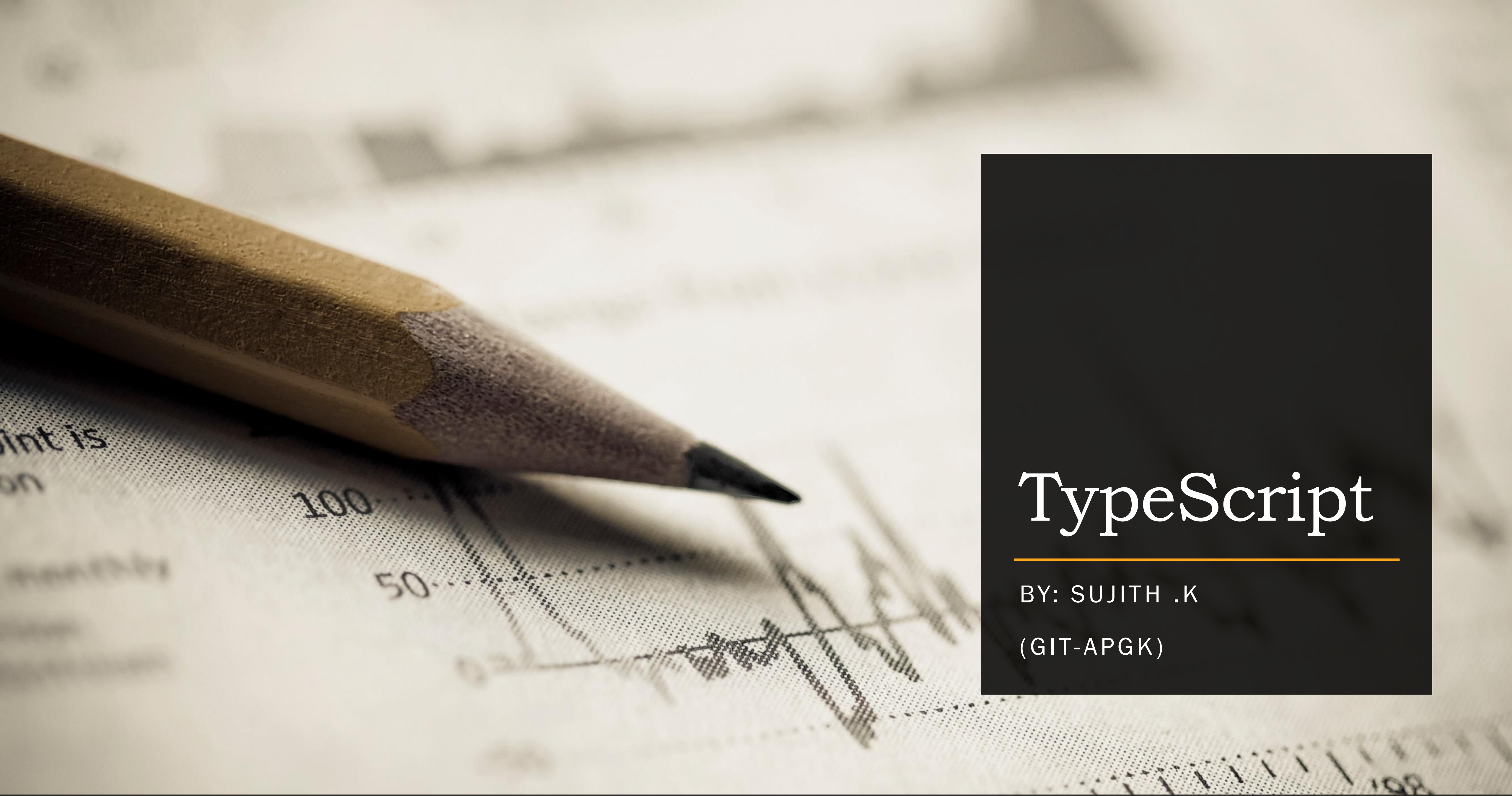


TypeScript

BY: SUJITH .K

(GIT-APGK)

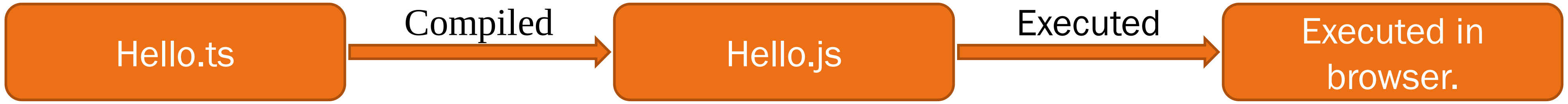
What is TypeScript?

