



Table of Content

I. TypeScript

- Introduction to TypeScript
 - What is TypeScript
 - Features of TypeScript
 - Version of TypeScript
- First Example in Typescript
 - Installation of Node.js
 - Installation of TypeScript
 - Installation of VS Code
 - First Program on TypeScript
- TypeScript Basics
 - Variables
 - DataTypes
- TypeScript OOPS
 - Class
 - Object
 - Constructor
 - Inheritance
 - Access Modifiers
 - Interface
 - Enumeration
 - Moudles

II. Angular

- What is Angular?
 - SPA
 - Goals of Angular
 - Versions
 - AngularJS vs Angular

- Fundamental of Angular
 - Building Blocks of Angular
 - Angular Architecture
 - Steps to Prepare First Angular Application.
 - Creating Application Folder
 - Creating @angular/cli package
 - Craeting new angular application
 - Open the angular application in vs code
 - Modify app.component.html
 - Run the application
 - Folder Structure of Angular Application
 - package.json
 - packages of angular
 - tsconfig.json
 - protractor.config.js
 - karma.config.js
 - angular-cli.json
 - src/style.css
 - src/index.html
 - src/main.ts
 - src/app/app.module.ts
 - src/app/app.component.ts
 - src/app/app.component.html
 - src/app/app.component.css
 - src/app/app.component.spec.ts
 - Etc..
 - Components
 - Modules
 - Data Bindings
 - Build in directives
 - ngIf
 - ngIf and else
 - ngSwitch
 - style
 - ngClass
 - ngFor
 - Working with Multiple components
 - Children of Components
 - Life Cycle Hooks
 - Services

- Pipes
- Forms And Validations
 - Template Driven Form
 - Reactive Form
- Routing
- Guards

- RxJS
- Angular Material
- Unit Testing

Course : Angular

No of Days : 20 days

Daily Session : 1 hour

Pre-Requisites

HTML,CSS,JS,TypeScript

DAY-1

Agenda:

1. What is Angular?
 - SPA
 - Goals of Angular
 - Versions
 - AngularJS vs Angular
2. Angular Set Up.
 - Installation of Node.js
 - Installation of VS Code
 - Insallation of Angular

Introduction to Angular:

- ❖ Angular is a client side framework, which is used to create web applications.
- ❖ The framework provides skeleton of the project and specifies clear guidelines, where to write which type of code.
- ❖ Angular can be used in combination with any server side platform such as Java, NodeJS, Asp.Net, PHP, Python etc.
- ❖ Angular is developed using "TypeScript" language, which is a superset of JavaScript language.

- ❖ Angular is the most-used client side framework.
- ❖ Angular was developed by Google.
- ❖ Angular is free-to-use (commercially too).
- ❖ Angular is open-source. That means the source code of angular is available online for free of cost.
- ❖ Angular is cross-platform. That means it works in all the operating systems.
- ❖ Angular is cross-browser compatible. That means it works in all the browsers, except less than IE 9 (which is completely out-dated).
- ❖ Angular is mainly used to create "data bindings". That means, we establish relation between a variable and html element; When the value of the variable is changed, the same will be automatically effected in the corresponding html element; and vice versa.
- ❖ So that the developer need not write any code for DOM manipulations (updating values of html tags, based on user requirements. for example, updating the list of categories when a new category added by the user). Thus the developer can fully concentrate on the application logic, instead of writing huge code for DOM manipulations. So we can achieve clean separation between "application logic" and "DOM manipulations".
- ❖ Angular mainly works based on "Components". The component is a class, which reprensents a specific section (part) of the web page.

Goals of Angular:

- ❖ Make a Single Page Application:
A Single Page Application (SPA) is a web application that loads and renders all necessary content on a single web page. Instead of navigating to different pages and reloading the entire page, SPAs dynamically update the content on the page as users interact with it, providing a more seamless and responsive user experience.
 - It make Development easier.
 - SPA provides client-side navigation system; but can communicate with server only through AJAX; the web page never gets refreshed fully.
 - **Ex:** Gmail, PayPal, Pinterest, Gmail, Facebook
- ❖ Seperation HTML logic from Application Logic.
- ❖ Performing Unit Testing

Versions:

Angular Version	Release Date	Features
AngularJS 1.x	Oct 2010	Initial release of AngularJS, a JavaScript-based framework for building dynamic web applications.
Angular 2	Sep 2016	Complete rewrite of AngularJS. Introduced a component-based architecture, improved performance, and better tooling support.
Angular 4	Mar 2017	Improved performance, smaller bundle sizes, and added new features such as the introduction of the 'else' clause in Angular templates.
Angular 5	Nov 2017	Improved build optimizer, support for progressive web apps (PWA), and introduction of Angular Universal Transfer API.
Angular 6	May 2018	Introduced Angular Elements for building reusable components, Angular Material starter components, and improved Angular CLI commands.
Angular 7	Oct 2018	Improved performance and introduced features

		like Angular CLI prompts, drag-and-drop capabilities, and Virtual Scroll.
Angular 8	May 2019	Introduced Ivy Renderer as an opt-in preview, differential loading for smaller bundles, and improved Angular CLI commands.
Angular 9	Feb 2020	Introduced the Ivy Renderer as the default rendering engine, improved performance, and updated dependencies to their latest versions.
Angular 10	June 2020	Improved performance and introduced stricter type checking with the 'strict' mode enabled by default.
Angular 11	Nov 2020	Improved performance, support for webpack 5, and updated dependencies.
Angular 12	May 2021	Introduced stricter type checking for templates, improved build and testing processes, and updated dependencies.
Angular 13	Nov 2021	Improved performance, enhanced developer experience, and updated dependencies.

Difference between AngularJS and Angular

AngularJS	Angular
AngularJS is a JavaScript-based framework.	Angular is a complete rewrite of AngularJS and is built with TypeScript .
It follows the MVC architecture	It follows the components and directives.
It uses ng-bind in order to bind data from view to model and vice versa.	It uses () and [] to bind the data between view and model.
It uses \$routeProvider.when() for routing configuration	It uses @Route Config{(..)} for routing configuration.
It doesn't use the CLI tool	It uses the CLI tool

Browser Compatability of Angular

S.NO	Browser	Support Version
1.	Google Chrome	Any version
2.	Mozilla Firfox	Any version
3.	MS Internet Explorer	9+

DAY-2

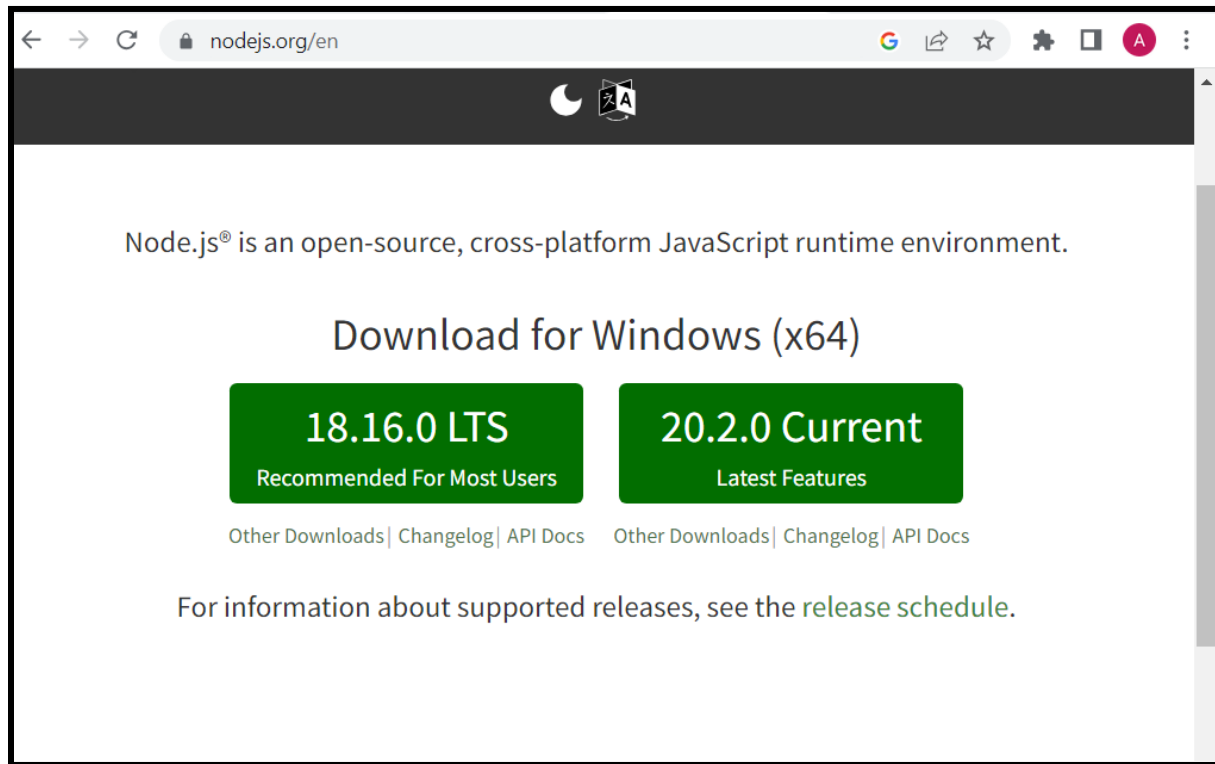
2. Angular Set Up.

Angular

=====

The following are the steps involved to install Angular in the window operating system.

Step1: Install NodeJS in the windows. In order to download we use the following url:
<https://nodejs.org/en>



After download the nodejs the following is the way of installation of nodejs:



Node.js Setup



Welcome to the Node.js Setup Wizard



The Setup Wizard will install Node.js on your computer.


Back

Next

Cancel

Node.js Setup

End-User License Agreement



Please read the following license agreement carefully

Node.js is licensed for use as follows:

Copyright Node.js contributors. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so,

