>node ternary.js
Valid
```

It is dynamic memory allocating operator that allocates memory for specified object type and loads its members into memory.

Syntax:



```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
console.log(fruits);
fruits.push("yogat","MARIGOLD");
console.log(fruits);
fruits.push("kiwi");
console.log(fruits);
fruits.pop();
console.log(fruits);
```



```
f:\TS\Image>tsc pic.ts

f:\TS\Image>node pic.js
[ 'Banana', 'Orange', 'Apple', 'Mango' ]
[ 'Banana', 'Orange', 'Apple', 'Mango', 'yogat', 'MARIGOLD' ]
[ 'Banana', 'Orange', 'Apple', 'Mango', 'yogat', 'MARIGOLD', 'kiwi' ]
[ 'Banana', 'Orange', 'Apple', 'Mango', 'yogat', 'MARIGOLD' ]
```

Ternary Operator:

It is a Special Operator used to Evaluate a value based on Boolean Condition.

Syntax: (condition)?Value If True:Value If False

Example:

```
  let pin:number=3030;

console.log((pin==3040)? "Valid" :"Invalid");
```

2."new" Operator

"new" Dynamic memory alocating operator

It is a Dynamic Memory allocating operator that allocates Memory for Specified Object type and loads its members into Memory.

Syntax: let values:string[ ]=new Array( );

```
                         let Mfd:any=new Date( );

                         let pic:any=new Image( );

               values.length;

                          values.push( );

               Mfd.getMonth( );

                pic.src="Image/Car.jpg"
```

3.typeof Operator:

It is a Special Operator used to verify the typeof Value stored in a reference and return its datatype.

Syntax: let x:number=10;

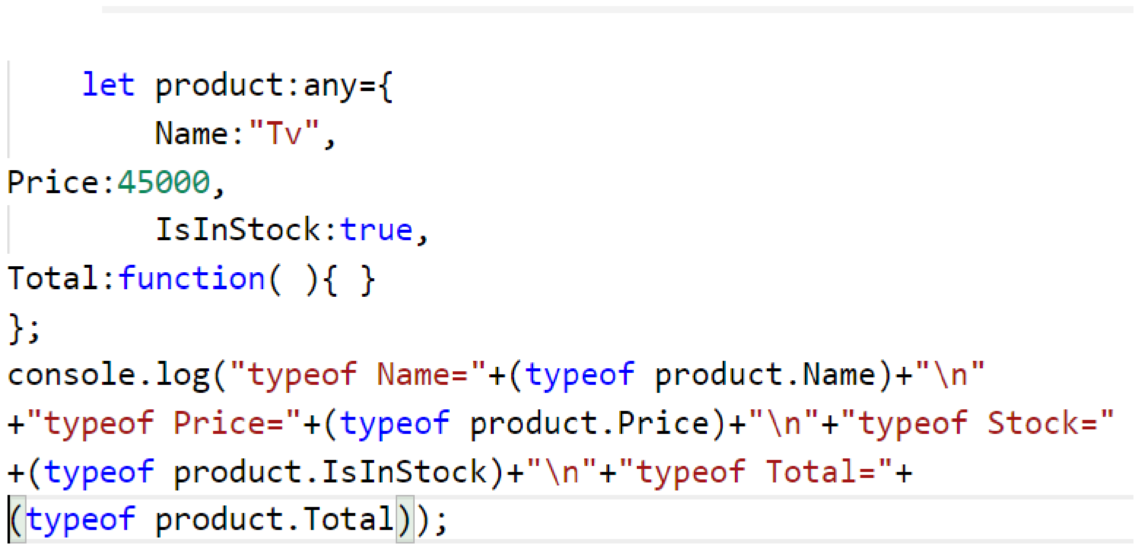
typeof x;

Example:

```
    let product:any={
                              Name:"Tv",
                  Price:45000,
                              IsInStock:true,
                  Total:function( ){ }
};
```

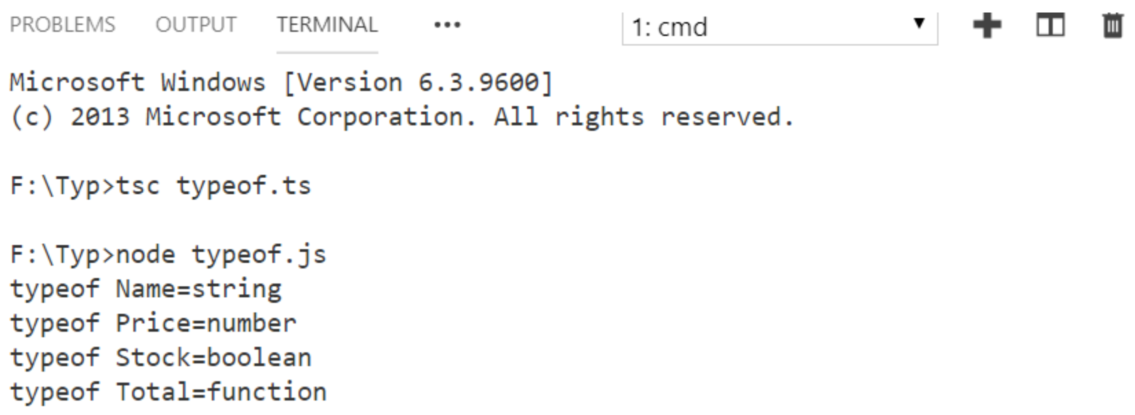
console.log("typeof Name="+(typeof product.Name)+"\n"+"typeof Price="+

(typeof product.Price)+"\n"+"typeof Stock="+(typeof product.IsInStock)+"\n"+"typeof Total="+(typeof product.Total));



```
    let product:any={
        Name:"Tv",
Price:45000,
        IsInStock:true,
Total:function( ){ }
};
console.log("typeof Name="+(typeof product.Name)+"\n"
+"typeof Price="+(typeof product.Price)+"\n"+"typeof Stock="
+(typeof product.IsInStock)+"\n"+"typeof Total="+
(typeof product.Total));
```

output:


PROBLEMS OUTPUT TERMINAL ··· 1: cmd

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

F:\Typ>tsc typeof.ts

F:\Typ>node typeof.js
typeof Name=string
typeof Price=number
typeof Stock=boolean
typeof Total=function
```

4.Instanceof Operator:

Instance of is Special Operator that Verifies Wheather the Specified Object is derived from given class.It returns Boolean true,If the Instance is Specified class type.

Syntax: yourObject instance of yourclass;

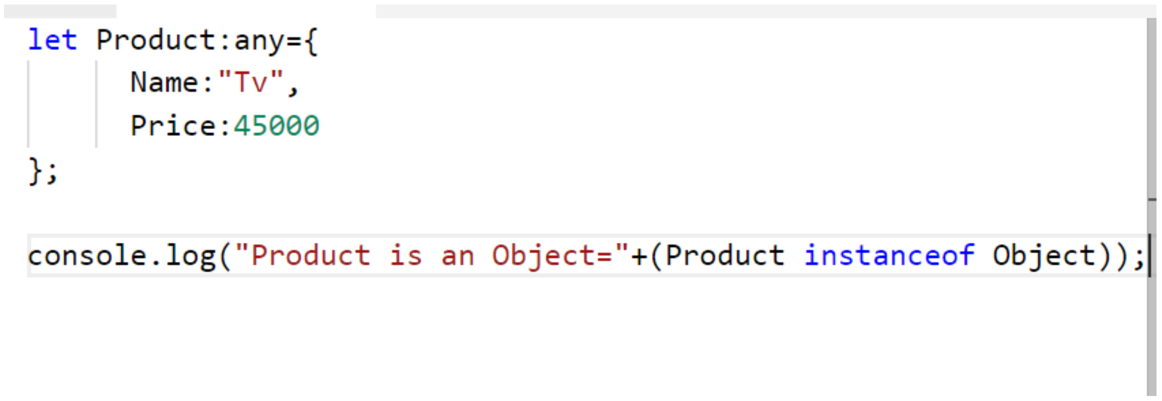
Example: let product:any={

Name:"Tv",

Price:45000

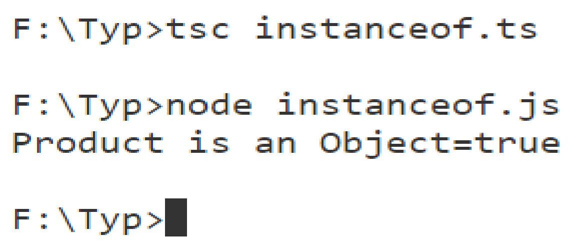
};

console.log("product is an object="+product instanceof Object));



```
let Product:any={
      Name:"Tv",
      Price:45000
};

console.log("Product is an Object="+(Product instanceof Object));
```

output



```
F:\Typ>tsc instanceof.ts

F:\Typ>node instanceof.js
Product is an Object=true

F:\Typ>
```

6) "in" Operator

It is a Special Operator used to verify whether the given property is avalible in an object it returns boolean false if property "Not Found" .