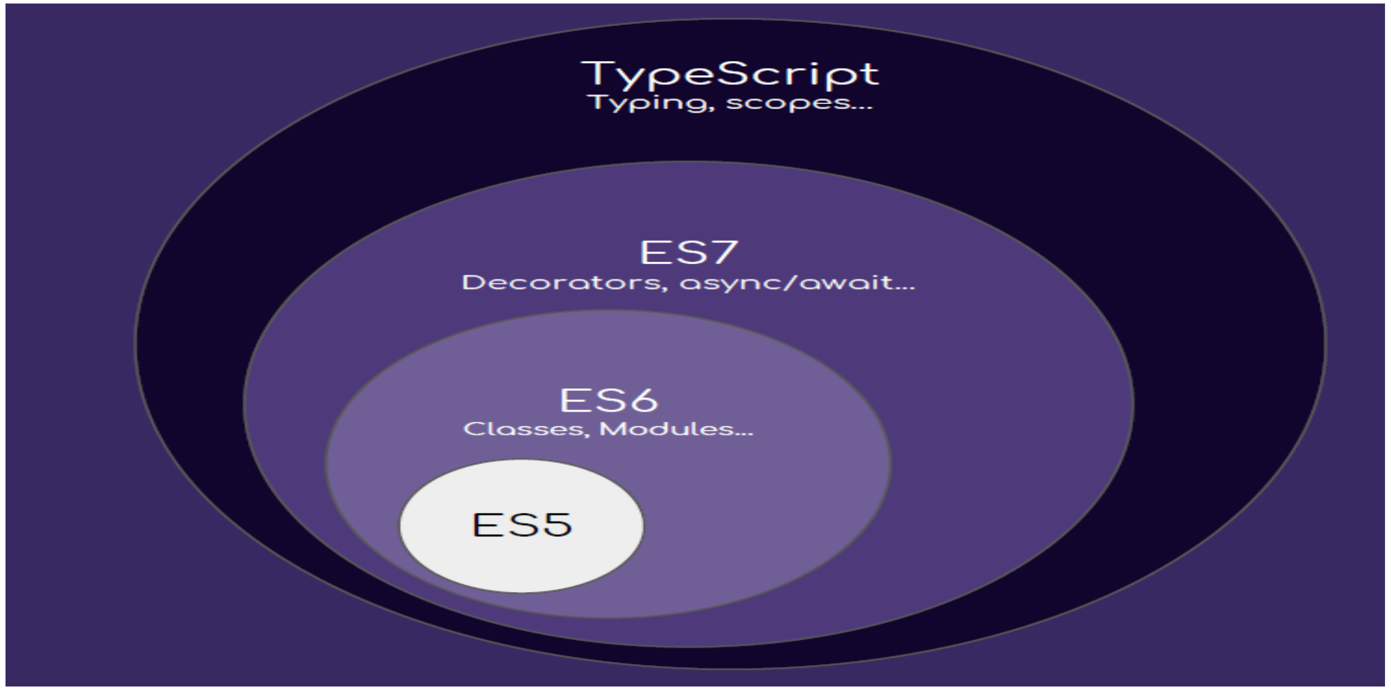
- The word TypeScript means, Specifying and providing more focus on types.
- TypeScript lets you write JavaScript the way you really want to.
- TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
- Any browser. Any host. Any OS. Open Source.

Why use TypeScript?

