Welcome

Type Script Essentials
By Debangshu Nag

Course content

- Overview
- Setup of Environment & Editor
- Syntax
- Types
- Variables
- Operators
- Decision statement
- Loops

Course content (Cont..)

- Functions
- Numbers
- Strings
- Arrays
- Tuples
- Unions
- Interfaces
- Classes

Course content briefly (Cont..)

- Objects
- Namespace
- Modules
- Modules Vs Namespaces
- Next Step..

Overview

- What is Typescript?
  - TypeScript is a strongly typed, object oriented, compiled language.
  - It was designed by Anders Hejlsberg (designer of C#) at Microsoft.
  - TypeScript compiled into javascript.
- Why Use TypeScript?
  - Compile time error detection: JavaScript is an interpreted language & error is got detected at run time but typescript is transpilled or compiled and transpiler of typescript 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. Which helps developers to write a code like in high level language like Java or C# etc.

Overview (Cont..)

- Supports of OOPs: TypeScript supports Object Oriented Programming concepts like classes, interfaces, inheritance, etc.

Environment Setup

- Online Editor to run Typescript & get equivalent js
  - https://www.typescriptlang.org/play/
- Editors
  - Visual Studio Code
  - Sub lime Text
  - Note pad++
- How to transpile & run code?
  - Typescript file need to be saved in .ts extension file. Say program.ts
  - To generate .js file command is : tsc program.ts

Environment Setup (Cont..)

- The previous command will generate program.js file.

- Node Js
  - Node.js is an open source, cross-platform runtime environment for server-side JavaScript.
  - Node.js is required to run JavaScript without a browser support.
  - It uses Google V8 JavaScript engine to execute code.
  - We can download nodejs from https://nodejs.org/en/download/
- NPM: NPM stands for Node package Manager. Npm package manager is a tool which will allow you to install third party libraries (other people's code) by using the command line. Example: npm install express
- Install Typescript Globally: npm install -g typescript

Environment Setup (Cont..)

- How to install typescript locally?
  - Go to the target folder & then create package.json: npm init
  - Install Typescript locally & save to package.json as development dependency: npm install -s typescript
  - Now open package.json & remove "main" & replace "scripts": { tsc": "tsc”}
  - To check version of the typescript: npm run tsc -- -v
  - To compile any code: npm run tsc prog1.tsc
  - Check the output: node prog1.js

Syntax

- Let's create our first program name: prog1.ts

Now to compile or transpile the program: tsc prog1.ts

To execute the program: node prog1.js

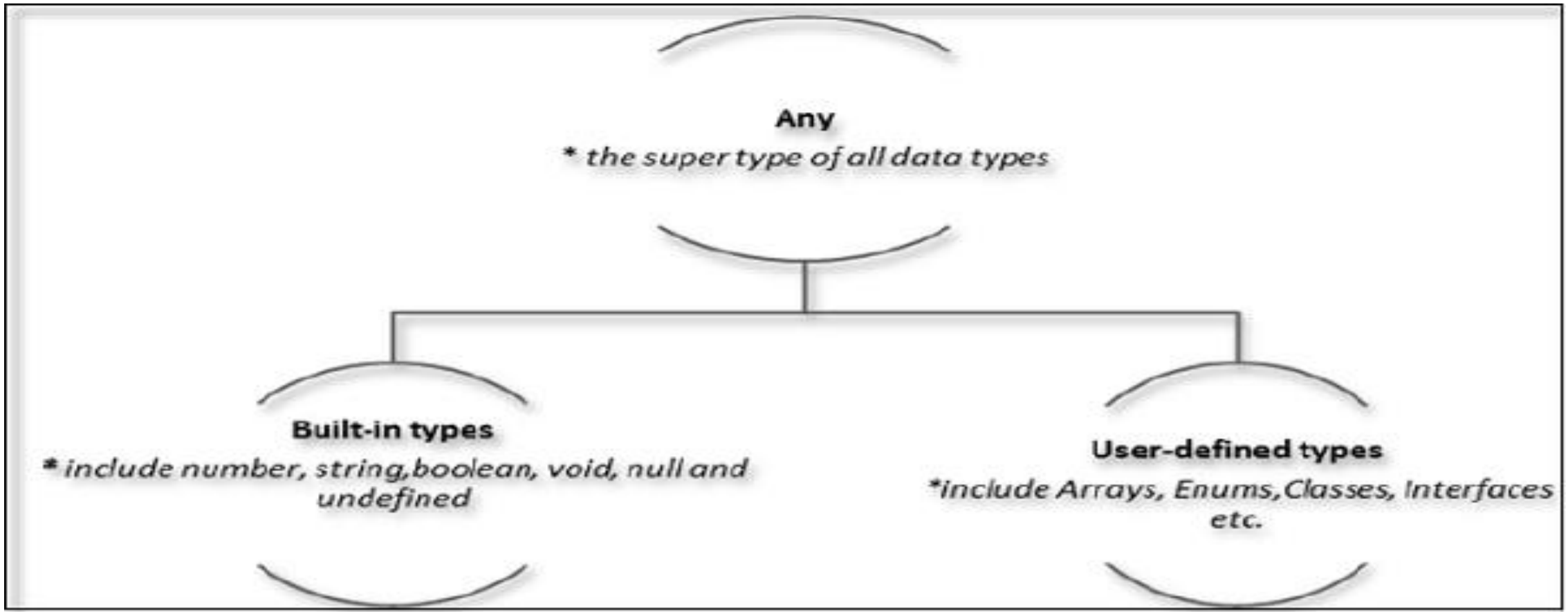
To compile & execute single line command: tsc prog1.ts && node prog1.js