Syntax:

"PropertyName" in Object;

"Tv" in product

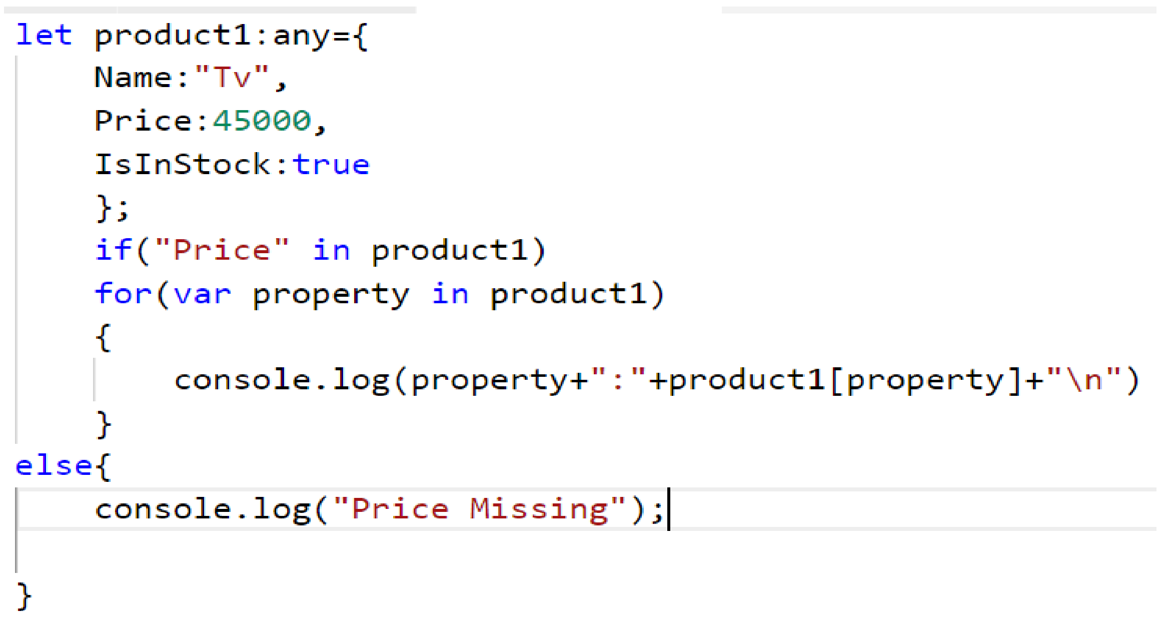
Tv is Value Name is a property

Example:

```
    let product:any={
                                  Name="Tv",
                                                          Price:45000,
                                  IsInStock:true
                                                              };
                                                        if("Price" in product){
                                                      for(var property in product){
                              console.log("property+":"+product[property]+"\n");
                                                        }
                                                        }
                                else{
```

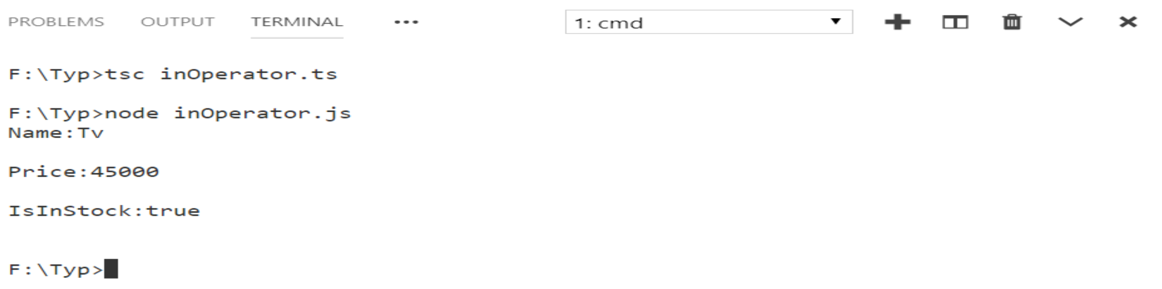
console.log("Price Missing");

}



```
let product1:any={
    Name:"Tv",
    Price:45000,
    IsInStock:true
    };
    if("Price" in product1)
    for(var property in product1)
    {
        console.log(property+":"+product1[property]+"\n")
    }
else{
    console.log("Price Missing");

}
```

output:



```
F:\Typ>tsc inOperator.ts

F:\Typ>node inOperator.js
Name:Tv

Price:45000

IsInStock:true


F:\Typ>
```

7)Of Operator:

It is a Special Operator   used to verify the values of an given object and return the values.