- TypeScript provides more flexibility and helps to write more error free code.
- The main reason is the difference between JavaScript and TypeScript, that is JavaScript is Function oriented programming language where, TypeScript is more like Object Oriented Programming language which supports classes, interfaces, generics, annotations, public/private/protected access and more.
- This help to make the code modular and reduces the code repetition.

How TypeScript work?





How to install TypeScript ?

- Download and install NodeJS: https://nodejs.org/en/download/

```
>> node --version
```

- Open terminal and type the command has follows:

```
>> tsc --version
>> npm install typescript -g
```

Setup environment for TypeScript?

Open VS Code editor → Go to Extensions → Search for "**JavaScript and TypeScript Nightly**"

→ install that package in the Editor.

Now to Compile and Run your file:

```
>> tsc [filename].ts
>> node [filename].js
```

To reduce the compilation for each change we can type:

```
>> tsc --init
>> tsc --watch
>> node [filename].js
```

Data Types

- Any
- Primitive:
  - Number
  - Boolean
  - String
  - Void
  - Null
  - Undefined - Same as JS
- Array
- Enum

Hello Word Program

Hello.ts

```
var message:string = "Hello World"
console.log(message)
```

$\rightarrow$

Hello.js

```
//Generated by typescript 1.8.10
var message = "Hello World";
console.log(message);
```

Declaring variables in TypeScript.

```
var name:string = "John";
var score1:number;
score1 = 50;
var score2 = 42.50;
var sum;
sum = score1 + score2;
console.log("name"+name);
console.log("first score: "+score1);
console.log("second score: "+score2);
console.log("sum of the scores: "+sum);
```

```
var name = "John";
var score1;
score1 = 50;
var score2 = 42.50;
var sum;
sum = score1 + score2;
console.log("name" + name);
console.log("first score: " + score1);
console.log("second score: " + score2);
console.log("sum of the scores: " + sum);
```

What is Interface?

- Interfaces define properties, methods, and events, which are the members of the interface.
- Interfaces contain only the declaration of the members.
- It is the responsibility of the deriving class to define the members.
- Syntax:

```
interface interface_name { }
```

Example:

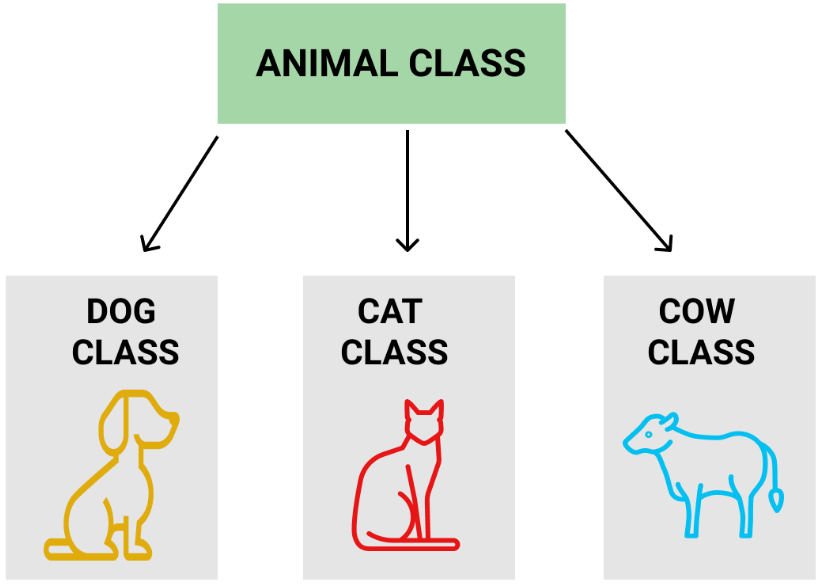
```
interface IPerson {
   firstName:string,
   lastName:string,
   sayHi: ()=>string
}

var customer:IPerson = {
   firstName:"Tom",
   lastName:"Hanks",
   sayHi: ():string =>{return "Hi there"}
}

console.log("Customer Object ")
console.log(customer.firstName)
console.log(customer.lastName)
console.log(customer.sayHi())

var employee:IPerson = {
   firstName:"Jim",
   lastName:"Blakes",
   sayHi: ():string =>{return "Hello!!!"}
}

console.log("Employee  Object ")
console.log(employee.firstName);
console.log(employee.lastName);
```