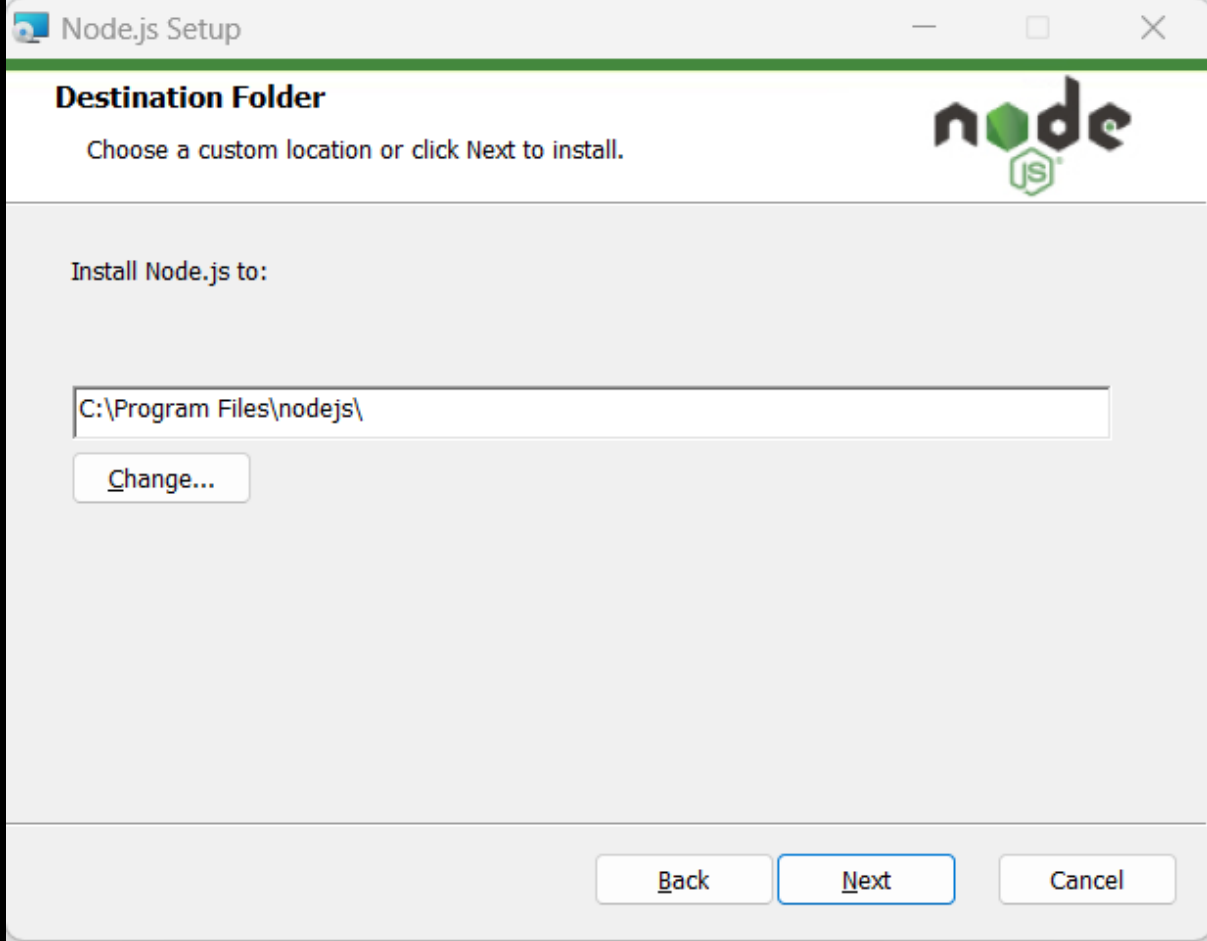
☒ I accept the terms in the License Agreement

