Visual Studio Code: Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

Working With Visual Studio Code

press windowskey+r,type cmd,press enter

C:\users\nipuna>cd\

C:\>md htmlworks

C:\>cd htmlworks

C:\>code .

click New File

type any filename.html Ex: mydemos.html

type the code,press ctrl+s

click Run

click Start Debugging

click on the browser name

create a css file Ex: backstyle.css

body

{

background-color:red;

}

save it and link it in the above html file

<head>

<title>demo</title>

<link rel="stylesheet" type="text/css" href="backstyle.css">

<meta charset="UTF-8">

</head>

create a javascript file Ex: test.js

function greetone()

{

alert("welcome to my site")

}

Save and Link It In html file

Code after linking css and javascript in HTML file

<!DOCTYPE html>

<html>

<head>

<title>demo</title>

<link rel="stylesheet" type="text/css" href="backstyle.css">

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

<meta charset="UTF-8">

</head>

<body onLoad="greetone()">

To Daay I am writing code in visual studio code.

</body>

</html>

Deleting Entire work

close visual studio

C:\htmlworks>cd\

C:\>rd htmlworks /s

press y

install Node JS 14

in google.com type node js 14.20.0 download for windows

click Node v14.20.0 (LTS)

click the link at Windows 64-bit Installer: https://nodejs.org/dist/v14.20.0/node-v14.20.0-x64.msi

install the software

C:\Users\Nipuna>node --version

v14.2.0

C:\Users\Nipuna>npm -v

6.14.4

C:\Users\Nipuna>npm install -g @angular/cli@^8.0.0

Angular CLI: 8.0.0

C:\Users\Nipuna>ng version

Working with Angular

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

? Would you like to add Angular routing? Yes

press enter

press enter

C:\demo>cd databind-app

C:\demo\databind-app>npm install querystring

PS C:\demo\databind-app> ng serve

press ctrl, click on http://localhost:4200

You will see the output on browser

deleting App

close the browser window

press ctrl+c to ternminate the app

C:\demo\databind-app>cd\

C:\>rd demo /s

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component test

C:\demo\databind-app>code . src-->App-->test

test.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.scss']

})

export class TestComponent implements OnInit {

appName = "My first app in Angular";

constructor() { }

ngOnInit() {

}

}

test.component.html

<h1>{{appName}}</h1>

app.component.html

<app-test></app-test>

C:\demo\databind-app>npm install querystring

C:\demo\databind-app>ng serve

ctrl+mouse click on url to see the output on the browser

----------------------------------------------------------------------------------------------------------------------------

Example 2

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component test

C:\demo\databind-app>code . src-->App-->test

test.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.scss']

})

export class TestComponent implements OnInit

{

ngOnInit(): void {

throw new Error('Method not implemented.');

}

showData($event: any)

{

console.log("button is clicked!"); if($event) {

console.log($event.target);

console.log($event.target.value);

}

}

}

test.component.html

<h2>Event Binding</h2>

<button (click)="showData($event)">Click here</button

app.component.html

<app-test></app-test>

C:\demo\databind-app>npm install querystring

C:\demo\databind-app>ng serve

press F12,click the button, click console,see the message button is clicked!

-------------------------------------------------------------------------------------------------------------

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component test

C:\demo\databind-app>code . src-->App-->test

test.component.ts

test.compoents.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.scss']

})

export class TestComponent implements OnInit

{

userName:string = "Peter";

ngOnInit(): void {

throw new Error('Method not implemented.');

}

}

test.component.html

<input type="text" [value]="userName">

app.component.html

<app-test></app-test>

C:\demo\databind-app>npm install querystring

C:\demo\databind-app>ng serve

--------------------------------------------------------------------------------------------------------------

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component test

C:\demo\databind-app>code . src-->App-->test

test.component.ts

test.component.css

.red {

color: red;

}

.blue {

color: blue;

}

test.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.css']

})

export class TestComponent {

myCSSClass = "red";

applyCSSClass = false;

}

test.component.html

<p [class]="myCSSClass">This paragraph class comes from \*myClass\* property </p>

<p [class.blue]="applyCSSClass">This paragraph class does not apply</p>

app.component.html

<app-test></app-test>

C:\demo\databind-app>npm install querystring

C:\demo\databind-app>ng serve

-----------------------------------------------------------------------------------------

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component test

C:\demo\databind-app>code . src-->App-->test

test.component.ts

myColor = 'brown';

test.component.html.