OUTPUT:

```
Customer object
Tom
Hanks
Hi there
Employee  object
Jim
Blakes
Hello!!!
```

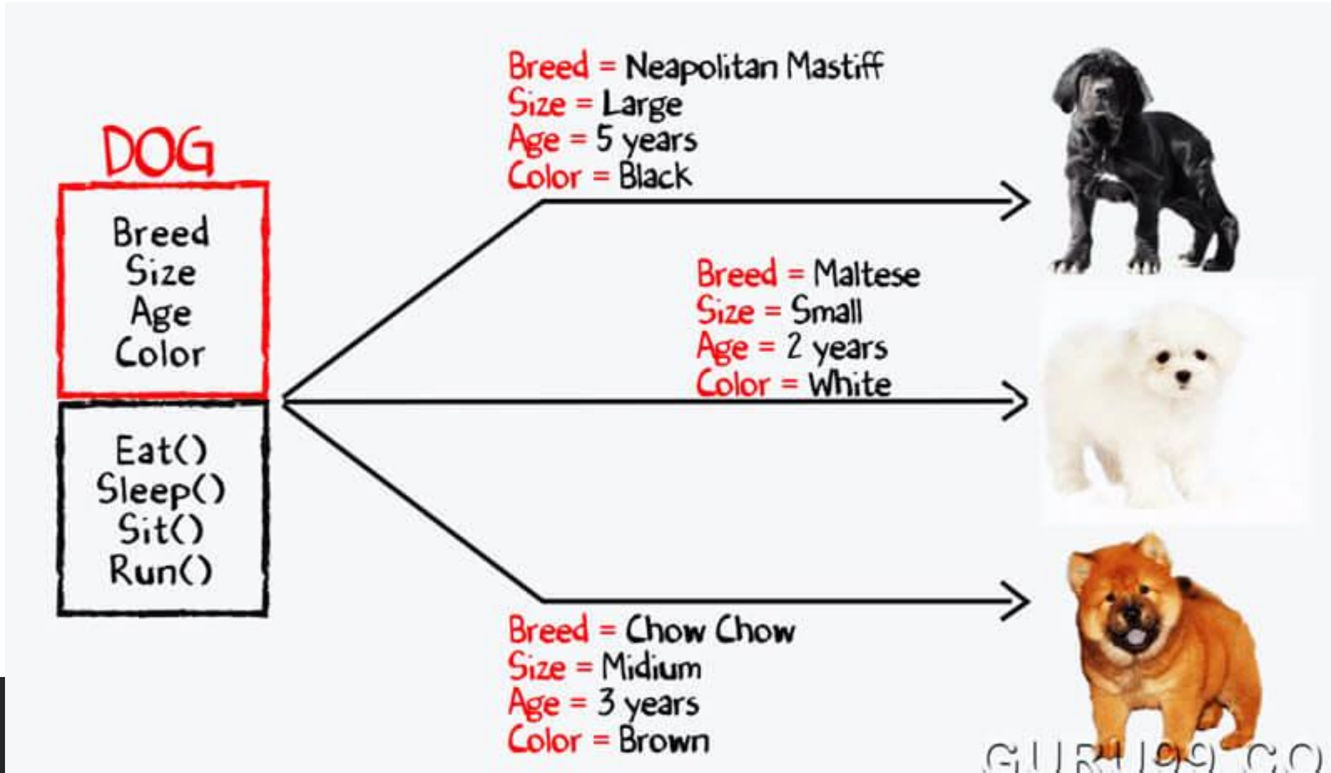
What is Class?

- A class in terms of OOP is a blueprint for creating objects.
- A class encapsulates data for the object.
- Typescript gives built in support for this concept called class.
- JavaScript ES5 or earlier didn’t support classes.
- Typescript gets this feature from ES6.