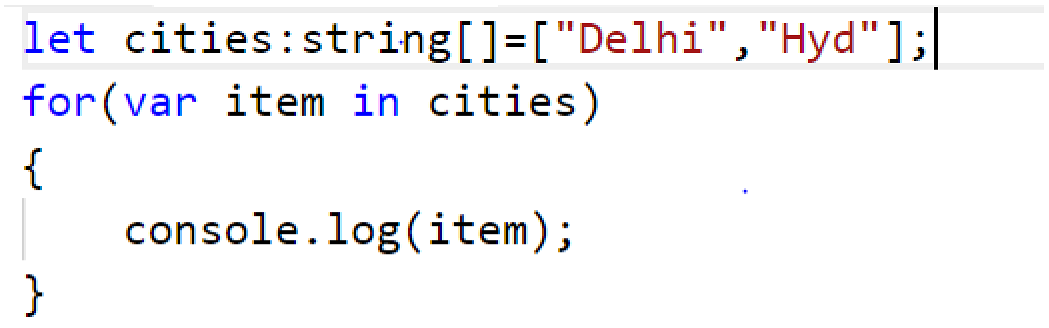
Example:

let cities:string[ ]=["Delhi","Hyderabad"]

for(var item of cities){

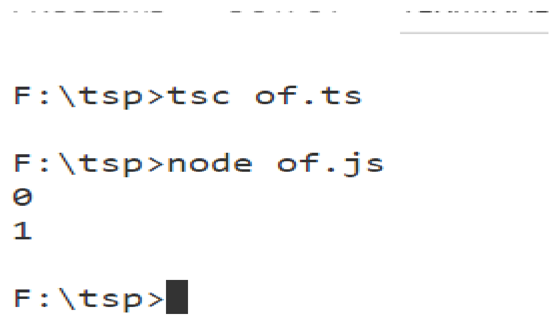
console.log(item);

}



```
let cities:string[]=["Delhi","Hyd"];
for(var item in cities)
{
    console.log(item);
}
```

output:



```
F:\tsp>tsc of.ts

F:\tsp>node of.js
0
1

F:\tsp>
```

Union Types:

- TypeScript is Strictly Typed
- You can define a reference to handle "any" type of value by using "any" datatype.
- TypeScript supports union Type that allows to create a reference that can handle various types that restricted with only specified.
- The union is defined by using pipe Symbol.

Create a refernce that can handle various types that restricted with only specified.

The Union is defined by using Pipe Symbol.

Example:

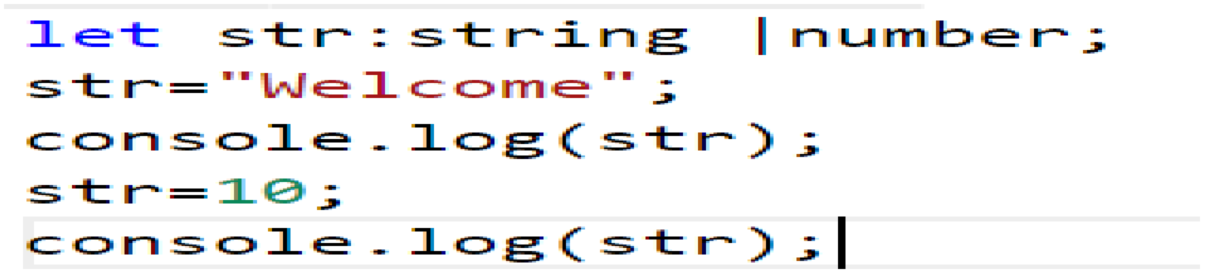
let str:string |number;

str="Welcome";

console.log(str);

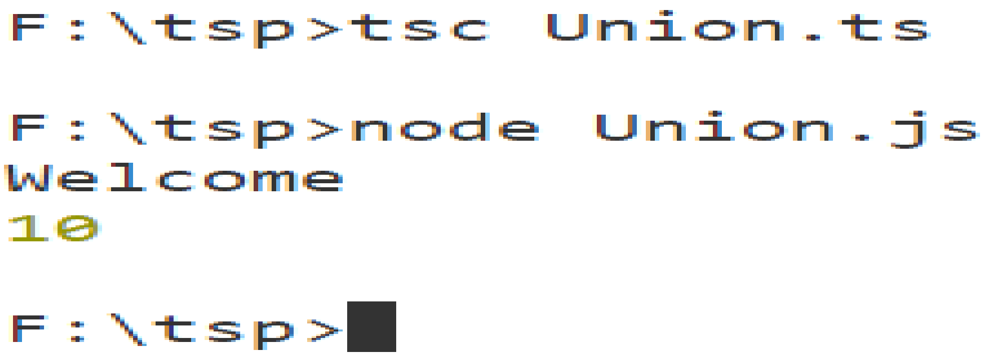
str=10;

console.log(str);



```
let str:string |number;
str="Welcome";
console.log(str);
str=10;
console.log(str);
```

output:



```
F:\tsp>tsc Union.ts

F:\tsp>node Union.js
Welcome
10

F:\tsp>
```

TypeScript Statements:

A Statement in computer programming is used to control the Execution flow.