<p [style.color]="myColor">Text color is styled using style binding</p>

app.component.html

C:\demo\databind-app>npm install querystring

C:\demo\databind-app>ng serve

--------------------------------------------------------------------------------------------------------------

Two Way Data Binding

src/app/app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { TestComponent } from './test/test.component';

import { FormControl, FormsModule, ReactiveFormsModule } from '@angular/forms';

@NgModule({

declarations: [

AppComponent,

TestComponent

],

imports: [

BrowserModule,

FormsModule,

ReactiveFormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

test.component.html

<input type="text" [(ngModel)]="userName" />

<p>Two way binding! Hello {{ userName }}!</p>

tsconfig.json

{

"compileOnSave": false,

"compilerOptions": {

"baseUrl": "./",

"outDir": "./dist/out-tsc",

"sourceMap": true,

"declaration": false,

"module": "esnext",

"moduleResolution": "node",

"emitDecoratorMetadata": true,

"experimentalDecorators": true,

"importHelpers": true,

"target": "es2015",

"typeRoots": [

"node\_modules/@types"

],

"lib": [

"es2018",

"dom"

]

}

}

app.component.html

<app-test></app-test>

C:\demo\databind-app>npm install querystring

C:\demo\databind-app>ng serve

--------------------------------------------------------------------------------------------------------------

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component test

C:\demo>code .

test.componet.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

constructor() { }

age=45;

ngOnInit() {

}

}

test.component.html

<p>Age={{age}}</p>

app.component.html

<app-test></app-test>

C:\demo\databind-app> npm install querystring

C:\demo\databind-app>ng serve

----------------------------------------------------------------------------------------------------------------------

Arithmetical Operations

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component test

C:\demo>code .

test.componet.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

constructor() { }

ngOnInit() {

}

add(a: number, b: number): number {

return a + b;

}

subtract(a: number, b: number): number {

return a - b;

}

multiply(a: number, b: number): number {

return a \* b;

}

divide(a: number, b: number): number {

if (b === 0) {

return NaN; // Handle division by zero

}

return a / b;

}

modulus(a: number, b: number): number {

if (b === 0) {

return NaN; // Handle division by zero

}

return a % b;

}

expo(a: number, b: number): number {

if (b === 0) {

return NaN; // Handle division by zero

}

return a \*\* b;

}

}

test.component.html

<p>Result of arithmetic operations:</p>

<ul>

<li>Addition: {{ add(5, 3) }}</li>

<li>Subtraction: {{ subtract(8, 2) }}</li>

<li>Multiplication: {{ multiply(4, 6) }}</li>

<li>Division: {{ divide(10, 2) }}</li>

<li>Modulus: {{ modulus(10, 2) }}</li>

<li>Exponentiation: {{ expo(5, 3) }}</li>

</ul>

app.component.html

<body style="background-color:red;height:400px;">

<app-test></app-test>

</body>

C:\demo\databind-app> npm install querystring

C:\demo\databind-app>ng serve

Directives

ngmodel

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

app.component.html

<p>Input something in the input box:</p>

<p>Name: <input type="text" [(ngModel)]="firstName"></p>

<p>You wrote: {{ firstName }}</p>

app.component.ts

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

firstName = 'John';

}

app.module.ts

import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms'; // Import this

import { AppComponent } from './app.component';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

FormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

----------------------------------------------

ng generate component cost-calculator

cost-calculator.component.html

<h2>Cost Calculator</h2>

Quantity: <input type="number" [(ngModel)]="quantity">

Price: <input type="number" [(ngModel)]="price">

<p><b>Total in dollar:</b> {{ quantity \* price }}</p>

cost-calculator.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-cost-calculator',

templateUrl: './cost-calculator.component.html',

styleUrls: ['./cost-calculator.component.css']

})

export class CostCalculatorComponent implements OnInit {

quantity = 1;

price = 5;

constructor() { }

ngOnInit(): void {

}

}

/app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

import { CostCalculatorComponent } from './cost-calculator/cost-calculator.component';