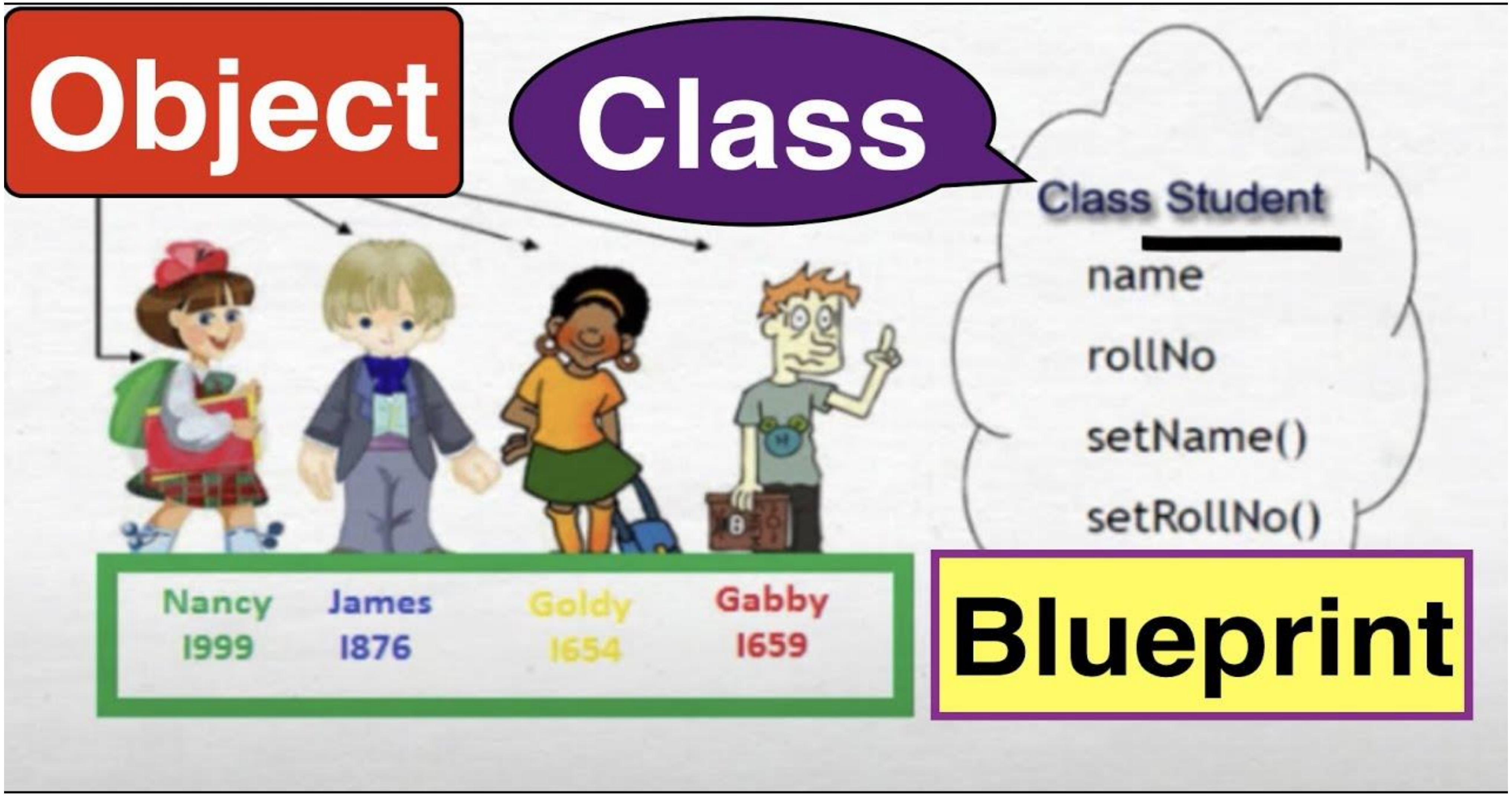


What is Object?