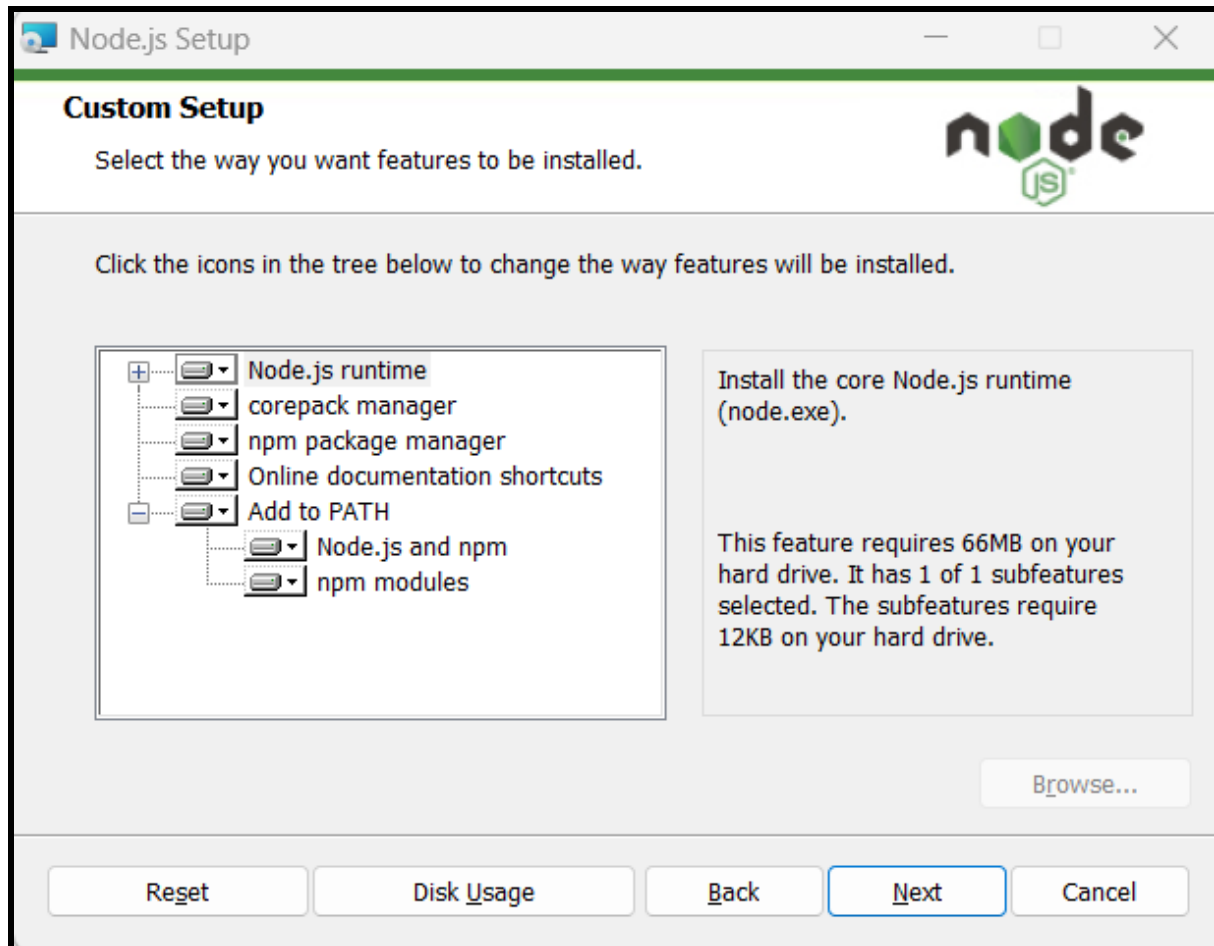
Print

Back

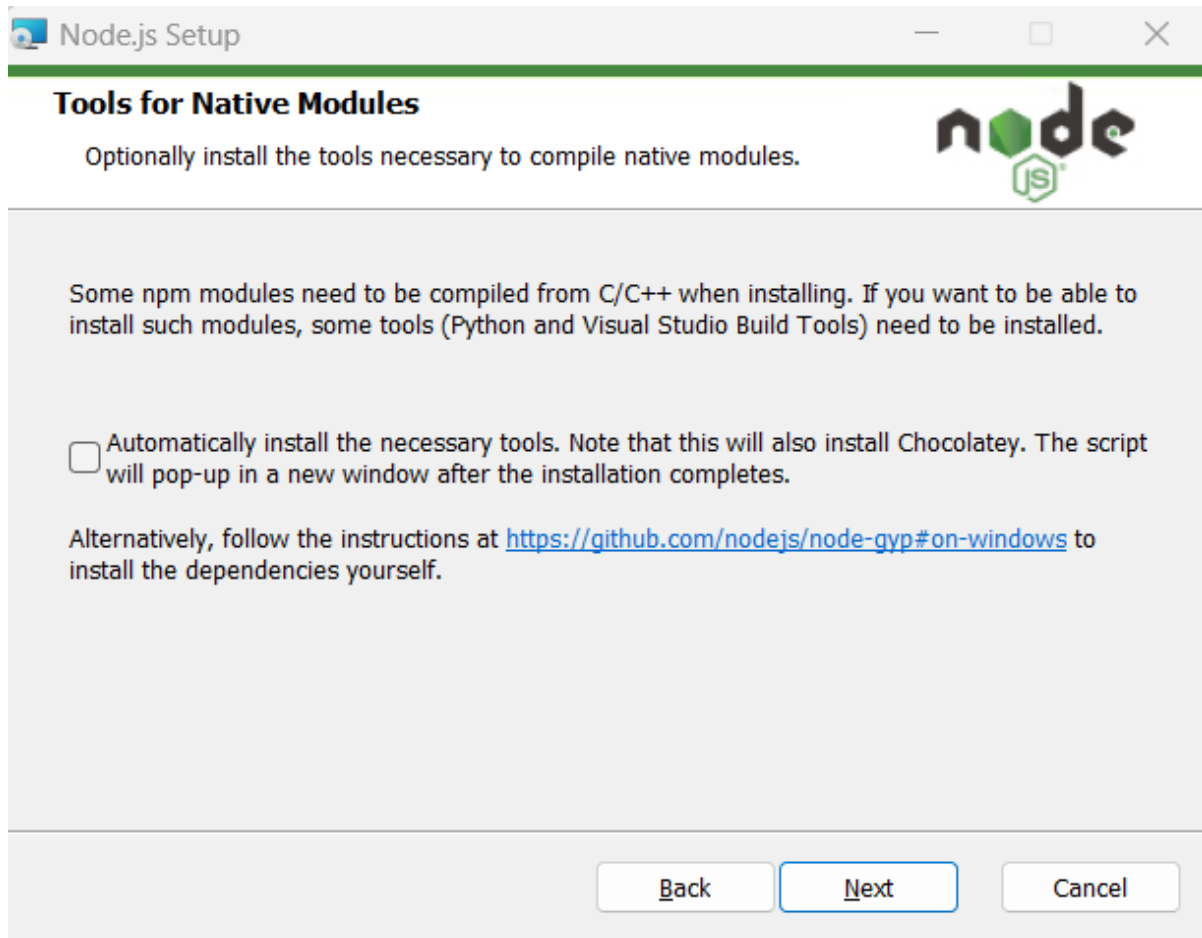
Next

Cancel

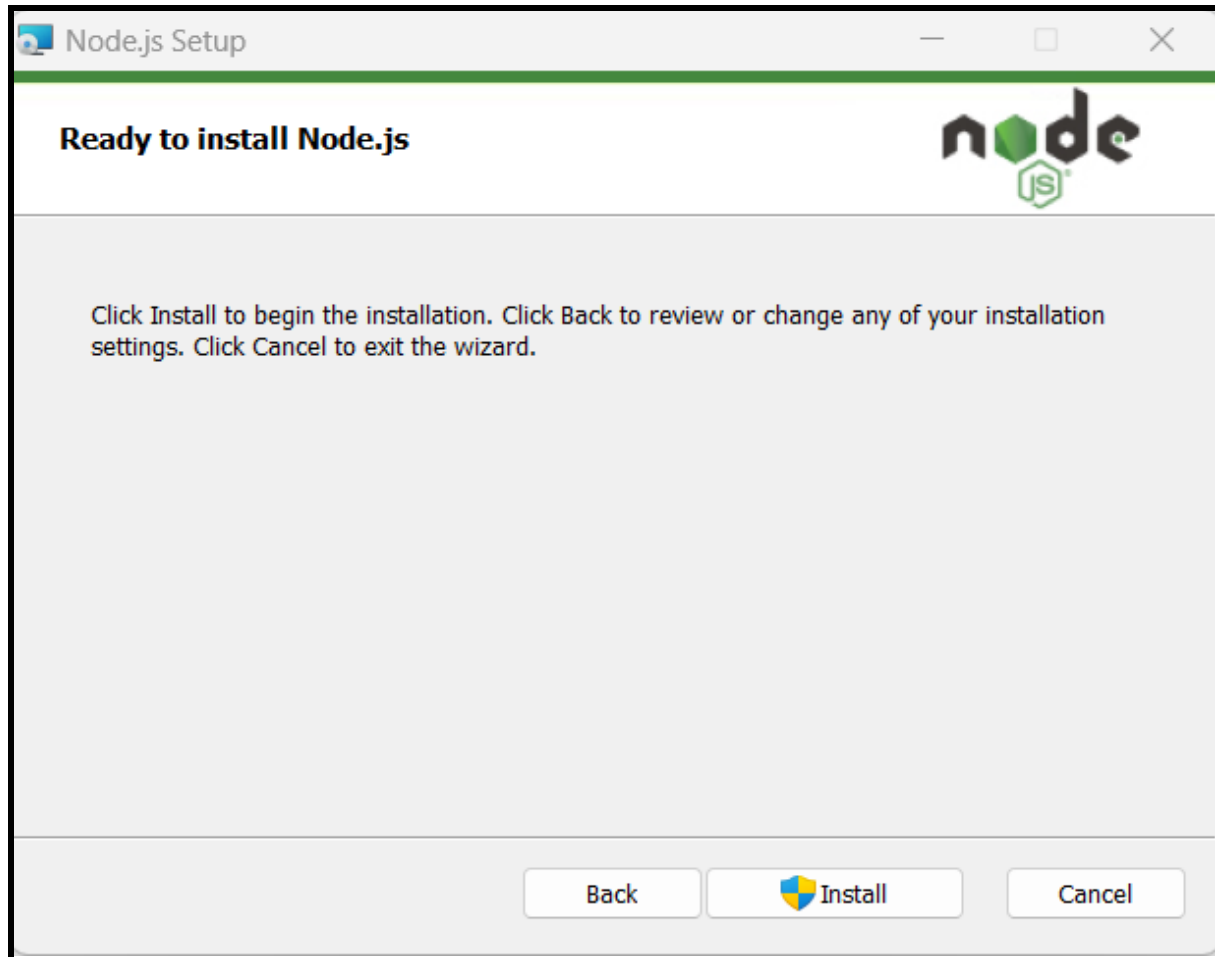




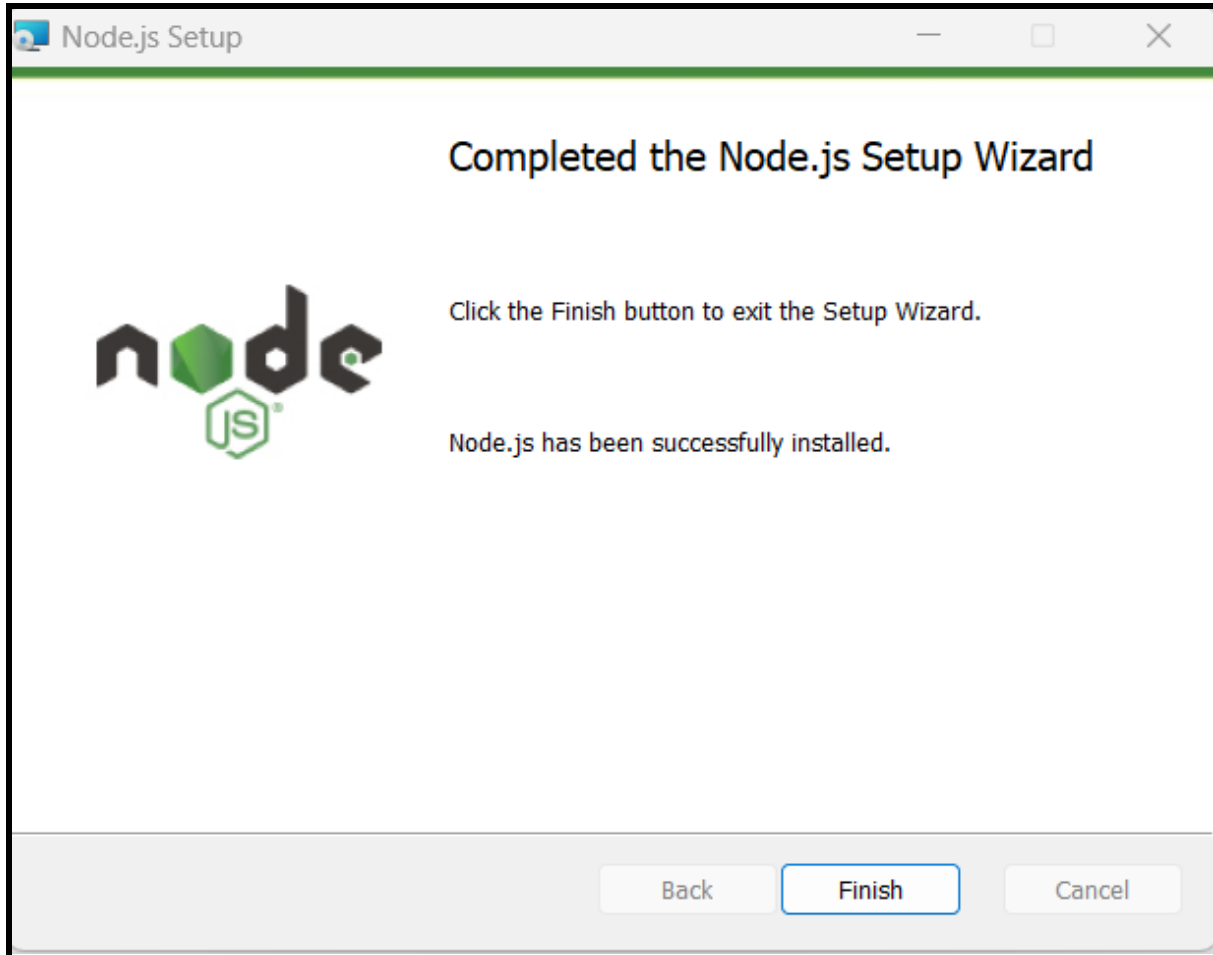
Click on Next button



Click On Next Button



Now Click on Install button



Click on Finish Button

Once we install the nodejs we can cross check the node js by opening the command prompt.

```
C:\WINDOWS\system32\cmd. x + v - [icon] X
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>node --version
v18.16.0

C:\Users\ADMIN>
```


Let's learn about the npm which is by default installed while we install a node js.

What is NPM?

- NPM – or "Node Package Manager" – is the default package manager for JavaScript's runtime Node.js.
- It's also known as "Ninja Pumpkin Mutants", "Nonprofit Pizza Makers", and a host of other random names that you can explore and probably contribute to over at npm-expansions.
- **NPM consists of two main parts:**
 - a CLI (command-line interface) tool for publishing and downloading packages.
 - an online repository that hosts JavaScript packages.
<https://www.npmjs.com/>

Let's learn how to install angular in the window operating system.
The following are the below steps to install the angular:

Step1 : Check the version of nodejs, npm as below command:

```
C:\Users\ADMIN>node --version
v18.16.0

C:\Users\ADMIN>npm --version
9.5.1

C:\Users\ADMIN>
```

Step2: Now install the angular by visiting the official website of angular

<https://angular.io/docs>

Step3: Open the command prompt and install the angular 10

Note : If angular is already there in your system you can uninstall by using following command:

```
C:\Users\ADMIN>npm uninstall -g angular-cli  
  
up to date in 82ms  
  
C:\Users\ADMIN>
```

1. npm cache clean --force
2. npm cache verify

As of now we have not installed angular any version.

Let's Learn how to install

```
D:\angular practise>npm install -g @angular/cli
```

Now check the version of angular by using following command:

```
D:\angular practise>ng version
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. Yes
```

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

```
ng analytics disable --global
```

```
Global setting: enabled
Local setting: No local workspace configuration file.
Effective status: enabled
```

Angular CLI

```
Angular CLI: 16.0.4
Node: 18.16.0
Package Manager: npm 9.5.1
OS: win32 x64
```

```
Angular:
...
```

Package	Version
@angular-devkit/architect	0.1600.4 (cli-only)
@angular-devkit/core	16.0.4 (cli-only)
@angular-devkit/schematics	16.0.4 (cli-only)
@schematics/angular	16.0.4 (cli-only)

Step4: Create the new angular project i.e. myapp by using below command:

```

D:\angular practise>ng new myapp
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE myapp/angular.json (2695 bytes)
CREATE myapp/package.json (1036 bytes)
CREATE myapp/README.md (1059 bytes)
CREATE myapp/tsconfig.json (901 bytes)
CREATE myapp/.editorconfig (274 bytes)
CREATE myapp/.gitignore (548 bytes)
CREATE myapp/tsconfig.app.json (263 bytes)
CREATE myapp/tsconfig.spec.json (273 bytes)
CREATE myapp/.vscode/extensions.json (130 bytes)
CREATE myapp/.vscode/launch.json (470 bytes)
CREATE myapp/.vscode/tasks.json (938 bytes)
CREATE myapp/src/main.ts (214 bytes)
CREATE myapp/src/favicon.ico (948 bytes)
CREATE myapp/src/index.html (291 bytes)
CREATE myapp/src/styles.css (80 bytes)
CREATE myapp/src/app/app.module.ts (314 bytes)
CREATE myapp/src/app/app.component.html (23083 bytes)
CREATE myapp/src/app/app.component.spec.ts (889 bytes)
CREATE myapp/src/app/app.component.ts (209 bytes)
CREATE myapp/src/app/app.component.css (0 bytes)
CREATE myapp/src/assets/.gitkeep (0 bytes)
✓ Packages installed successfully.
warning: in the working copy of '.editorconfig', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.vscode/extensions.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.vscode/launch.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.vscode/tasks.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'angular.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/app/app.component.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/app/app.component.spec.ts', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/app/app.component.ts', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/app/app.module.ts', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/index.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main.ts', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/styles.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'tsconfig.app.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'tsconfig.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'tsconfig.spec.json', LF will be replaced by CRLF the next time Git touches it
Successfully initialized git.

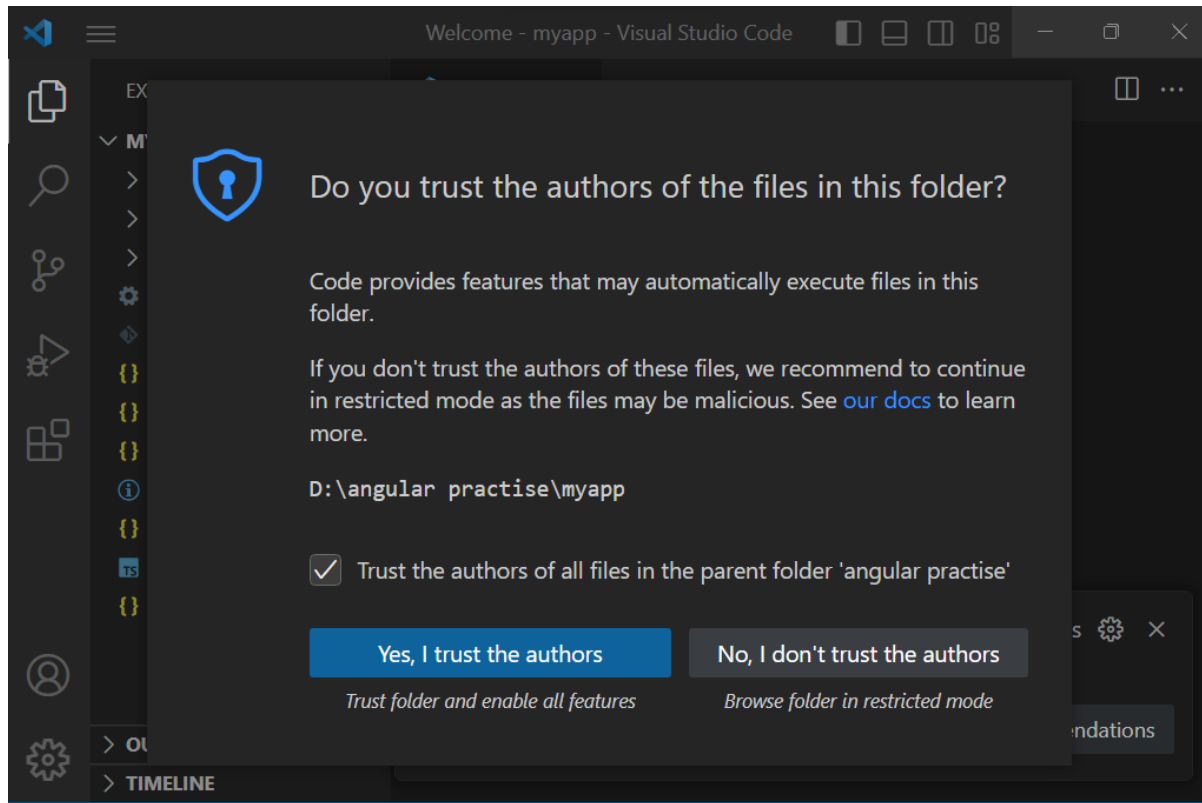
```