- Multiple files can be compiled as: tsc prog1.ts, prog2.ts, prog3.ts
- Naming convention is same like Java e.g: firstName, first_name etc.

Syntax (Cont..)

- It supports all the keywords like java with some new like module, export, yield etc.
- TypeScript is case-sensitive.
- Semicolons are optional in Typescript.
- It supports two 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.

Types


Any
* the super type of all data types
Built-in types
* include number, string,boolean, void, null and undefined
User-defined types
*include Arrays, Enums,Classes, Interfaces etc.

Variables

- var name:string = ”mary”
  - The variable stores a value of type string
- var name:string;
  - The variable is a string variable. The variable’s value is set to undefined by default
- var name = ”mary”
  - The variable’s type is inferred from the data type of the value. Here, the variable is of the type string
- var name;
  - The variable’s data type is any. Its value is set to undefined by default.

Variables (Cont..)

- Lets try prog2.ts
- var num:number = "hello" // will result in a compilation error
- Type Assertion in TypeScript
  - It allows changing a variable from one type to another. TypeScript refers to this process as Type Assertion. The syntax is to put the target type between < > symbols and place it in front of the variable or expression.
  - “type assertions” are purely a compile time construct and a way for you to provide hints to the compiler on how you want your code to be analyzed.
  - Lets try prog3.ts

Variables (Cont..)

- Inferred Typing in TypeScript
  - TypeScript also encourages dynamic typing of variables. This means that, TypeScript encourages declaring a variable without a type. In such cases, the compiler will determine the type of the variable on the basis of the value assigned to it.
  - Lets check prog4.ts.
  - Its throwing error by transpiler because we are trying to assign a string to a number variable.