- An **object** is an instance which contains set of key value pairs.
- The values can be scalar values or functions



Example:

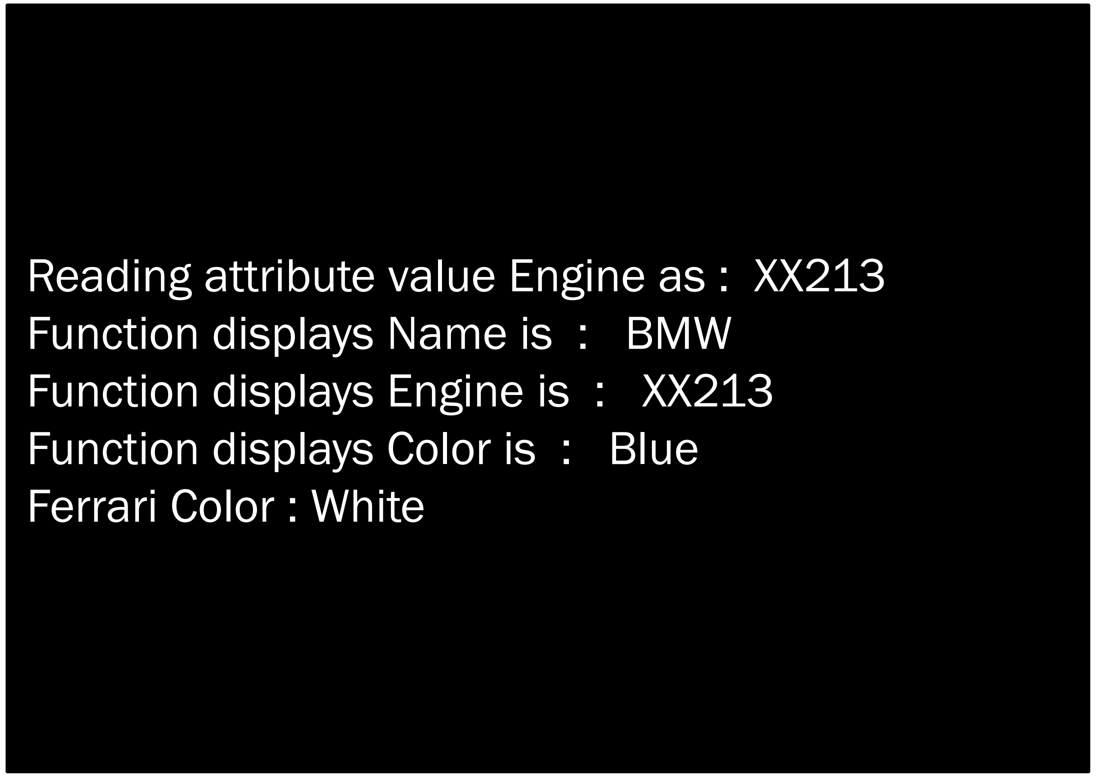

Object
Class
Class Student
name
rollNo
setName()
setRollNo()
Nancy 1999
James 1876
Goldy 1654
Gabby 1659
Blueprint

Checking the code……

```
class Car {
   //field
   Name:string;
   Engine:string;
   Color:string;
   Seats:number;

   //constructor
   constructor(Name:string, Engine:string,  Seats?:number, Col
or:string = "White",) {
     this.Name = Name;
     this.Engine = Engine ;
     this.Color = Color;
     this.Seats = Seats;
   }

   //function
   disp():void {
      console.log("Function displays Name is  :   "+this.Name)

      console.log("Function displays Engine is  :   "+this.Eng
ine)
      console.log("Function displays Color is  :   "+this.Colo
r)
   }
}
```

```
//Create an object
let Car1 = new Car("BMW","XX213", 4, "Blue");
let Car2 = new Car("Ferrari","RX2Z7",2);

//Access the field
console.log("Reading attribute value Engine as
:  "+Car1.Engine)

//Access the function
Car1.disp();
// Car2.disp();
console.log("Ferrari Color : "+Car2.Color);
```

OUTPUT:

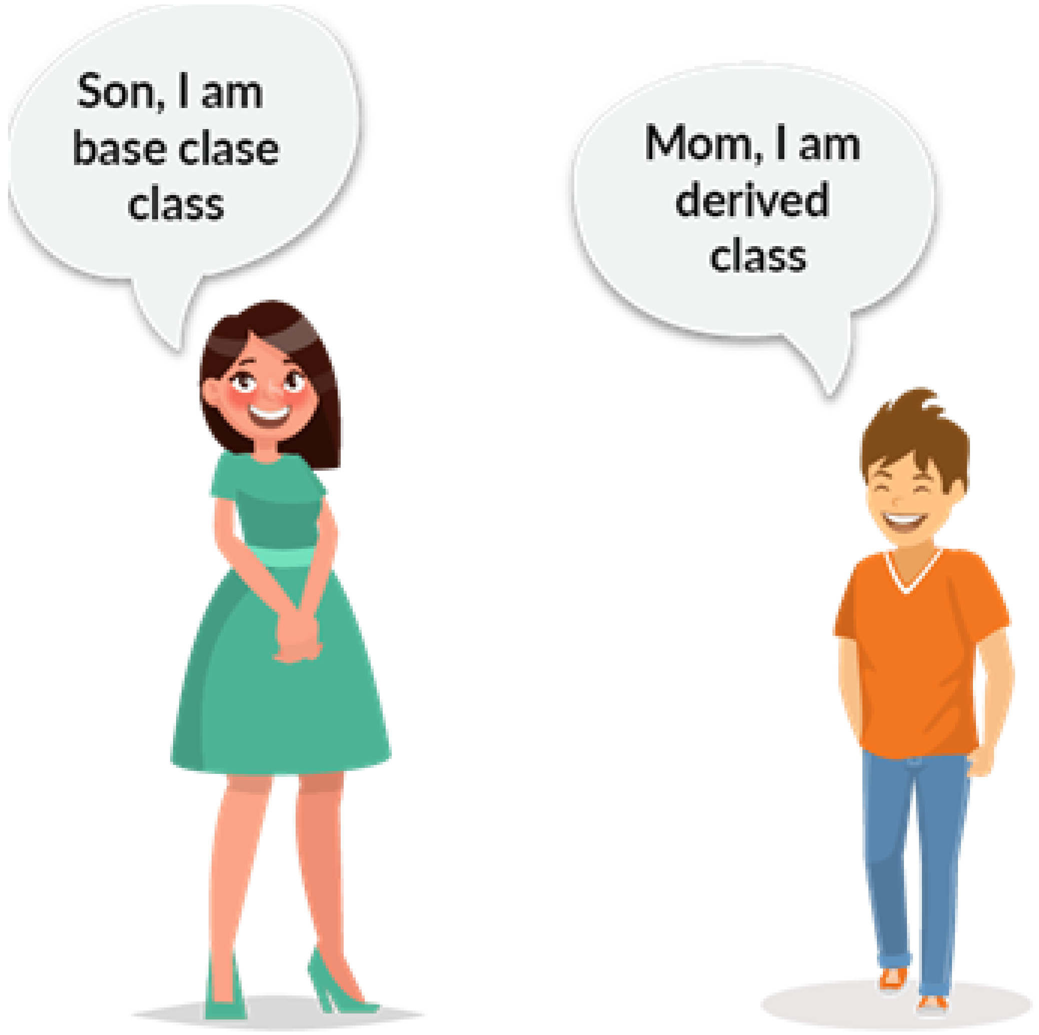


```
Reading attribute value Engine as :  XX213
Function displays Name is  :   BMW
Function displays Engine is  :   XX213
Function displays Color is  :   Blue
Ferrari Color : White
```

Class with Interface…………..

What is inheritance?

- When a class derives from another class.
- The child class will **inherit** all the public and protected properties and methods from the parent class.
- In addition, it can have its own properties and methods.
- An **inherited** class is defined by using the **extends** keyword.