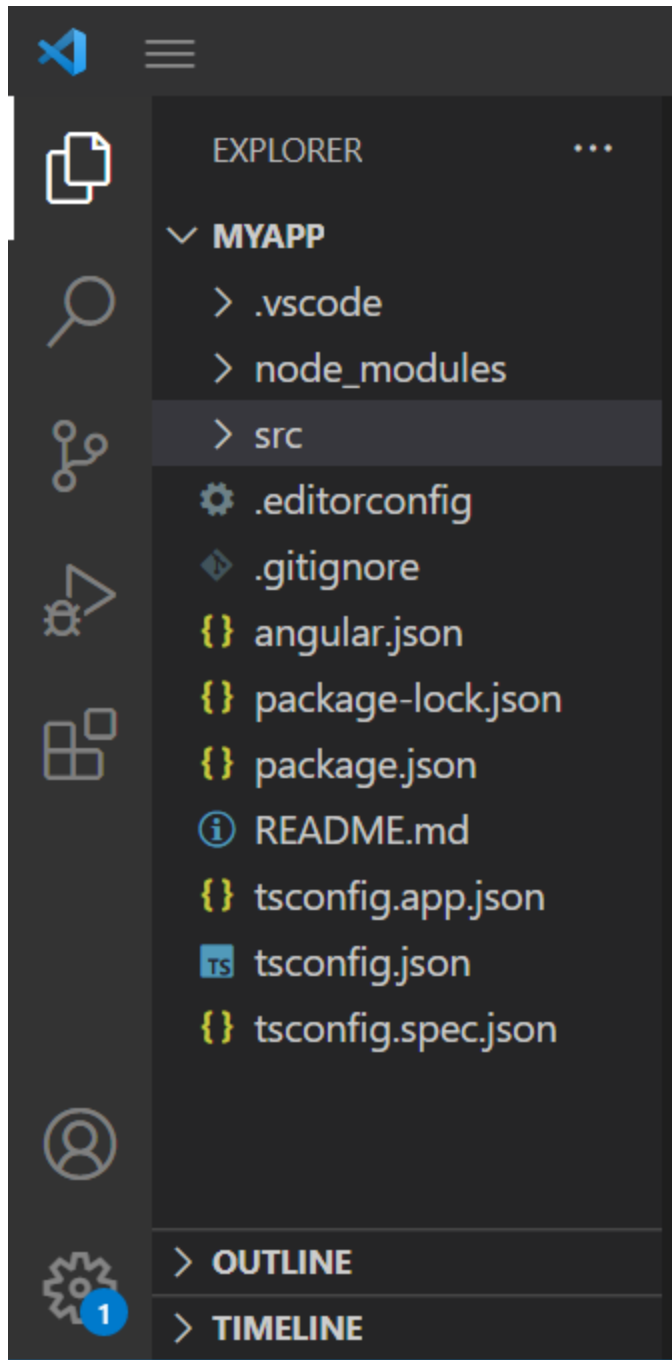
Step5: Now move to the myapp directory and open the project into the visual studio.

```

D:\angular practise\myapp>code .

```

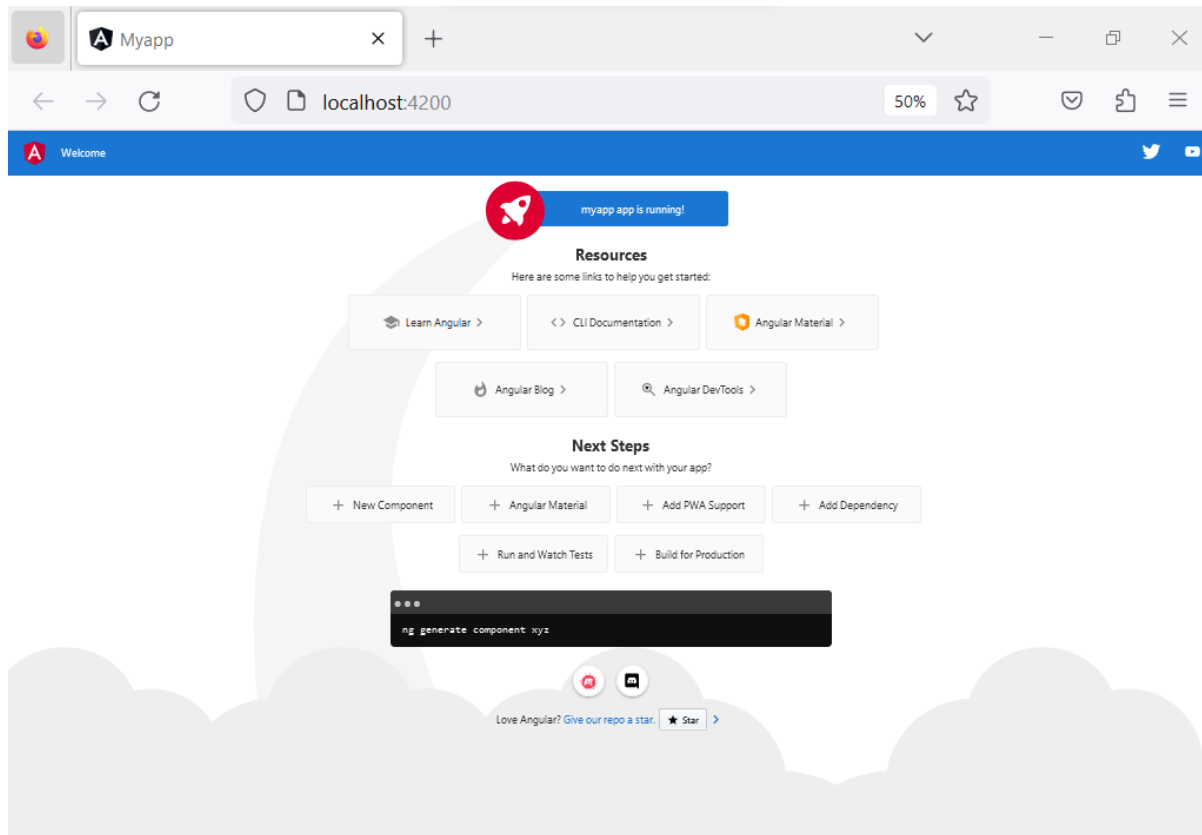




Step6: Open the terminal and start the angular application

```
PS D:\angular practise\myapp> npm start
```

Step7: Open into the browser and test it



DAY-3

Agenda:

- **TypeScript**
 - What is TypeScript?
 - Difference between TypeScript and JavaScript
 - Features of TypeScript
 - Version of TypeScript
 - Steps to Prepare First Example in TypeScript

TypeScript:

- It is programming language, which is developed based on JS.
- It is superset of JS, which adds data types, classes, interfaces and other features.
- It is developed by Microsoft Corporation in 2012.

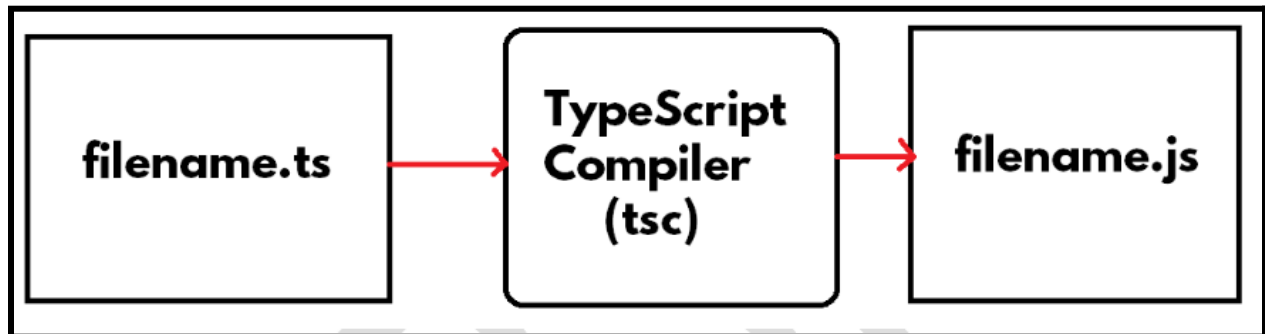
Difference between TypeScript and JavaScript

S.NO	TypeScript	JavaScript
1.	Supports static typing and allows declaring types for variables, function parameters, and return values.	Dynamically typed language where types are determined at runtime.
2.	Requires a compilation step to convert TypeScript code into JavaScript code.	No compilation step required, as JavaScript code runs directly in the browser or on a server.
3.	In Case TS we have features like interfaces, classes, modules, and generics.	It doesn't have build-in support for interfaces, classes, modules.
4.	The extension is .ts	The extension is .js
5.	It supports object-oriented programming concepts like classes, interfaces, inheritance, generics etc..	It just a scripting language.

Features of TypeScript

- **Static Typing :**
 - TypeScript allows you to explicitly declare the types of variables, function parameters, and return values. It helps catch errors during development by ensuring that values are used correctly and consistently.
- **Modules :**
 - TypeScript provides a module system that helps organize code into reusable and independent units. Modules allow you to encapsulate related functionality and control the visibility of variables and functions, making it easier to manage large codebases.
- **Interfaces :**
 - TypeScript introduces interfaces, which define the structure and shape of objects. They enable you to specify the expected properties and methods that an object should have, promoting code clarity and maintainability.
- **Classes:**
 - TypeScript supports object-oriented programming concepts like classes, inheritance, and access modifiers (e.g., public, private, protected). Classes allow you to define blueprints for creating objects with shared properties and behaviors.
- **Generics :**
 - TypeScript supports generics, allowing you to create reusable components that can work with different types. Generics provide a way to write flexible and type-safe code by abstracting over specific data types.
- **Tooling and IDE support :**

- TypeScript offers excellent tooling and support in popular IDEs like Visual Studio Code. This includes features like autocompletion, type checking, and refactoring tools, which enhance developer productivity and help identify errors early.
- **Browser Compatability :**
 - TypeScript code can be transpiled into JavaScript that is compatible with older browsers. You can write modern JavaScript syntax and features and then convert it into a version that works in a wide range of environments.