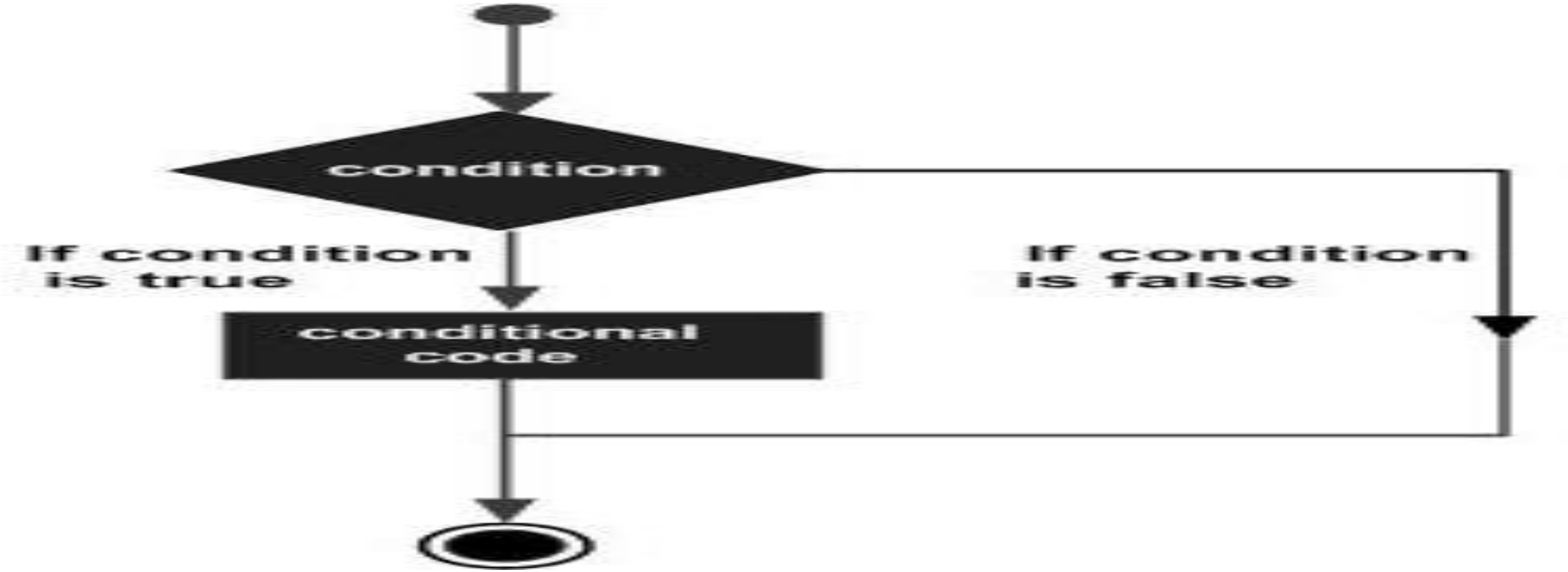
Variables (Cont..)

- TypeScript Variable Scope
  - Global Scope − These variables can be accessed from anywhere within your code.
  - Class Scope − Fields or class variables are declared within the class but outside the methods. These variables can be accessed using the object of the class. Fields can also be static. Static fields can be accessed using the class name.
  - Local Scope − Local variables, as the name suggests, are declared within the constructs like methods, loops etc. Local variables are accessible only within the construct where they are declared.
  - Lets check prog5.ts