@NgModule({

declarations: [

AppComponent,

CostCalculatorComponent

],

imports: [

BrowserModule,

FormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

app.component.html

<app-cost-calculator></app-cost-calculator>

ng serve --open

------------------------------------------------------------------------

ng-repeat

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component name-list

C:\demo\databind-app>code .

name-list.component.html

<p>Looping with ngFor:</p>

<ul>

<li \*ngFor="let name of names">

{{ name }}

</li>

</ul>

name-list.component.ts

import { Component } from '@angular/core';

@Component({

selector: 'app-name-list',

templateUrl: './name-list.component.html',

styleUrls: ['./name-list.component.css']

})

export class NameListComponent {

names: string[] = ['Jani', 'Hege', 'Kai'];

}

app.component.html

<app-name-list></app-name-list>

ng serve --open

-----------------------------------------

ng generate component object-list

object-list.component.html

<p>Looping with objects:</p>

<ul>

<li \*ngFor="let obj of names">

{{ obj.name }}, {{ obj.country }}

</li>

</ul>

Replace the content of src/app/object-list/object-list.component.ts with the following:

typescript

Copy code

import { Component } from '@angular/core';

@Component({

selector: 'app-object-list',

templateUrl: './object-list.component.html',

styleUrls: ['./object-list.component.css']

})

export class ObjectListComponent {

names = [

{ name: 'Jani', country: 'Norway' },

{ name: 'Hege', country: 'Sweden' },

{ name: 'Kai', country: 'Denmark' }

];

}

app.module.ts.

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

import { ObjectListComponent } from './object-list/object-list.component';

@NgModule({

declarations: [

AppComponent,

ObjectListComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

app.component.html

<app-object-list></app-object-list>

ng serve --open

---------------------------------------------------------------------

ngIf

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\>databind-app>ng generate component ngif-example

C:\demo\>databind-app>code .

ngif-example.component.html

<div \*ngIf="isVisible">

<p>This paragraph will only be visible if isVisible is true.</p>

</div>

<button (click)="toggleVisibility()">Toggle Visibility</button>

ngif-example.component.ts

import { Component } from '@angular/core';

@Component({

selector: 'app-ngif-example',

templateUrl: './ngif-example.component.html',

styleUrls: ['./ngif-example.component.css']

})

export class NgifExampleComponent {

isVisible: boolean = true;

toggleVisibility() {

this.isVisible = !this.isVisible;

}

}

app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

import { NgifExampleComponent } from './ngif-example/ngif-example.component';

@NgModule({

declarations: [

AppComponent,

NgifExampleComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

app.component.html

<app-ngif-example></app-ngif-example>

ng serve --open

------------------------------------------------

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\>databind-app>ng generate component ngif-else-example

C:\demo\>databind-app>code .

ngif-else-example.component.html

<div \*ngIf="isLogged; else notLoggedIn">

<p>Welcome, user!</p>

</div>

<ng-template #notLoggedIn>

<p>Please log in to access the content.</p>

</ng-template>

<button (click)="toggleLogin()">Toggle Login</button>

ngif-else-example.component.ts

import { Component } from '@angular/core';

@Component({

selector: 'app-ngif-else-example',

templateUrl: './ngif-else-example.component.html',

styleUrls: ['./ngif-else-example.component.css']

})

export class NgifElseExampleComponent {

isLogged: boolean = false;

toggleLogin() {

this.isLogged = !this.isLogged;

}

}

app.component.html

<app-ngif-else-example></app-ngif-else-example>

npm install querystring

ng serve

--------------------------------------------------------------------

Create a new Angular component:

bash

Copy code

ng generate component largest-number

Replace the content of src/app/largest-number/largest-number.component.html with the following:

html

Copy code

<div>

<label for="num1">Number 1:</label>

<input type="number" id="num1" [(ngModel)]="number1">

</div>

<div>

<label for="num2">Number 2:</label>

<input type="number" id="num2" [(ngModel)]="number2">

</div>

<div \*ngIf="number1 !== null && number2 !== null">

<p \*ngIf="number1 > number2; else secondIsLargest">The largest number is: {{ number1 }}</p>

<ng-template #secondIsLargest>

<p>The largest number is: {{ number2 }}</p>

</ng-template>

</div>

Replace the content of src/app/largest-number/largest-number.component.ts with the following:

typescript

Copy code

import { Component } from '@angular/core';