Version History

Version	Year
0.8	2012 (first version)
0.9	2013
1.0	2014
2.0	2016
2.6	2017
2.8	2018
...	..
5.0	2023
https://en.wikipedia.org/wiki/TypeScript	

Steps to Prepare First Example in TypeScript

Step1: Install the Node.JS

- <https://nodejs.org/en>

Step2: Install the TypeScript

- <https://www.typescriptlang.org/download>

Step3: Install VS Code

Step4 : Create a folder structure

Step5 : Create the TypeScript file and write the code

Step6 : Compile the TypeScript Program

Step7: Execute the TypeScript Program

- In order to install the typescript via npm we can use the below command:
 - `npm install typescript -g`

```
C:\Users\MOHAMMED IMTIAZ>npm install typescript -g
```

```
C:\Users\MOHAMMED IMTIAZ>tsc -version  
Version 5.1.3
```

DAY-4

Agenda

- TypeScript Basics
 - Variables
 - DataTypes
- TypeScript OOPS
 - Class
 - Object
 - Constructor
 - Inheritance
 - Access Modifiers
 - Interface
 - Enumeration
 - Moudles

TypeScript Basics

Variables

- Variable is a named memory location in RAM, to store a value at run time.
 - **Syntax:** var variable : datatype = value;
 - **Example:** var a : number = 100;
- TypeScript supports “optional static typing”. That means it is optional to specify datatype
- for the variable in TypeScript.
 - **Static Typing:**
 - If you specify the data type for the variable, it is not possible assign other data type of value into the variable; if you do so, “tsc” compiler will generate coding-time and compile-time errors; but the code will be compiled and executed also, even though it is having errors.

- **Dynamic Typing:** If you don't specify the data type for the variable, we can assign any type of value into the variable.
- In TypeScript, we have "optional static typing". That means it is optional to specify datatype for the variable in TypeScript

Data Types

- **The following are the list of DataTypes in TypeScript:**
 - number : All types of numbers(integer, floating)
 - string : Collection of characters in double quotes or single quotes
 - boolean : true or false
 - any : Any type of value

Object

- Object is the primary concept in OOP (Object Oriented Programming).
- "Object" represents a physical item, that represents a person or a thing.
- Object is a collection of properties (details) and methods (manipulations).
- For example, a student is an "object".
- We can create any no. of objects inside the program.

Property

- Properties are the details about the object.
- Properties are used to store a value.
- For example, **studentname="Scott"** is a property in the "student object".
- Properties are stored inside the object.
- The value of property can be changed any no. of times

Method

- Methods are the operations / tasks of the object, which manipulates / calculates the data and do some process.
- Method is a function in the object.
- Methods are stored inside the object.

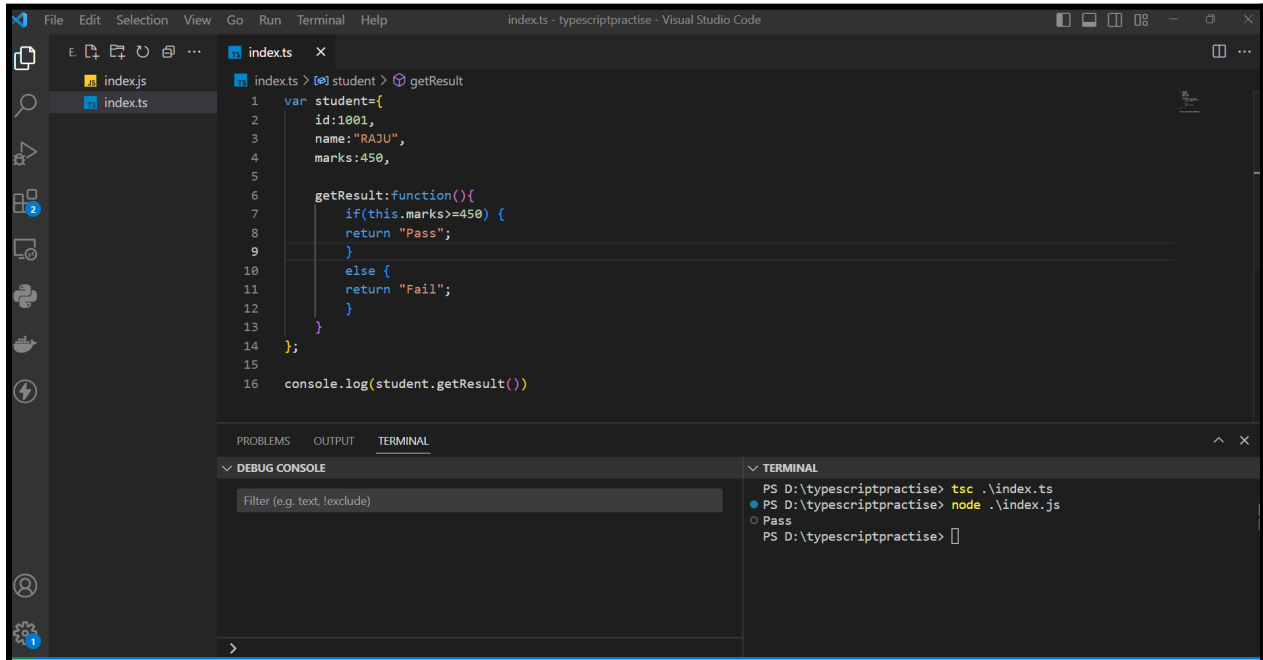
Creating Object in TypeScript

- Object can be created by using "Object Literal Pattern" in TypeScript.
- Object literal" is a collection of properties and methods, that are enclosed within curly braces { }.
- **Syntax** to create Object: { property: value, ..., method: function() { code here } }

Reference Variable

- The "reference variable" is a variable, that can store the reference of an object.
- We can access all the properties and methods of the object, by using the "reference variable".
- Syntax to create Object and store its reference in the "reference variable":
- `var referenceVariable = { property: value, ..., method: function() { code here } };`

Example :



The screenshot shows the Visual Studio Code editor with a file named `index.ts` open. The code defines a `student` object with properties `id`, `name`, and `marks`, and a `getResult` method that returns "Pass" if marks are 450 or more, and "Fail" otherwise. The code is as follows:

```
1 var student={
2   id:1001,
3   name:"RAJU",
4   marks:450,
5
6   getResult:function(){
7     if(this.marks>=450) {
8       return "Pass";
9     }
10    else {
11      return "Fail";
12    }
13  }
14 };
15
16 console.log(student.getResult())
```

The bottom panel shows the `TERMINAL` with the following commands and output:

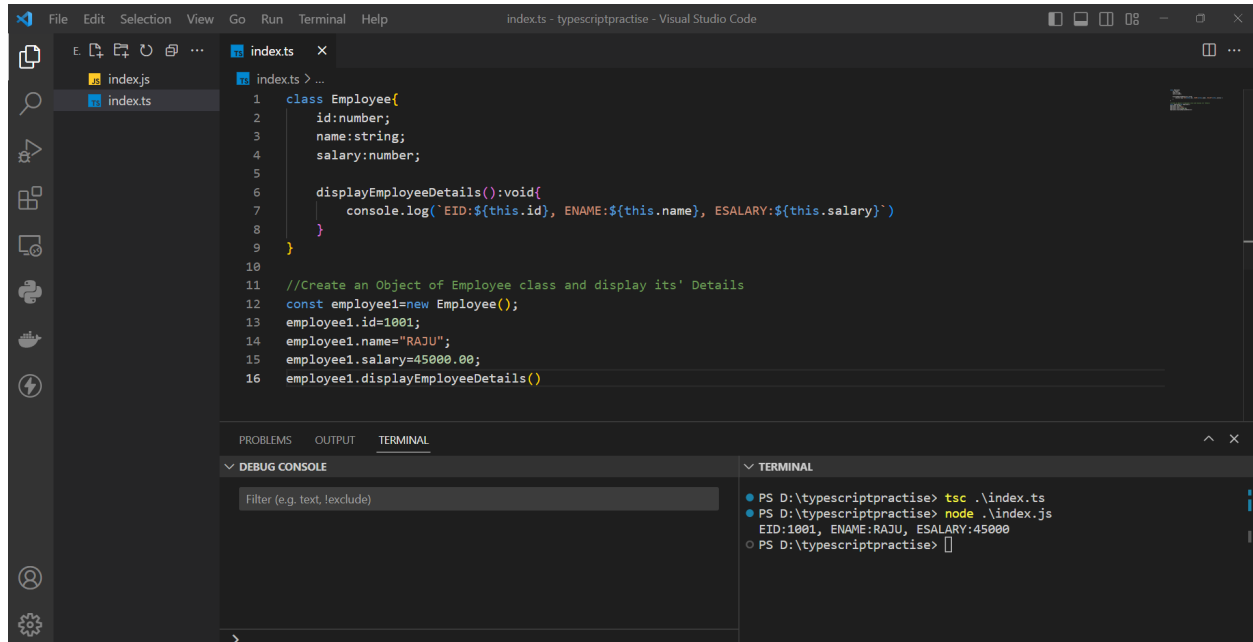
```
PS D:\typescriptpractise> tsc .\index.ts
PS D:\typescriptpractise> node .\index.js
Pass
PS D:\typescriptpractise>
```

Class

- It is represent a model of the object, which defines list of properties and method of the object.
- Example : “ Employee “ class represents structures(list of properties and methods) of every Employee object.
- We can create any no of objects based on a class by using a “new” keyword.
- All the objects of a class, shares same set of properties and method of the class.
- We can store the reference of the object in “reference variable”; using which you can access the object.
- Syntax:

```
class className{
  //properties
  //methods
}
```


Example:



```
1 class Employee{
2   id:number;
3   name:string;
4   salary:number;
5
6   displayEmployeeDetails():void{
7     console.log(`EID:${this.id}, ENAME:${this.name}, ESALARY:${this.salary}`)
8   }
9 }
10
11 //Create an Object of Employee class and display its' Details
12 const employee1=new Employee();
13 employee1.id=1001;
14 employee1.name="RAJU";
15 employee1.salary=45000.00;
16 employee1.displayEmployeeDetails()
```

PROBLEMS OUTPUT TERMINAL

DEBBUG CONSOLE

Filter (e.g. text, exclude)

TERMINAL

```
PS D:\typescriptpractise> tsc .\index.ts
PS D:\typescriptpractise> node .\index.js
EID:1001, ENAME:RAJU, ESALARY:45000
PS D:\typescriptpractise>
```

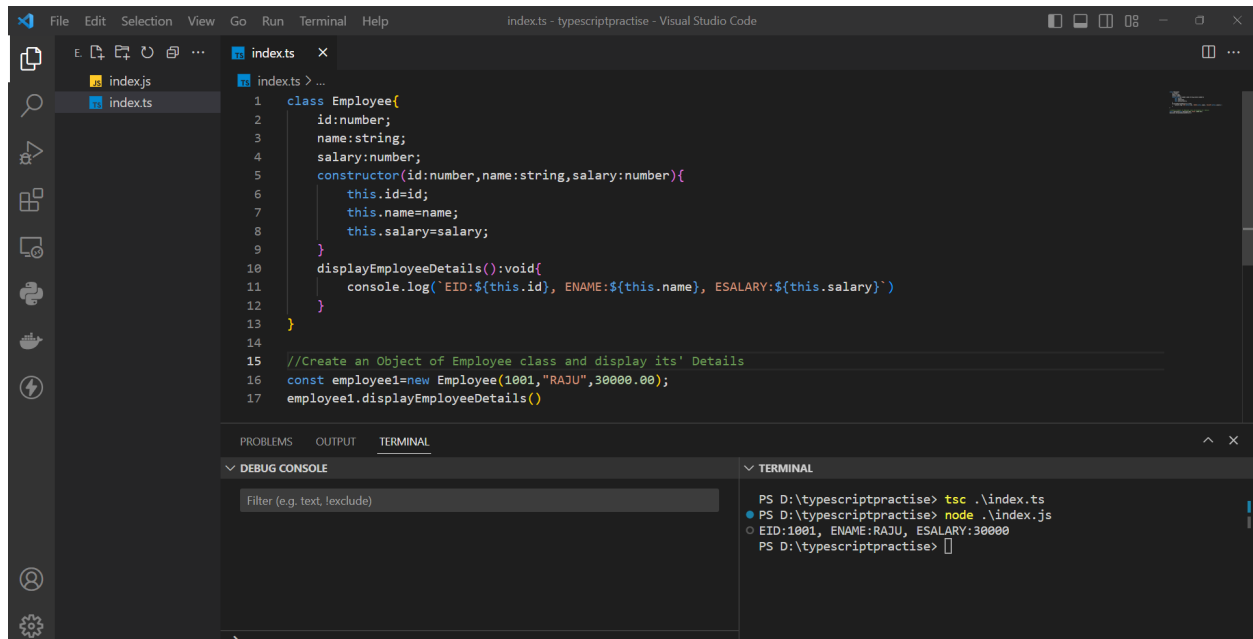
Constructor

- It is a special function which is a part of the class.
 - It will invoked automatically when we create an instance of the class.
 - Note : When we create the multiple instance of the particular class every time it will invoke the constructor.
 - It is basically used to initialize the properties of the class.
 - We use the name as “constructor”.
 - Constructor can take any no of arguments but not the return type.
- Note : We can't define multiple constructor in TypeScript.