Operators

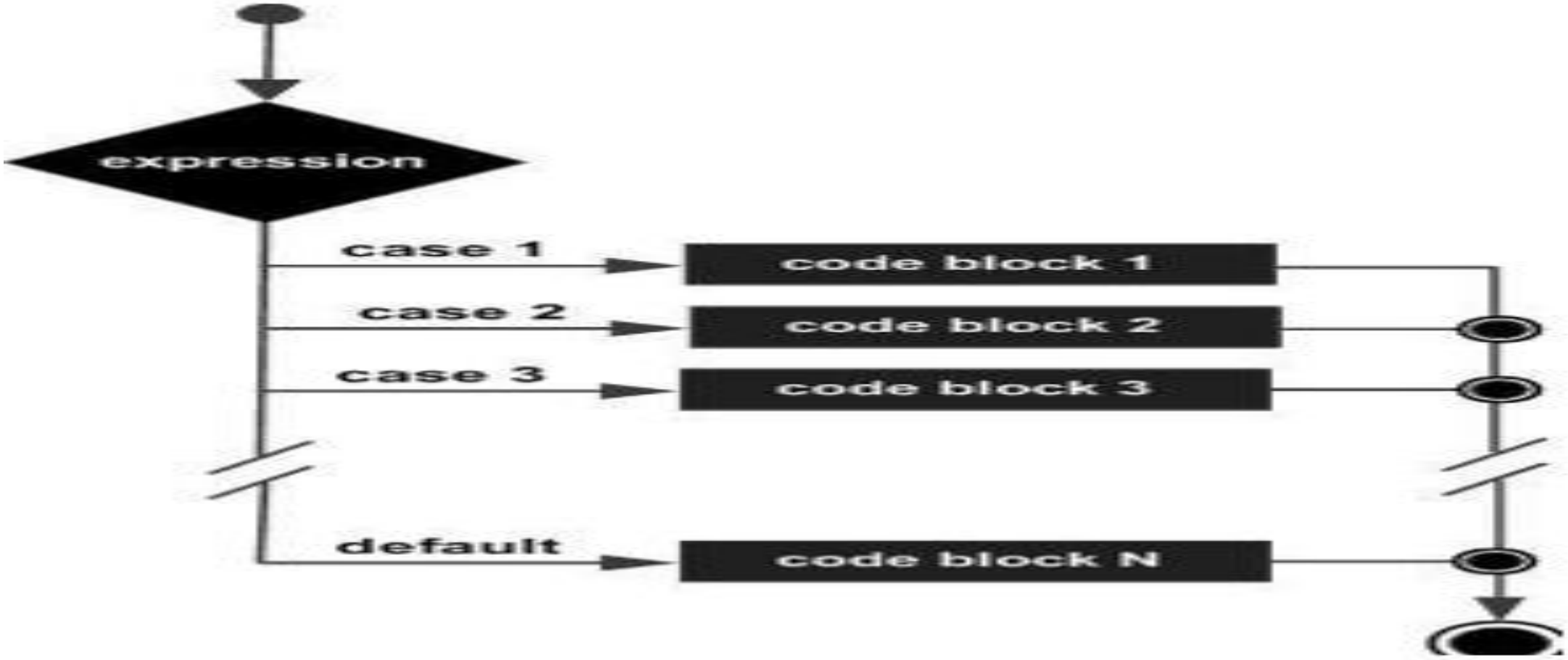
- Arithmetic operators
- Logical operators
- Relational operators
- Bitwise operators
- Assignment operators
- Ternary/conditional operator
- String operator
- Type Operator
- Lets Check prog6.ts

Decision Statement (IF ELSE)



Lets check prog7.ts

Decision Statement (SWITCH CASE)



Lets check prog8.ts

Loops

```
for (initial_count_value; termination-condition; step) {
   //statements
}
while(condition) {
   // statements if the condition is true
}
do {
   //statements
} while(condition);
```

Lets check Prog9.ts

Functions

- Functions are the building blocks of readable, maintainable, and reusable code. A function is a set of statements to perform a specific task. Functions organize the program into logical blocks of code. Once defined, functions may be called to access code. This makes the code reusable. Moreover, functions make it easy to read and maintain the program's code.
- Lets check prog10.ts

Numbers

❖TypeScript like JavaScript supports numeric values as Number objects. A number object converts numeric literal to an instance of the number class. The Number class acts as a wrapper and enables manipulation of numeric literals as they were objects.

❖ Lets check prog11.ts

Strings

❖The String object lets you work with a series of characters. It wraps the string primitive data type with a number of helper methods

❖charAt() : Returns the character from the specified index. Characters in a string are indexed from left to right. The index of the first character is 0, and the index of the last character in a string, called **stringName**, is **stringName.length – 1**.

❖indexOf() : Returns the index within the calling String object of the first occurrence of the specified value, starting the search at **fromIndex** or **-1** if the value is not found.

❖split(): Splits a String object into an array of strings by separating the string into substrings.

Strings (Cont..)

- substr(): Returns the characters in a string beginning at the specified location through the specified number of characters.
- substring(): Returns the new sub-string based on given parameters.
- toLowerCase(): Returns the calling string value converted to lowercase.
- toUpperCase(): Returns the calling string value converted to uppercase.
- Lets check prog12.ts

Arrays

- concat() method returns a new array comprised of this array joined with two or more arrays.
- forEach() method calls a function for each element in the array.
- join() method joins all the elements of an array into a string.
- map() method creates a new array with the results of calling a provided function on every element in this array.
- pop() method removes the last element from an array and returns that element.
- push() method appends the given element(s) in the last of the array and returns the length of the new array.

Arrays (Cont..)