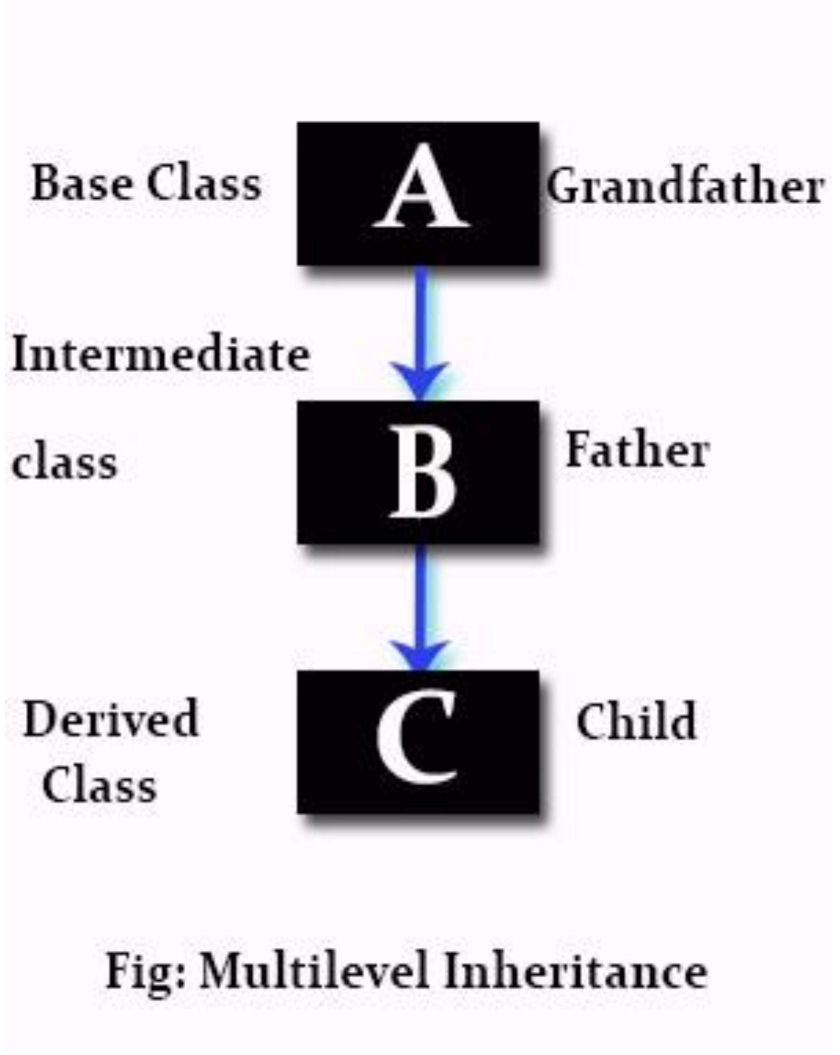


Fig: Multilevel Inheritance

Let’s see a demo for this >>>>>>>>

Exporting and Importing a Class.

To make the code for readable format we should always create different modules of the project.

To do so, we have to remember the keyword like: **import**, **from**, **export**.

For example:

```
Import {class_name} from ‘[relative_path]’;

Example: import {Shape} from './Shape';
```

```
export class {class_name}{//the code};

Example: export class Shape {}
```

Let’s see a demo for this >>>>>>>>

What are access specifiers?

Class can control the visibility of its data members. This is done using access modifiers.

There are three types of access modifiers in TypeScript: public, private and protected.

Public

By default, all members of a class in TypeScript are public. All the public members can be accessed anywhere without any restrictions.

Private

The private access modifier ensures that class members are visible only to that class and are not accessible outside the containing class.

Protected

The protected access modifier is similar to the private access modifier, except that protected members can be accessed using their deriving classes.

Let’s see a demo for this >>>>>>>>

How to work with packages?

- To install Node packages we have Node Package Manager (NPM) which gets pre-installed with NodeJS.
- To install the dependencies we can use the command:

```
npm install [package name] –save
Example: npm install lite-server --save
```

- This will create a “package.json” file which we can modify.

Thank you

Source code: https://github.com/Sujithk007/TypeScript-Workshop