@Component({

selector: 'app-largest-number',

templateUrl: './largest-number.component.html',

styleUrls: ['./largest-number.component.css']

})

export class LargestNumberComponent {

number1: number = null;

number2: number = null;

}

Make sure to add LargestNumberComponent to the declarations array of the AppModule in src/app/app.module.ts.

typescript

Copy code

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

import { LargestNumberComponent } from './largest-number/largest-number.component';

@NgModule({

declarations: [

AppComponent,

LargestNumberComponent

],

imports: [

BrowserModule,

FormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

Finally, in src/app/app.component.html, replace the content with the following:

html

Copy code

<app-largest-number></app-largest-number>

Now you can run your Angular 8 application:

bash

Copy code

ng serve --open

----------------------------------------------------------------------------------------

test.component.html.

<p>test works!</p>

<div \*ngIf="true">Display data</div>

Add the test component in your app.component.html file as follows −

<app-test></app-test>

Start your server (if not started already) using the below command −

ng serve

test.component.ts file.

export class TestComponent implements OnInit {

isLogIn : boolean = false;

isLogOut : boolean = true;

}

Add the following code in test.component.html file as follows −

<p>ngIfElse example!</p>

<div \*ngIf="isLogIn; else isLogOut">

Hello you are logged in

</div>

<ng-template #isLogOut>

You're logged out..

</ng-template>

Finally, start your application (if not done already) using the below command −

ng serve

ttest.component.ts

list = [1,2,3,4,5];

test.component.html

<h2>ngFor directive</h2>

<ul>

<li \*ngFor="let l of list">

{{l}}

</li>

</ul>

-------------------------------------------------------------------

test.component.ts

test.component.ts file.

export class TestComponent {

studentArr: any[] = [ {

"id": 1,

"name": "student1"

},

{

"id": 2,

"name": "student2"

},

{

"id": 3, "name": "student3"

},

{

"id": 4,

"name": "student4"

}

];

trackByData(index:number, studentArr:any): number {

return studentArr.id;

}

test.component.html file

<ul>

<li \*ngFor="let std of studentArr; trackBy: trackByData">

{{std.name}}

</li>

</ul>

-----------------------------------------------------

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\>databind-app>ng generate component test

C:\demo\>databind-app>code .

test.component.ts file.

export class TestComponent implements OnInit {

logInName = 'admin';

}

test.component.html file as follows −

<h2>ngSwitch directive</h2>

<ul [ngSwitch]="logInName">

<li \*ngSwitchCase="'user'">

<p>User is logged in..</p>

</li>

<li \*ngSwitchCase="'admin'">

<p>admin is logged in</p>

</li>

<li \*ngSwitchDefault>

<p>Please choose login name</p>

</li>

</ul>

app.component.html

<app-test></app-test>

npm install querystring

ng serve

---------------------------------------------------------------------------

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component test

C:\demo\databind-app>code .

test.component.html file.

<p [ngStyle]="{'color': 'blue', 'font-size': '14px'}">

paragraph style is applied using ngStyle

</p>

app.component.html

<app-test></app-test>

C:\demo\databind-app>npm install querystring

C:\demo\databind-app>ng serve

--------------------------------------------------

C:\demo\databind-app>ng g class User

Move to src/app/user.ts file and add the below code −

export class User {

userId : number; userName : string;

}

Open test.component.ts file and add the below changes −

import { User } from '../user';

export class TestComponent implements OnInit {

users: User[] = [

{

"userId": 1,

"userName": 'User1'

},

{

"userId": 2,

"userName": 'User2'

},

];

}

Here, we have declared a local variable, users and initialise with 2 users object.

Open test.component.css file and add below code

.highlight {

color: red;

}

Open your test.component.html file and add the below code −

<div class="container">

<br/>

<div \*ngFor="let user of users" [ngClass]="{

'highlight':user.userName === 'User1'

}">

{{ user.userName }}

</div>

</div>

ng serve

------------------------------------------

Custom directives

ng generate directive customstyle

Open app.module.ts. The directive will be configured in the AppModule through declarations meta data.

import { CustomstyleDirective } from './customstyle.directive';

@NgModule({

declarations: [

AppComponent,

TestComponent,

CustomstyleDirective

]

})

Open customstyle.directive.ts file and add the below code −

import { Directive, ElementRef } from '@angular/core';

@Directive({

selector: '[appCustomstyle]'

})

export class CustomstyleDirective {

constructor(el: ElementRef) {

el.nativeElement.style.fontSize = '24px';

}

}

Here, constructor method gets the element using CustomStyleDirective as el. Then, it accesses el’s style and set its font size as 24px using CSS property.