- reverse() method reverses the element of an array. The first array element becomes the last and the last becomes the first.
- shift()method removes the first element from an array and returns that element.
- unshift() method adds one or more elements to the beginning of an array and returns the new length of the array.
- sort() method sorts the elements of an array.
- splice() method changes the content of an array, adding new elements while removing old elements.
- reduce() method applies a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.

Tuples

- It represents a heterogeneous collection of values. In other words, tuples enable storing multiple fields of different types. Tuples can also be passed as parameters to functions.
- Tuple support destructing assignment.
- Lets check prog15.ts

Unions

- TypeScript 1.4 gives programs the ability to combine one or two types. Union types are a powerful way to express a value that can be one of the several types. Two or more data types are combined using the pipe symbol (|) to denote a Union Type. In other words, a union type is written as a sequence of types separated by vertical bars.
- Lets check prog16.ts

Interfaces

- An interface is a syntactical contract that an entity should conform to. In other words, an interface defines the syntax that any entity must adhere to.
- Lets check prog17.ts

Classes

- A class is a blueprint of an object or it can be considered as an object factory.
- TypeScript supports the concept of Inheritance. Inheritance is the ability of a program to create new classes from an existing class. The class that is extended to create newer classes is called the parent class/super class. The newly created classes are called the child/sub classes.A class inherits from another class using the ‘extends’ keyword. Child classes inherit all properties and methods except private members and constructors from the parent class.
- Method Overriding is a mechanism by which the child class redefines the super class’s method.
- The static keyword can be applied to the data members of a class. A static variable retains its values till the program finishes execution. Static members are referenced by the class name.

Classes (Cont..)

- The instanceof operator returns true if the object belongs to the specified type.
- Access specifier
  - public: A public data member has universal accessibility. Data members in a class are public by default.
  - private: Private data members are accessible only within the class that defines these members. If an external class member tries to access a private member, the compiler throws an error.
  - protected: A protected data member is accessible by the members within the same class as that of the former and also by the members of the child classes.
- Classes can also implement interfaces.
- Lets check prog18.ts

Objects

- It is blueprint of a class.
- Duck typing: In duck-typing, two objects are considered to be of the same type if both share the same set of properties. Duck-typing verifies the presence of certain properties in the objects, rather than their actual type.
- Lets check prog19.ts

Namespaces

- A namespace is a way to logically group related code. This is inbuilt into TypeScript unlike in JavaScript where variables declarations go into a global scope and if multiple JavaScript files are used within same project there will be possibility of overwriting or misconstruing the same variables, which will lead to the “global namespace pollution problem” in JavaScript.
- A namespace definition begins with the keyword namespace followed by the namespace name.
- The classes or interfaces which should be accessed outside the namespace should be marked with keyword export.To access the class or interface in another namespace, the syntax will be namespaceName.className

Namespaces (Cont..)

- The classes or interfaces which should be accessed outside the namespace should be marked with keyword export.To access the class or interface in another namespace, the syntax will be namespaceName.className
- If the first namespace is in separate TypeScript file, then it should be referenced using triple slash reference syntax.
  - /// <reference path = "SomeFileName.ts" />
- Here we have created two files Ishapen.ts & drawingn.ts.
- To compile: tsc –out outputFileName.js startingfile.ts(drawing.ts)
- To run: node outputFileName.js

Modules

- A module is a way to create a group of related variables, functions, classes, and interfaces, etc. It executes in the local scope, not in the global scope. In other words, the variables, functions, classes, and interfaces declared in a module cannot be accessible outside the module directly. We can create a module by using the export keyword and can use in other modules by using the import keyword.
- Here we have created two files Ishapem.ts & drawingm.ts
- To compile: tsc drawingm.ts
- To run: node drawingm.js

Modules vs Namespaces

Module	Namespace
It is also known as an external module.	It is also known as an internal module.
A module can declare their dependencies.	Namespaces cannot declare their dependencies.
We can use a module in other modules by using the import keyword.	The namespace must be included in a file by using triple-slash (///) reference syntax. e.g. /// <reference path="path to namespace file"/>

Next step

Understanding TypeScript - 2020 Edition by Maximilian Schwarzmüller from Udemy

Thank You!