Syntax:

```
constructor(arg1:data Type,arg2:data Type,arg3:data Typ,...){
}
}
```

Example:



```
1 class Employee{
2   id:number;
3   name:string;
4   salary:number;
5   constructor(id:number,name:string,salary:number){
6     this.id=id;
7     this.name=name;
8     this.salary=salary;
9   }
10  displayEmployeeDetails():void{
11    console.log(`EID:${this.id}, ENAME:${this.name}, ESALARY:${this.salary}`)
12  }
13 }
14
15 //Create an Object of Employee class and display its' Details
16 const employee1=new Employee(1001,"RAJU",30000.00);
17 employee1.displayEmployeeDetails()
```

PROBLEMS OUTPUT TERMINAL

DEBUG CONSOLE

Filter (e.g. text, exclude)

TERMINAL

```
PS D:\typescriptpractise> tsc .\index.ts
PS D:\typescriptpractise> node .\index.js
EID:1001, ENAME:RAJU, ESALARY:30000
PS D:\typescriptpractise>
```

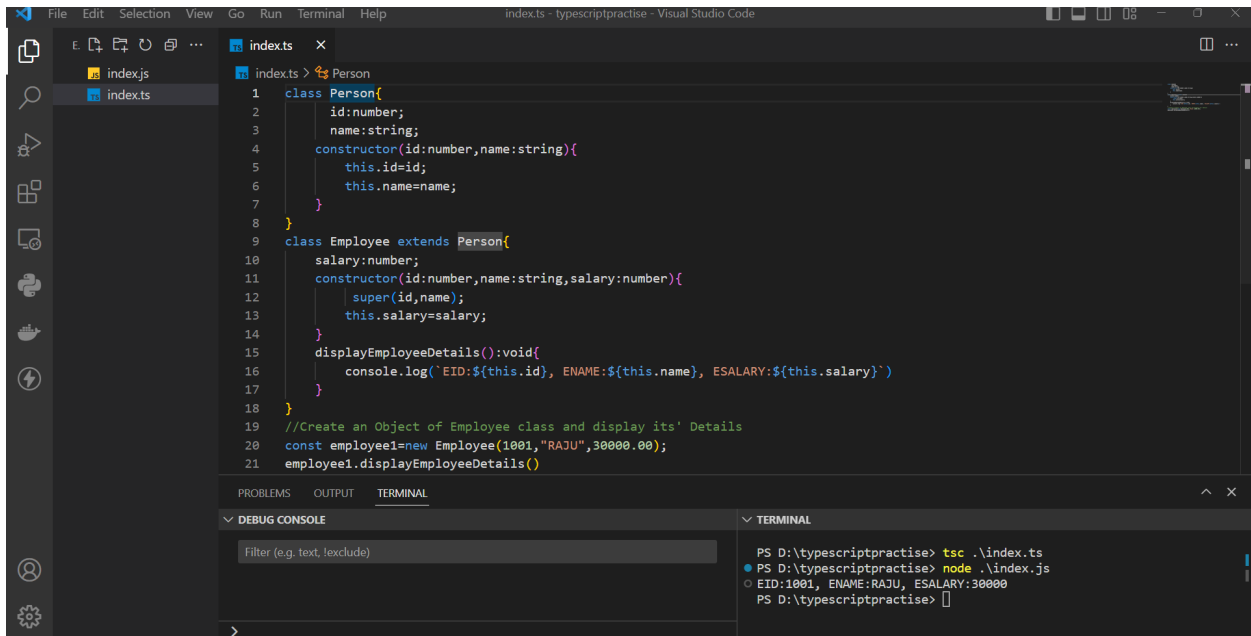
Inheritance

- Acquiring the properties from parent class to child class is called as Inheritance.
- The main advantage of inheritance is code reusability.

Key Points:

- ☐ The 'extends' keyword is used to create inheritance.
- ☐ The 'super' keyword is basically used to access the members of parent class from child class.
- ☐ When Parent class has a constructor, then child class has to invoke explicitly by using super(..)
- Example : Employee extends Person , Batsmen extends Player, ICICIBank extends Bank etc...
- Syntax :
 - class parentClass{
 //parent class members
}
 - class childClass extends parentClass{
 // child class members
}

Example:



```
index.ts | index.js | index.ts
class Person {
  id: number;
  name: string;
  constructor(id: number, name: string) {
    this.id = id;
    this.name = name;
  }
}
class Employee extends Person {
  salary: number;
  constructor(id: number, name: string, salary: number) {
    super(id, name);
    this.salary = salary;
  }
  displayEmployeeDetails(): void {
    console.log(`EID:${this.id}, ENAME:${this.name}, ESALARY:${this.salary}`);
  }
}
// Create an Object of Employee class and display its Details
const employee1 = new Employee(1001, "RAJU", 30000.00);
employee1.displayEmployeeDetails();
```

PROBLEMS OUTPUT TERMINAL

DEBUG CONSOLE

Filter (e.g. text, exclude)

TERMINAL

```
PS D:\typescriptpractise> tsc .\index.ts
PS D:\typescriptpractise> node .\index.js
EID:1001, ENAME:RAJU, ESALARY:30000
PS D:\typescriptpractise>
```

Access Modifiers

- It is used to specify where the member of a class can be accessible. That means it specifies whether the member of a class is accessible outside the class or not.
- Access Modifiers are used to implement "security" in OOP.
- Each member (property / method), we can specify the access modifier separately.
- TypeScript compiler and Visual Studio Code Editor shows errors, if a developer try to access the member, which is not accessible.
- "public" is the access modifier for all the members (property / method) in TypeScript class.

TypeScript supports three access modifiers:

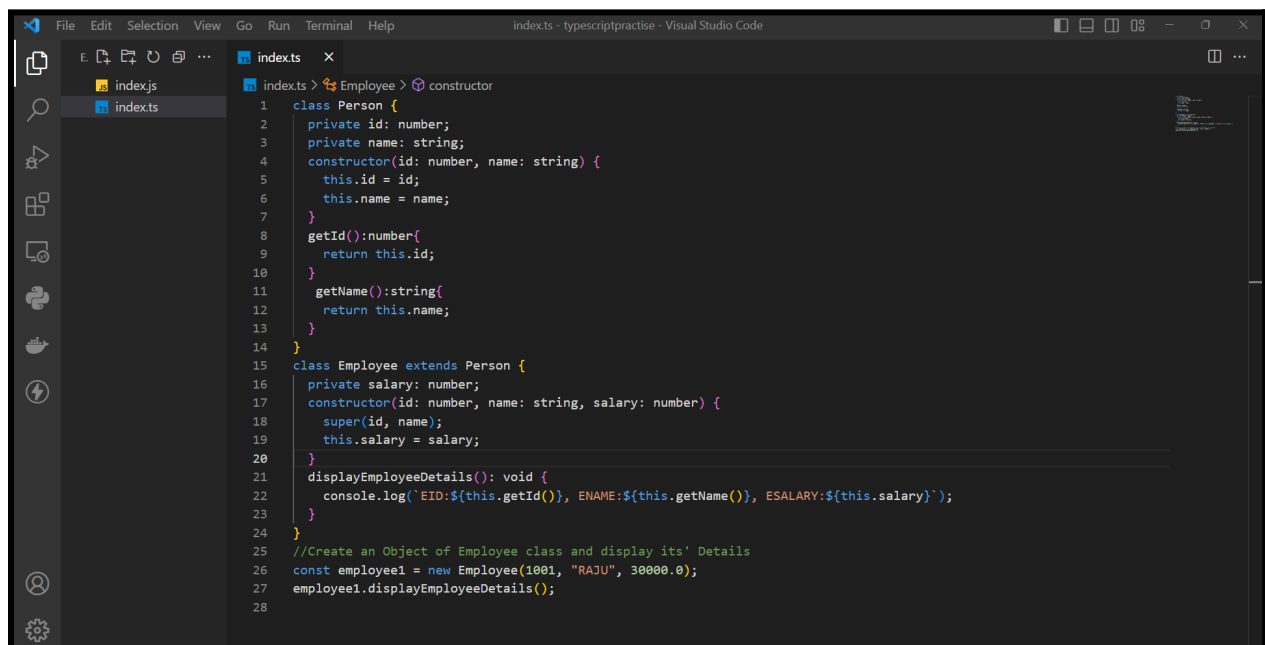
1. public
2. private
3. protected

1. **public:** Public members can be accessed from anywhere, both within the class and outside the class.
2. **private:** Private members can only be accessed within the class in which they are defined. They are not accessible from outside the class or from derived classes.
3. **protected:** Protected members can be accessed within the class and in derived classes. They are not accessible from outside the class.

Syntax:

```
class className{  
    accessModifier property:dataType;  
  
    accessModifier methodName(lis_of_args):returnType{  
  
    }  
  
}
```

Example:



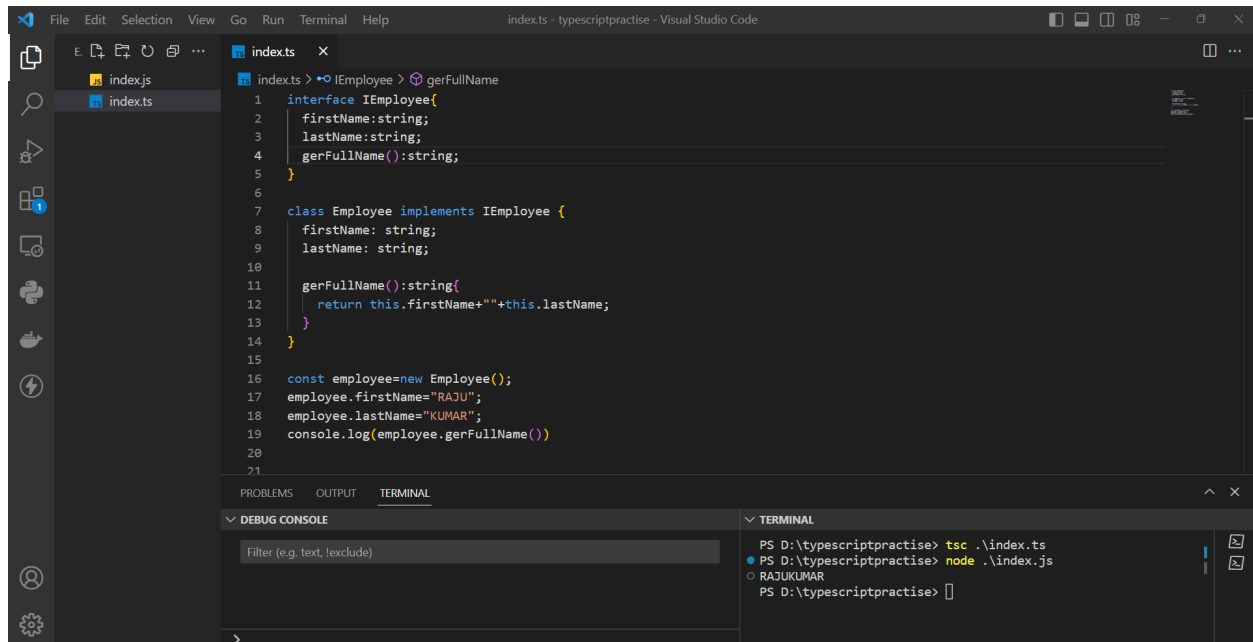
Interfaces

- Interface is the model of a class, which describes the list of properties and methods of a class.
- Interfaces doesn't contain actual code; contains only list of properties and methods.
- Interfaces doesn't contain method implementation (method definition); it contains method declaration only, which defines method access modifier, method name, method arguments and method return type.
- The child class that implements the interface must implement all the methods of the interface; if not, compile-time error will be shown at child class. The method name, parameters, return type and access modifier must be same in between "interface method (method at the interface)" and "class method (method at the class)".
- Interfaces act as a mediator between two or more developers; one developer implements the interface, other developer creates reference variable for the interface and invokes methods; so interface is common among them.
- The child class can implement the interface with "implements" keyword.
- All the methods of interface is by default, "public".
- One interface can be implemented by multiple classes; One class can implement multiple interfaces.
- We can develop "multiple inheritance" by implementing multiple interfaces at-a-time in the same class.