Finally, start your application (if not done already) using the below command −

ng serve

Now, run your application and you could see the below response −

Custom directives

ng-template

ng-template is used to create dynamic and reusable templates. It is a virtual element. If you compile your code with ng-template then is converted as comment in DOM.

For example,

Let’s add a below code in test.component.html page.

<h3>ng-template</h3>

<ng-template>ng-template tag is a virtual element</ng-template>

If you run the application, then it will print only h3 element. Check your page source, template is displayed in comment section because it is a virtual element so it does not render anything. We need to use ng-template along with Angular directives.

Normally, directive emits the HTML tag it is associated. Sometimes, we don’t want the tag but only the content. For example, in the below example, li will be emitted.

<li \*ngFor="let item in list">{{ item }}</li>

We can use ng-template to safely skip the li tag.

ng-template with structural directive

ng-template should always be used inside ngIf, ngFor or ngSwitch directives to render the result.

Let’s assume simple code.

<ng-template [ngIf]=true>

<div><h2>ng-template works!</h2></div>

</ng-template>

Here, if ngIf condition becomes true, it will print the data inside div element. Similarly, you can use ngFor and ngSwitch directives as well.

NgForOf directive

ngForOf is also a structural directive used to render an item in a collection. Below example is used to show ngForOf directive inside ng-template.

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-test',

template: `

<div>

<ng-template ngFor let-item [ngForOf]="Fruits" let-i="index">

<p>{{i}}</p>

</ng-template>

</div>`

,

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

Fruits = ["mango","apple","orange","grapes"];

ngOnInit()

{

}

}

If you run the application, it will show the index of each elements as shown below −

0

1

2

3

Component directives

Component directives are based on component. Actually, each component can be used as directive. Component provides @Input and @Output decorator to send and receive information between parent and child components.

Let us try use component as directive in our directive-app application.

Create a new ChildComponent using below command −

ng generate component child

CREATE src/app/child/child.component.html (20 bytes) CREATE src/app/child/child.component.spec.ts (621 bytes)

CREATE src/app/child/child.component.ts (265 bytes) CREATE src/app/child/child.component.css (0 bytes) UPDATE src/app/app.module.ts (466 bytes)

Open child.component.ts and add below code −

@Input() userName: string;

Here, we are setting a input property for ChildComponent.

Open child.component.html and add below code −

<p>child works!</p>

<p>Hi {{ userName }}</p>

Here, we are using the value userName to welcome the user.

Open test.component.ts and add below code −

name: string = 'Peter';

Open test.component.html and add below code −

<h1>Test component</h1>

<app-child [userName]="name"><app-child>

Here, we are using AppComponent inside the TestComponent as a directive with input property.

Finally, start your application (if not done already) using the below command −

ng serve

Now, run your application and you could see the below response −

[](images/directive-app/component\_as\_directive.PNG"

Working example

Let us add a new component in our ExpenseManager application to list the expense entries.

Open command prompt and go to project root folder.

cd /go/to/expense-manager

Start the application.

ng serve

Create a new component, ExpenseEntryListComponent using below command −

ng generate component ExpenseEntryList

Output

The output is as follows −

CREATE src/app/expense-entry-list/expense-entry-list.component.html (33 bytes)

CREATE src/app/expense-entry-list/expense-entry-list.component.spec.ts (700 bytes)

CREATE src/app/expense-entry-list/expense-entry-list.component.ts (315 bytes)

CREATE src/app/expense-entry-list/expense-entry-list.component.css (0 bytes)

UPDATE src/app/app.module.ts (548 bytes)

Here, the command creates the ExpenseEntryList Component and update the necessary code in AppModule.