The type of statements the TypeScript can Handle are categorized into following groups:

StatementType	Keyword
Selection	

	if.else,switch,case
Loop &Iteration Jump	for,while,dowhile
Jump	break,continue,return
Exception Handling	try,catch,throw,finally

TypeScript OOPS:

- TypeScript is an class based Object Oriented Programming
- JavaScript is an Object based Programming.while is psuedo OOP IE.OOP like Programming
- The OOP Concepts were Introduced in the Early 1960's by Jhon Olay & Nyguard
- OOP was formulated with a language called SIMULA-67
- The primary aim of OOP is to achieve code reusability
- In the early 1970's Trygve Introduced code seperation into Object Oriented Programming it was formulated with "SmallTalk"
- later th language c++ and Java made OOP Popular among developers

Features of OOP:

- Support Code Reusability
- Supports Code Extensibility
- Supports dynamic memory allocation
- Supports loosely Coupled Architecture

Drawback of OOP:

- It is Complex in Configuration
- It is Tidious and less Verbose.
- It cannot directly interact with hardware Services

- It needs more memory to operate

Characterstics of OOPS:

1. Inheritance
2. Polymorphism
3. Encapsulation
4. Abstraction

   code can be extended by a feature called Inheritance

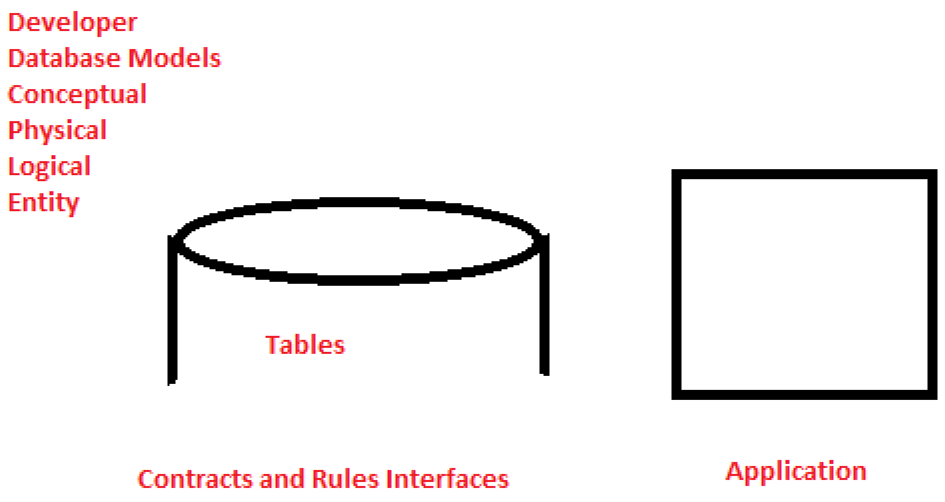
OOPS Terminology:

Term	**Description**
class	It is a Model or Blue print for creation of objects.It is a logical Entity that holds that encapsulates the logic
object	It is a physical representation of logical Entity
Properties	Defines the Physical Description of an object
Methods	Specifies the actions to be performed by an object
Event	Sends aMessage to the Subscriber inorder to notify change

Contracts and Rules:

- A contract in OOP Specifies the rules for designing a component
- Every component that implements any contract must be designed according to specified rules
- A contract can also define optional features,which are designated as optional for implementation
- A contract with optional features is also known as "Template"

- In oop "Interface" is used to define contract



Example: without contract

```
                    let product:[
          { Name:"Tv",Price:40000},
                                { Name :"Shoe",Price:"12000"},
];//Invalid
```

with contracts:

```
Interface IProduct{
                            Name:string;
                Price:number;
                           }
 let product:IProduct[ ]=[
              { Name:"Tv",Price:34000},
              {Name:"Mobile",Price:"4000"}
              ]//Invalid
```

Names is not a member

Price is number type.

? Nullable

Optional Members:

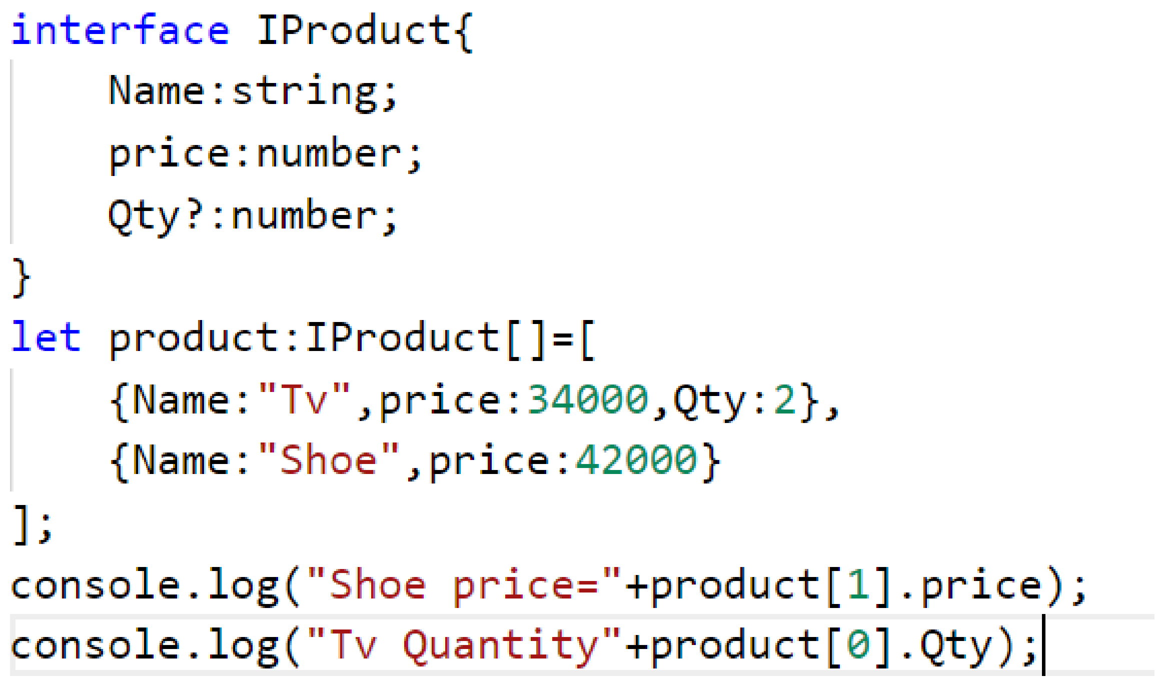
- Every member that you defined in an Interface is Mandatory to Implement
- You can configure optional Members by using Null refernce character "?"

Synatax:

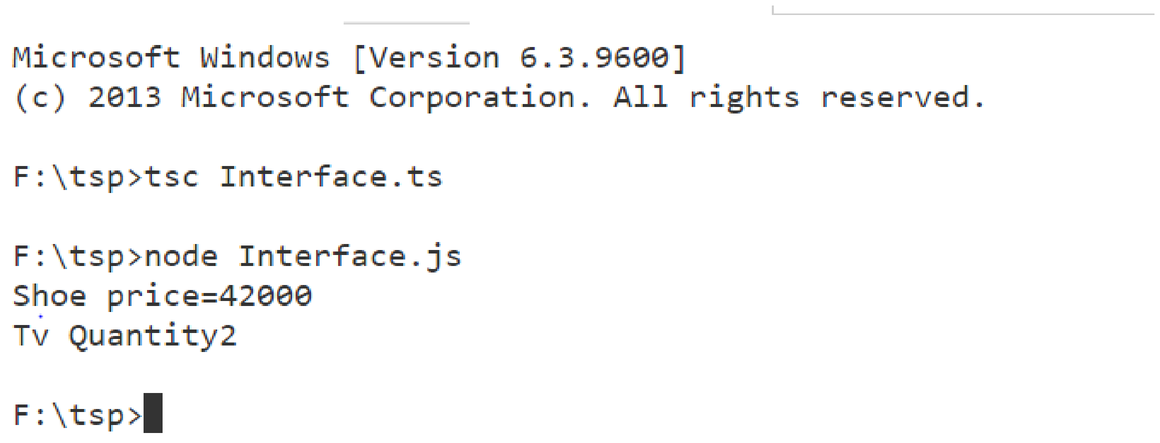
```
   Interface Iname{
  property?:type;//optional
  property:type; //required
 }
```

Example:-

```
interface Iname{
Name:string;
Price:number;
Qty?:number;
}
let product:IProduct[ ]=[
{Name:"Tv",Price:34000,Qty:2},
{Name:"Shoe",Price:42000}
];
console.log("Shoe Price="+product[1].price);
console.log("Tv Quantity="+product[0].Qty);
```



```
interface IProduct{
    Name:string;
    price:number;
    Qty?:number;
}
let product:IProduct[]=[
    {Name:"Tv",price:34000,Qty:2},
    {Name:"Shoe",price:42000}
];
console.log("Shoe price="+product[1].price);
console.log("Tv Quantity"+product[0].Qty);
```