Syntax:

```
interface interfaceName{  
    property:dataType;  
    method(list_of_arg):returnType;  
}
```

Example :



```
1 interface IEmployee {
2   firstName: string;
3   lastName: string;
4   getFullName(): string;
5 }
6
7 class Employee implements IEmployee {
8   firstName: string;
9   lastName: string;
10
11   getFullName(): string {
12     return this.firstName + " " + this.lastName;
13   }
14 }
15
16 const employee = new Employee();
17 employee.firstName = "RAJU";
18 employee.lastName = "KUMAR";
19 console.log(employee.getFullName());
20
21
```

DEBUG CONSOLE

Filter (e.g. text, exclude)

TERMINAL

```
PS D:\typescriptpractise> tsc .\index.ts
PS D:\typescriptpractise> node .\index.js
RAJUKUMAR
PS D:\typescriptpractise>
```

Enumerations:

- Enumeration is a collection of constants.
- Enumeration acts as a data type.
- We can use "enumeration" as a data type for the "enumeration variable" or "enumeration property".
- The enumeration variable or enumeration property can store any one of the constants of the same enumeration.
- Every constant of enumeration is represented with a number (starts from 0)

#Steps for creating enumeration

Step1: create the enumeration

```
enum enumerationName{  
    constant1,constant2,...;  
}
```

Step2: Creating the property of enumeration type

```
class className{  
    property:enumerationName;  
}
```

Step3: Create a variable of enumeration type

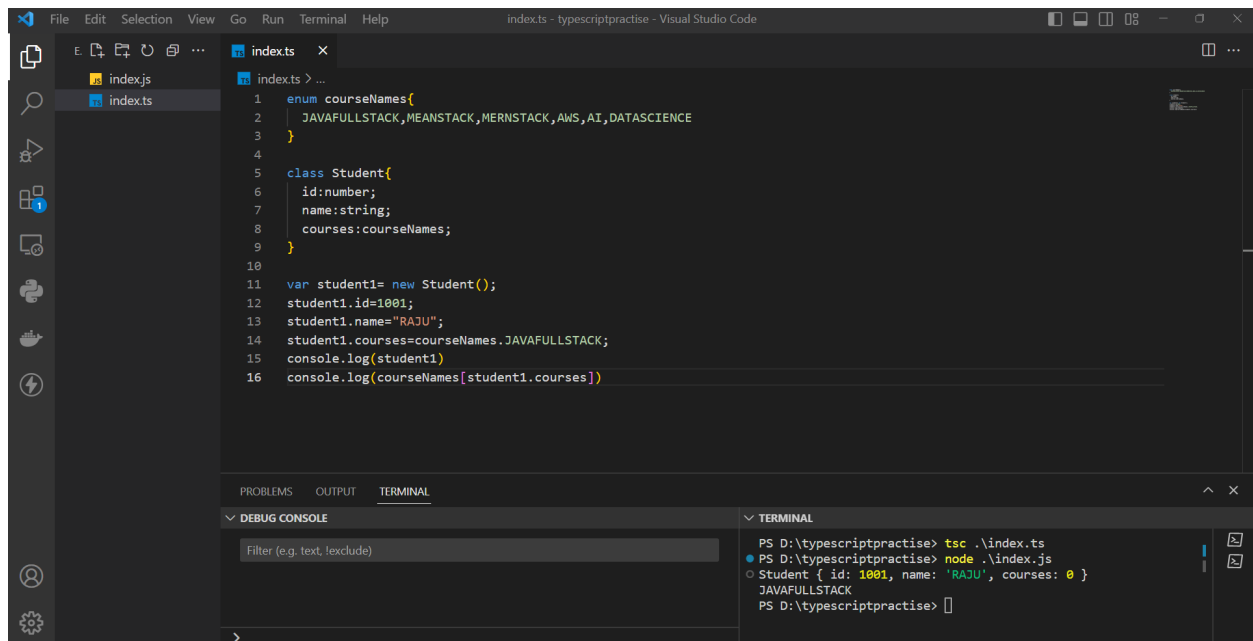
```
variableName : enumerationName;
```

Step4: Assign the value into enumeration variable or enumeration property

```
enumerationVariable = enumerationName.constant;
```

```
this.enumerationProperty=enumerationName.constant;
```

Example:



The screenshot shows a Visual Studio Code editor with a TypeScript file named `index.ts`. The code defines an enumeration `courseNames` with values `JAVAFULLSTACK`, `MEANSTACK`, `MERNSTACK`, `AWS`, `AI`, and `DATASCIENCE`. It also defines a `Student` class with properties `id` (number), `name` (string), and `courses` (of type `courseNames`). A `Student` object is created with `id=1001`, `name='RAJU'`, and `courses=courseNames.JAVAFULLSTACK`. The code logs the student object and the corresponding course name from the enumeration.

```
1  enum courseNames{  
2      JAVAFULLSTACK,MEANSTACK,MERNSTACK,AWS,AI,DATASCIENCE  
3  }  
4  
5  class Student{  
6      id:number;  
7      name:string;  
8      courses:courseNames;  
9  }  
10  
11  var student1= new Student();  
12  student1.id=1001;  
13  student1.name="RAJU";  
14  student1.courses=courseNames.JAVAFULLSTACK;  
15  console.log(student1)  
16  console.log(courseNames[student1.courses])
```

The terminal output shows the command `tsc .\index.ts` being executed, followed by `node .\index.js`. The output of the program is a JSON object representing the student: `{ id: 1001, name: 'RAJU', courses: 0 }`, where `0` corresponds to `JAVAFULLSTACK` in the enumeration.

```
PS D:\typescriptpractise> tsc .\index.ts  
PS D:\typescriptpractise> node .\index.js  
Student { id: 1001, name: 'RAJU', courses: 0 }  
JAVAFULLSTACK  
PS D:\typescriptpractise>
```

DAY-4

Agenda

- Modules
- NodeJS
 - What is NodeJS
 - Working with NodeJS Modules
 - FS
 - HTTP
 - Working with Express JS
- Understanding of Angular Project Structure

Modules:

- In large scale applications, it is recommended to write each class in a separate file.
- To access the class of one file in other file, we need the concept of "Modules".
- Module is a file (typescript file), which can export one or more classes (or interfaces,
- enumerations etc.) to other files; The other files can import classes (or interfaces,
- enumerations etc.), that are exported by the specific file. We can't import the classes (or
- others) that are not exported.
- We can export the class (or others), by using "export" keyword in the source file.
- We can import the class (or others), by using "import" keyword in the destination file.
- To represent:
 - current folder, we use "./".
 - sub folder in current folder, we use "../subfolder".
 - parent folder, we use "../".