Import ExpenseEntry into ExpenseEntryListComponent component (src/app/expense-entry-list/expense-entry-list.component)

import { ExpenseEntry } from '../expense-entry';

Add a method, getExpenseEntries() to return list of expense entry (mock items) in ExpenseEntryListComponent (src/app/expense-entry-list/expense-entry-list.component)

getExpenseEntries() : ExpenseEntry[] {

let mockExpenseEntries : ExpenseEntry[] = [

{ id: 1,

item: "Pizza",

amount: Math.floor((Math.random() \* 10) + 1),

category: "Food",

location: "Mcdonald",

spendOn: new Date(2020, 4, Math.floor((Math.random() \* 30) + 1), 10, 10, 10),

createdOn: new Date(2020, 4, Math.floor((Math.random() \* 30) + 1), 10, 10, 10) },

{ id: 1,

item: "Pizza",

amount: Math.floor((Math.random() \* 10) + 1),

category: "Food",

location: "KFC",

spendOn: new Date(2020, 4, Math.floor((Math.random() \* 30) + 1), 10, 10, 10),

createdOn: new Date(2020, 4, Math.floor((Math.random() \* 30) + 1), 10, 10, 10) },

{ id: 1,

item: "Pizza",

amount: Math.floor((Math.random() \* 10) + 1),

category: "Food",

location: "Mcdonald",

spendOn: new Date(2020, 4, Math.floor((Math.random() \* 30) + 1), 10, 10, 10),

createdOn: new Date(2020, 4, Math.floor((Math.random() \* 30) + 1), 10, 10, 10) },

{ id: 1,

item: "Pizza",

amount: Math.floor((Math.random() \* 10) + 1),

category: "Food",

location: "KFC",

spendOn: new Date(2020, 4, Math.floor((Math.random() \* 30) + 1), 10, 10, 10),

createdOn: new Date(2020, 4, Math.floor((Math.random() \* 30) + 1), 10, 10, 10) },

{ id: 1,

item: "Pizza",

amount: Math.floor((Math.random() \* 10) + 1),

category: "Food",

location: "KFC",

spendOn: new Date(2020, 4, Math.floor((Math.random() \* 30) + 1), 10, 10, 10),

createdOn: new Date(2020, 4, Math.floor((Math.random() \* 30) + 1), 10, 10, 10)

},

];

return mockExpenseEntries;

}

Declare a local variable, expenseEntries and load the mock list of expense entries as mentioned below −

title: string;

expenseEntries: ExpenseEntry[];

constructor() { }

ngOnInit() {

this.title = "Expense Entry List";

this.expenseEntries = this.getExpenseEntries();

}

Open the template file (src/app/expense-entry-list/expense-entry-list.component.html) and show the mock entries in a table.

<!-- Page Content -->

<div class="container">

<div class="row">

<div class="col-lg-12 text-center" style="padding-top: 20px;">

<div class="container" style="padding-left: 0px; padding-right: 0px;">

<div class="row">

<div class="col-sm" style="text-align: left;">

{{ title }}

</div>

<div class="col-sm" style="text-align: right;">

<button type="button" class="btn btn-primary">Edit</button>

</div>

</div>

</div>

<div class="container box" style="margin-top: 10px;">

<table class="table table-striped">

<thead>

<tr>

<th>Item</th>

<th>Amount</th>

<th>Category</th>

<th>Location</th>

<th>Spent On</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let entry of expenseEntries">

<th scope="row">{{ entry.item }}</th>

<th>{{ entry.amount }}</th>

<td>{{ entry.category }}</td>

<td>{{ entry.location }}</td>

<td>{{ entry.spendOn | date: 'short' }}</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

</div>

Here,

Used bootstrap table. table and table-striped will style the table according to Boostrap style standard.

Used ngFor to loop over the expenseEntries and generate table rows.

Open AppComponent template, src/app/app.component.html and include ExpenseEntryListComponent and remove ExpenseEntryComponent as shown below −

...

<app-expense-entry-list></app-expense-entry-list>

Finally, the output of the application is as shown below.

-----------------------------------------------------------------------------------------------------

PIPES

Pipes are referred as filters. It helps to transform data and manage data within interpolation, denoted by {{ | }}. It accepts data, arrays, integers and strings as inputs which are separated by ‘|’ symbol.

Program to Display Date

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component test

C:\demo\databind-app>code .

test.component.ts file.

export class TestComponent {

presentDate = new Date();

}

test.component.html file.