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

F:\tsp>tsc Interface.ts

F:\tsp>node Interface.js
Shoe price=42000
Tv Quantity2

F:\tsp>
```

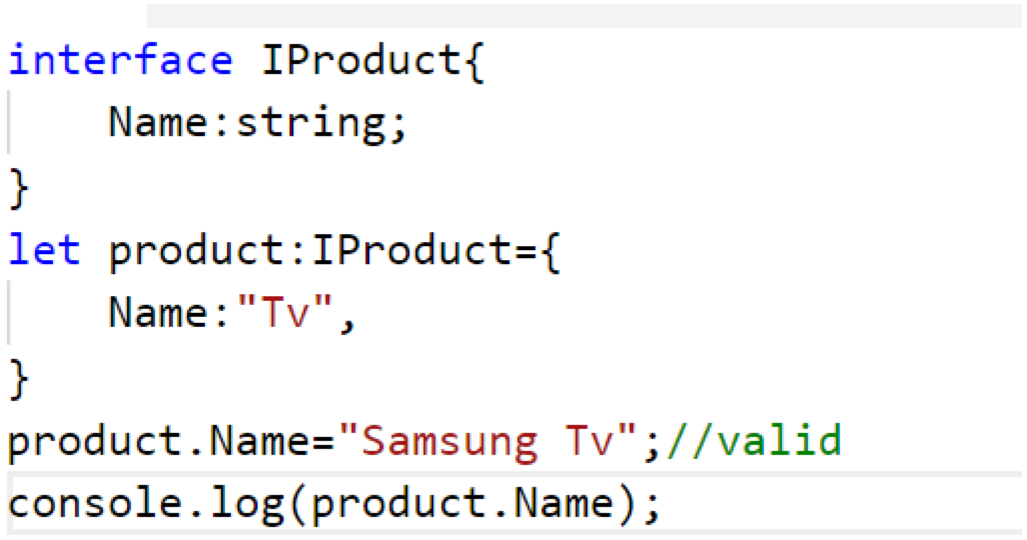
Every member in Interface is overridable by default you can restrict that by using Readonly once initialized with value we cannot redefine that we have to redefine and show an error as it is Readonly property.Readonly member cannot be overridden.

Readonly Members:

Interface members are by default overridable.ie you can redefine the value after Initialization.

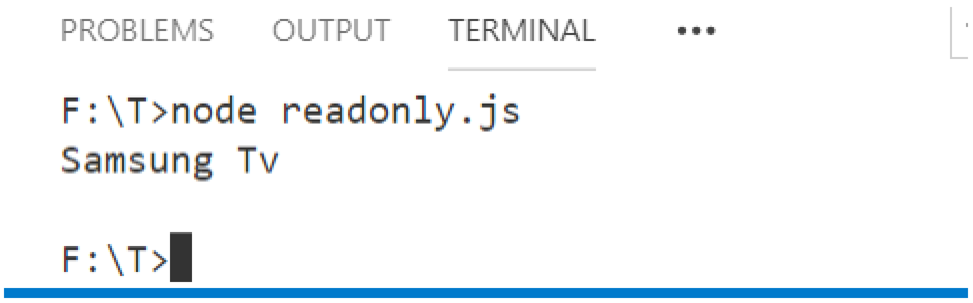
Syntax:

```
interface IProduct{
        Name:string;
          }
let product:IProduct={
              Name="Tv";
        }
product.Name="Samsung Tv";//valid
console.log(product.Name);
```



```
interface IProduct{
    Name:string;
}
let product:IProduct={
    Name:"Tv",
}
product.Name="Samsung Tv";//valid
console.log(product.Name);
```

output:


PROBLEMS OUTPUT TERMINAL ...

```
F:\T>node readonly.js
Samsung Tv

F:\T>
```

you can prevent redefining of values after Initialization by configuring them as Readonly.

Syntax:

```
   Interface Iproduct{

   readonly Name:string;

  }

let product:IProduct={

                Name="Tv"

}

product.Name="Samsung Tv";//Invalid

console.log(product.Name);
```

Interface Methods:

- A Method the behaviour of an Object
- In TypeScript method is a function that is configured with any functionality
- Interface Methods can be defined without return type by using the keyword void.
- The return type specified datatype for return value of function

Syntax: Method( ):Type{

```
                return Type;
```

```
                    }
                    Method( ):void{
                   };
```

Interface methods can have only Signature and not the definition.