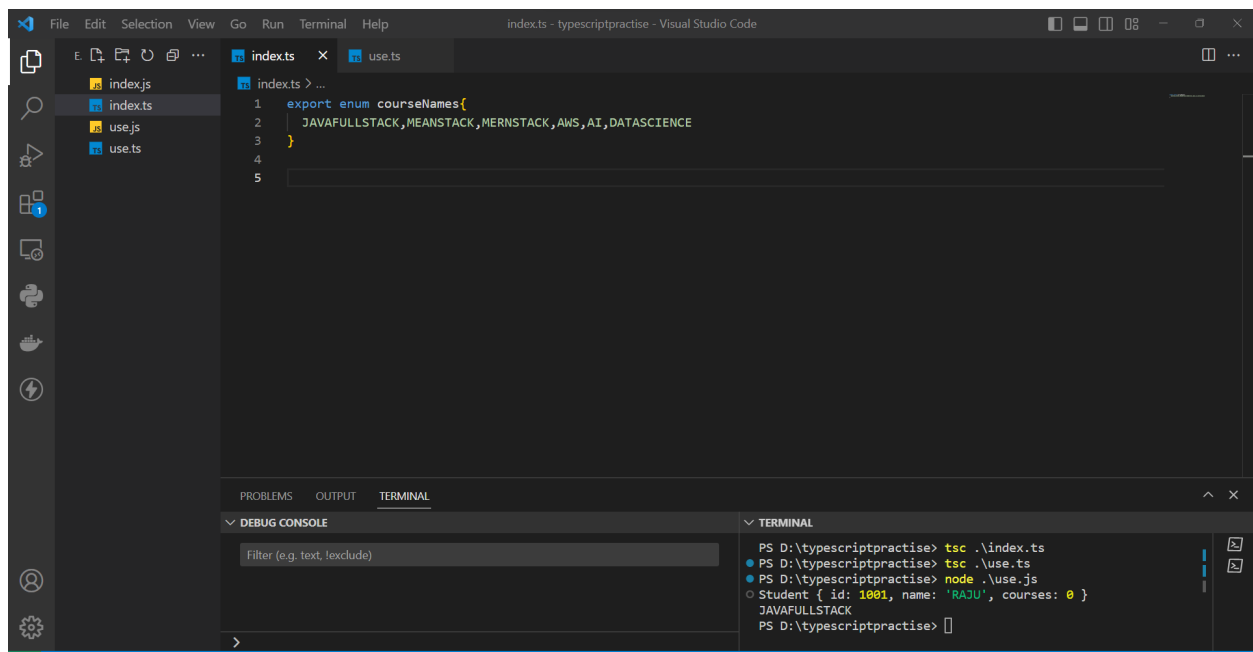
Steps for development of modules

Step1: export a class file1.ts

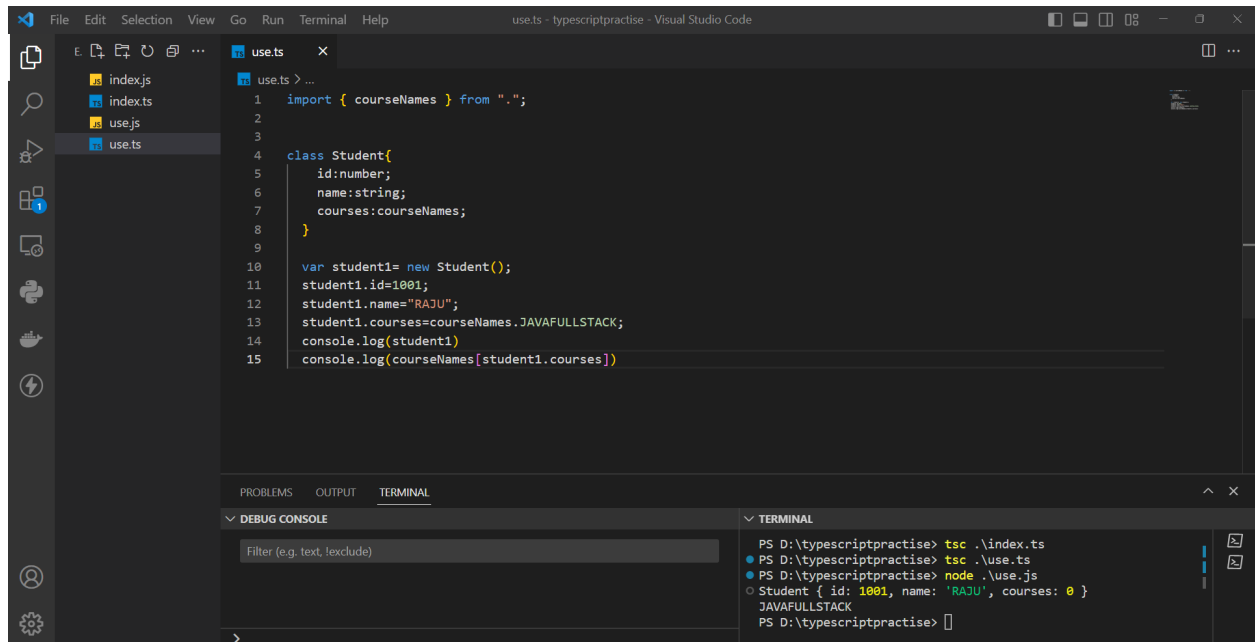
```
export class className{  
...  
}  
import {className} from './file1.ts';  
class className{  
...  
}
```

Example

Step1 : index.ts



Step2: use.ts



Node.JS

- NodeJS is a cross platform runtime environment and library for running JS application outside the browser.
- NodeJS is not a framework.
- Basically it is used for creating server-side and networking web applications.
- Many of the basic modules of Node.js is written in JS.

Working with Node.js

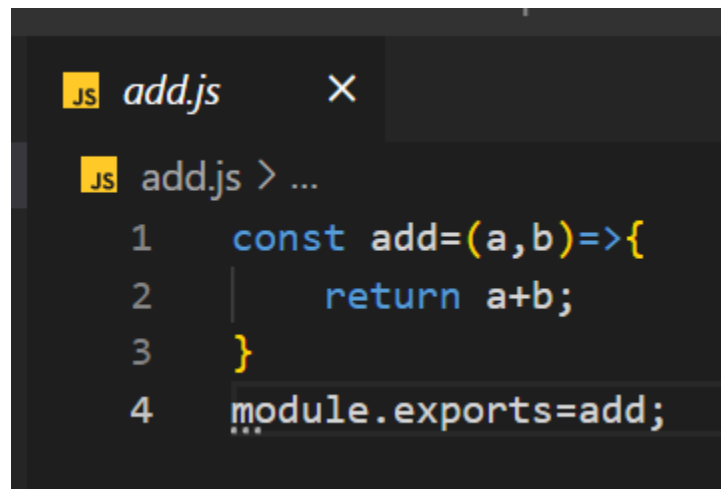
Step1: Install the node.js

<https://nodejs.org/en/download>

Step2: Create the project structure like a folder and create index.js file

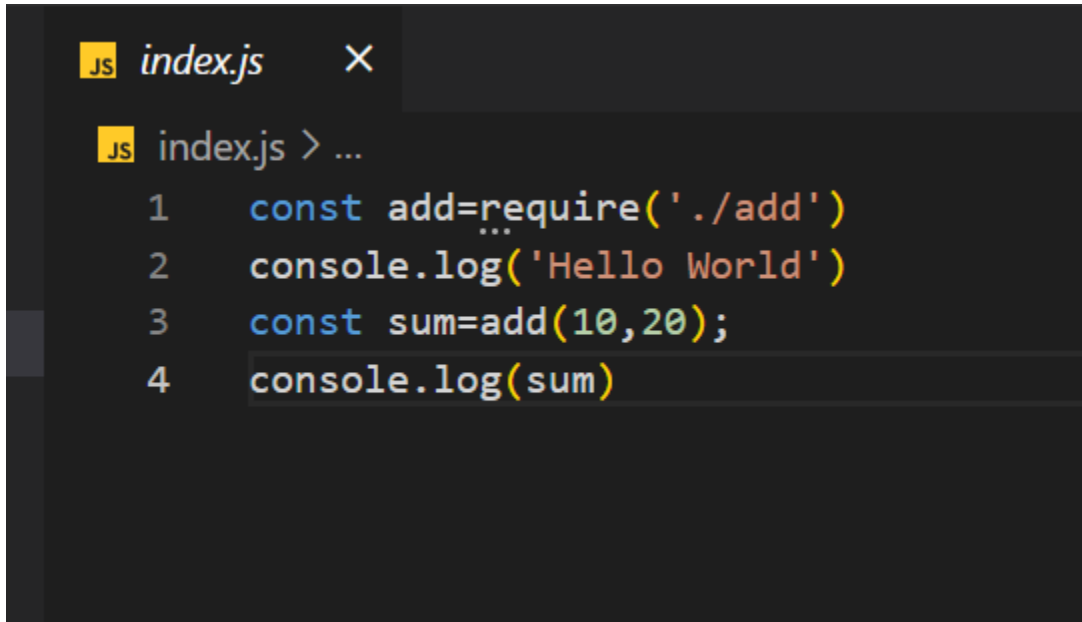
Step3: Inside the index.js file write the below code:

I.add.js

A screenshot of a code editor window with a dark theme. The window title is 'add.js' with a yellow 'JS' icon and a close button. The editor shows the following code:

```
1  const add=(a,b)=>{
2      return a+b;
3  }
4  module.exports=add;
```

II.index.js



```
JS index.js ×  
JS index.js > ...  
1  const add=require('./add')  
2  console.log('Hello World')  
3  const sum=add(10,20);  
4  console.log(sum)
```

Node.js File System(FS)

- In Node.js, file I/O is provided by simple wrappers around standard POSIX functions. Node File System (fs) module can be imported using following syntax:
 - var fs = require('fs');

Note: Each method in fs module has synchronous and asynchronous forms.

1. Example Program by using synchronous

```
JS index.js X
JS index.js > ...
1  const fs=require('fs')
2  //1. By using Synchronous
3  var data = fs.readFileSync('myfile.txt');
4  console.log(data.toString())
```

2. By using Asynchronous

Step1 : Create the myfile.txt

- Add some text

Step2: Create the index.js file

```
JS index.js X
JS index.js > 📦 callback
1  const fs=require('fs')
2  //1. By using Synchronous
3  var data = fs.readFile('myfile.txt',callback)
4
5  function callback(err,data){
6      if(err)
7          console.err('some error')
8      else
9          console.log(data.toString())
10 }
11
```

Node.js Http

- In order to make HTTP requests in Node.js, there is a built-in module HTTP in Node.js to transfer data over the HTTP. To use the HTTP server in the node, we need to require the HTTP module. The HTTP module creates an HTTP server that listens to server ports and gives a response back to the client.
- syntax:
- `const http=require('http')`
- Example:

```
JS workwithhttp.js X
JS workwithhttp.js > [?] server > [?] http.createServer() callback
1  const http=require('http')
2  const server=http.createServer((req,res)=>{
3      console.log('http://localhost:5000/');
4      if(req.url=='/home'){
5          res.write("Welcome to Home Page")
6      }else if(req.url=='/') {
7          res.write("WELCOME TO HOME PAGE...");
8      }else{
9          res.write(`
10             <h1>Wrong ULR</h1>
11             <a href='/'>Click Here to Home</a>
12             `)
13      }
14      res.end();
15  })
16  server.listen(5000)
```

Express.js

- It is a small framework that works on top of Node.js web server functionality to simplify its APIs and add helpful new features. It makes it easier to organize your application's functionality with middleware and routing. It adds helpful utilities to Node.js HTTP objects and facilitates the rendering of dynamic HTTP objects.

Why We need Express.js

- Develops Node.js web applications quickly and easily.
- It's simple to set up and personalise.
- Allows you to define application routes using HTTP methods and URLs.
- Includes a number of middleware modules that can be used to execute additional requests and responses activities.
- Simple to interface with a variety of template engines, including Jade, Vash, and EJS.
- Allows you to specify a middleware for handling errors.

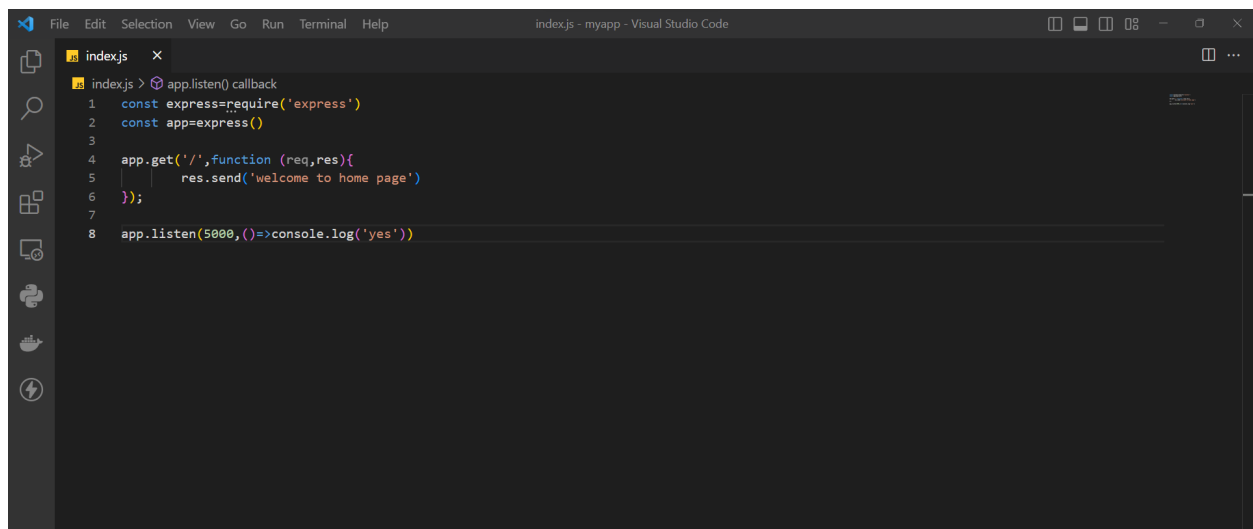
Example :

Step1: Install the express.js

> npm init

> npm i express —save

Step2: Create the index.js file

A screenshot of the Visual Studio Code editor interface. The title bar at the top reads "index.js - myapp - Visual Studio Code". The editor window shows a file named "index.js" with the following JavaScript code:

```
index.js > app.listen() callback
1  const express=require('express')
2  const app=express()
3
4  app.get('/',function (req,res){
5      res.send('welcome to home page')
6  });
7
8  app.listen(5000,()=>console.log('yes'))
```

The code is color-coded: 'const' is blue, 'require' is orange, 'function' is green, 'res.send' is blue, and 'console.log' is orange. The left sidebar shows the Explorer, Search, and Run and Debug views. The Explorer view is active, showing the file structure with 'index.js' listed.