<div>

Today's date :- {{presentDate}}

</div>

app.component.html

<app-test></app-test>

C:\demo\databind-app>npm install querystring

C:\demo\databind-app>ng serve

Add Date pipe: Let’s add date pipe in the above html file.

<div>

Today's date :- {{presentDate | date }}

</div>

Example 3

test.component.html to see date and time in different formats

<div>

Today's date :- {{presentDate}}<br/>

Today's date :- {{presentDate | date }}<br/>

short date :- {{presentDate | date:'shortDate' }} <br/>

Full date :- {{presentDate | date:'fullDate' }} <br/>

Formatted date:- {{presentDate | date:'M/dd/yyyy'}} <br/>

Hours and minutes:- {{presentDate | date:'h:mm'}}

</div>

Chained pipes: We can combine multiple pipes together. This will be useful when a scenario associates with more than one pipe that has to be applied for data transformation.

<div>

Date with uppercase :- {{presentDate | date:'fullDate' | uppercase}} <br/>

Date with lowercase :- {{presentDate | date:'medium' | lowercase}} <br/>

</div>

AsyncPipe: If data comes in the form of observables, then Async pipe subscribes to an observable and returns the transmitted values.

C:\>md demo

C:\>cd demo

C:\demo>ng new databind-app

C:\demo>cd databind-app

C:\demo\databind-app>ng generate component test

C:\demo\databind-app>code .

test.component.ts

import { Component, OnInit } from '@angular/core';

import { Observable } from 'rxjs';

import { Observer } from 'rxjs';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

appName = "My first app in Angular 8";

currentTime: Observable<string>;

constructor() { }

ngOnInit() {

this.currentTime = new Observable<string>((observer) => {

setInterval(() => {

observer.next(new Date().toString());

}, 1000);

});

}

}

test.component.html.

<div>

<p>Current Time: {{ currentTime | async }}</p>

</div>

Add the below code inside your app.component.html

C:\demo\databind-app>npm install querystring

C:\demo\databind-app>ng serve

CurrencyPipe: It is used to convert the given number into various countries currency format.

test.component.ts

import { Component, OnInit } from '@angular/core';

import { Observable } from 'rxjs';

import { Observer } from 'rxjs';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

price : number = 20000;

constructor() { }

ngOnInit() {}

}

test.component.html

<div>

<h3> Currency Pipe</h3>

<p>{{price}}</p>

<p>{{ price | currency:'EUR':true}}</p>

<p>{{ price | currency:'INR' }}</p>

</div>

app.component.html

<app-test></app-test>

SlicePipe: Slice pipe is used to return a slice of an array.

test.component.ts

import { Component, OnInit } from '@angular/core';

import { Observable } from 'rxjs';

import { Observer } from 'rxjs';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

Fruits = ["Apple","Orange","Grapes","Mango","Kiwi","Pomegranate"];

constructor() { }

ngOnInit() {}

}

test.component.html

<div>

<h3>Start index:- {{Fruits | slice:2}}</h3>

<h4>Start and end index:- {{Fruits | slice:1:4}}</h4>

<h5>Negative index:- {{Fruits | slice:-2}}</h5>

<h6>Negative start and end index:- {{Fruits | slice:-4:-2}}</h6>

</div>

app.component.html

<app-test></app-test>

DecimalPipe: It is used to format decimal values. It is also considered as CommonModule.

test.component.ts

import { Component, OnInit } from '@angular/core';

import { Observable } from 'rxjs';

import { Observer } from 'rxjs';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

decimalNum1: number = 8.7589623;

decimalNum2: number = 5.43;

constructor() { }

ngOnInit()

{

}

}

test.component.html

<div>

<h3>Decimal Pipe</h3>

<p> {{decimalNum1 | number}} </p>

<p> {{decimalNum2 | number}} </p>

</div>

Formatting values

We can apply string format inside number pattern. It is based on the below format −

number:"{minimumIntegerDigits}.{minimumFractionDigits} - {maximumFractionDigits}"

Let’s apply the above format in our code,

@Component({

template: `

<div style="text-align:center">

<p> Apply formatting:- {{decimalNum1 | number:'3.1'}} </p>

<p> Apply formatting:- {{decimalNum1 | number:'2.1-4'}} </p>

</div>

`,

})

Here,