```
Method( ):void{ }
Method:void ->Signature
{ }->Definition
interface IProduct{
        Name:string;
        Price:number;
        print( ):void;
}
```

Example: Interface Method

```
Interface IProduct{
        Name:string;
        Price:number,
        Print?( ):void
}
let product:IProduct={
Name:"Tv",
Price:34000.50
print:function( ){
console.log("Name="+product.Name+"\n"+"Price="+product.Price);
}
```

};

console.log(product.Print( ));

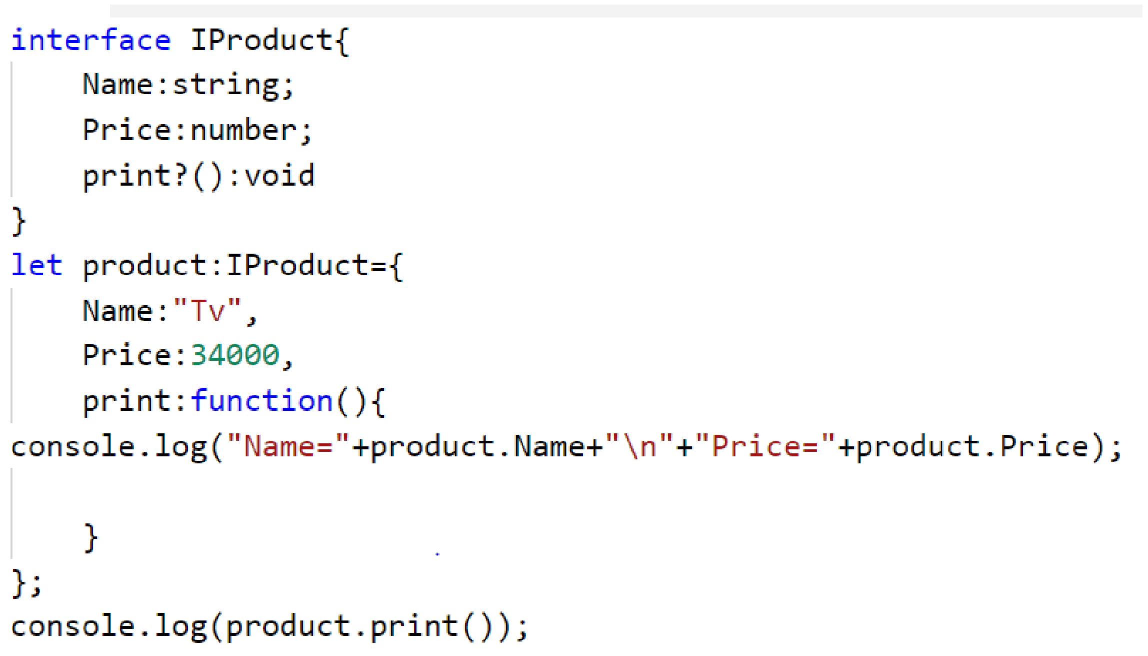
oracle,Sql Sever-Stored procedures parameters(how many we can pass)

1024->paramters we can pass

while implementing formal parameters

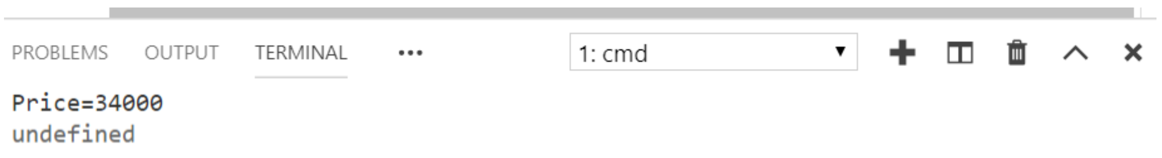
calling function-actual parameters

formal parameter doesnot contain value only reference



```
interface IProduct{
    Name:string;
    Price:number;
    print?():void
}
let product:IProduct={
    Name:"Tv",
    Price:34000,
    print:function(){
console.log("Name="+product.Name+"\n"+"Price="+product.Price);

    }
};
console.log(product.print());
```

output:



```
Price=34000
undefined
```

Interface Methods with return type and parameterized:

- The return type specifies the data type of value returned by function
- The Keyword "return" is used to return and to return the value and

terminate execution

- break; will terminate block but continue the function
- return; any statement we return is not reachable to compiler

```
  function f1( ){
  console.log("Statm-1");
console.log("Statm-2");
console.log("Statm-3");
}
f1( );
```

return;//it is used for returing stuf

stuf are code blocks.which compiler cannot reach that return is not returning a value.we have to write condition and break;

return;   // Jump Statements

break; //Jump Statements

break; Continues even after we break

Example:

```
interface IProduct{
                    Name:string;
                    Price:number;
                    qty:number;
  Total?(qty:number,price:number,price:number):number;
Print?( ):void
}
let product:IProduct ={
```

```
Name:"Tv",

Price:34000.50,

qty:2,

Total:function(qty,price){

return qty*price;

};

print:function( ){

console.log("Name="+product.Name+"\n"+"qty="+product.qty+"\n"+"Tota="+pro
duct.Total(product.price));

}

};

console.log(product.Print( ));
```