**https://app.smartsheet.com/b/form/0d4e989091fa4983a17d44e7fe1828c5**

**Developing Web Frontend with Angular**

**Introduction to Angular**

Angular is a popular **application-design framework and development platform** for creating efficient and sophisticated single-page applications (SPAs). [It’s built on **TypeScript** and maintained by a dedicated team at Google*1*](https://angular.dev/overview/). [Angular provides a broad suite of tools, APIs, and libraries to simplify and streamline your development workflow*1*](https://angular.dev/overview/).

**What is Angular?**

Angular is a **component-based framework** that helps in building scalable web applications. It includes a collection of well-integrated libraries that cover a wide variety of features, such as:

* **Routing**
* **Forms management**
* **Client-server communication**

**Key Features**

* **Component Architecture**: Angular organizes your project into manageable, well-organized parts with clear responsibilities.
* **TypeScript**: Most Angular code is written using TypeScript, which provides improved tooling and enhanced maintainability.
* **Developer Tools**: Angular comes with a suite of developer tools to help you develop, build, test, and update your code.

**TypeScript** is a powerful language that is commonly used in Angular applications. It’s a superset of JavaScript, which means any valid JavaScript code is also valid TypeScript code. [TypeScript provides static typing, which can help catch errors early in the development process and improve code quality and maintainability*1*](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/).

Angular’s architecture is designed to facilitate the development of complex client-side applications. It’s composed of several core concepts:

* **Modules**: These are the building blocks of an Angular app. An Angular application is a set of NgModules, which provide a compilation context for components. [Modules collect related code into functional sets; an Angular app is defined by a set of NgModules](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*2*](https://v2.angular.io/docs/ts/latest/guide/architecture.html).
* [**Components**: They are the main way we build and specify elements and logic on the page, through which we can display data and respond to user actions](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*3*](https://angular.io/guide/architecture).
* **Templates**: These define views in an Angular application. [They are a form of HTML that tells Angular how to render the component](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*3*](https://angular.io/guide/architecture).
* **Metadata**: This tells Angular how to process a class. [For example, what is its template, what selectors it responds to, etc](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*2*](https://v2.angular.io/docs/ts/latest/guide/architecture.html).
* **Data binding**: It’s the mechanism to coordinate parts of a template with parts of a component. [Angular supports two-way data binding, meaning that changes in the DOM, such as user choices, are also reflected in the component’s state](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*3*](https://angular.io/guide/architecture).
* [**Directives**: These are classes that add additional behavior to elements in your Angular applications](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*2*](https://v2.angular.io/docs/ts/latest/guide/architecture.html).
* **Services**: They are a wide category encompassing any value, function, or feature that an app needs. A service is typically a class with a narrow, well-defined purpose. [It should do something specific and do it well](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*3*](https://angular.io/guide/architecture).
* **Dependency Injection (DI)**: It’s a design pattern and mechanism for creating and delivering parts of an application to other parts of an application that request them. [Angular’s DI framework provides dependencies to a class upon instantiation](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*3*](https://angular.io/guide/architecture).

1. **Installing Node.js and npm**:
   * Node.js is a JavaScript runtime required to run Angular applications.
   * npm (Node Package Manager) is used to install Angular CLI and other Angular packages.
   * You can download and install Node.js and npm from the [Node.js official website](https://docs.npmjs.com/downloading-and-installing-node-js-and-npm/?ref=sfeir.dev). It’s recommended to use a Node version manager like nvm for easier management of Node.js versions.
2. **Installing Angular CLI**:
   * Angular CLI is a command-line interface tool that helps you create projects, generate application and library code, and perform a variety of ongoing development tasks.
   * Install Angular CLI globally using npm with the command:
   * npm install -g @angular/cli
   * This will allow you to use Angular CLI commands from any directory in your system.
3. **Creating a New Angular Project**:
   * Once Angular CLI is installed, you can create a new project using the ng new command followed by your project name, like so:
   * ng new my-angular-app
   * This command will create a new directory with all the necessary files and dependencies for starting an Angular application.
4. **Running Your Angular Project**:
   * Navigate into your project directory and start the development server using:
   * cd my-angular-app
   * ng serve --open
   * The ng serve command will compile the application and serve it on http://localhost:4200.
   * The --open (or -o) flag will automatically open your default web browser to this URL.

**Update NPM**

Angular requires NPM v6.11 or later. Check the NPM version on your local machine by opening a terminal/command window and type the following command:

>npm -v  
 6.14.4

This will display the version number of NPM installed on your local machine. If you don't have the latest version of NPM, then update it using the following command on Windows.

npm install -g npm

If you are on MAC, use sudo npm install -g npm command.

After installing Node.js and NPM, install the Angular CLI.

After installing Node.js and NPM, install the Angular CLI.

**Install Angular CLI**

Angular provides many libraries and packages for application development. You can install libraries required for your application using Angular CLI (Command Line Interface). Angular CLI is also used to generate, build, run, and deploy Angular application.

To install the Angular CLI globally using NPM, open a terminal/command window, and enter the following command:

npm install -g @angular/cli@latest

In the above command -g indicates global, so that you can use angular CLI anywhere on your local machine. @latest is specifies to install the latest verion of angular CLI.

**The following creates a new angular application named "FirstAngularApp**" in the AngularApps folder.

D:\AngularApps> ng new FirstAngularApp

The ng new command prompts you for information about features to include in the initial app project. You can accept the defaults by pressing an Enter key,

**Angular Component**

Angular Component = Html +component class+ metadata

Component class – Type script class ( properties and methods)

Metadata – some extra data about the component ( about Html files,css,,selectors,providers)

**Generate component**

**ng g component greetings**

**CREATE src/app/greetings/greetings.component.html (25 bytes)**

**CREATE src/app/greetings/greetings.component.spec.ts (636 bytes)**

**CREATE src/app/greetings/greetings.component.ts (258 bytes)**

**CREATE src/app/greetings/greetings.component.css (0 bytes)**

greetings component.ts

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

@Component({

selector:'app-greetings',

standalone:true,

imports: [],

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

styleUrl:'./greetings.component.css'

})

exportclass GreetingsComponent

{

constructor() { }

ngOnInit():void {

}

name:string= "rani";

greeting():void

{

alert("Hello "+this.name);

};

}

Greetings component.html

<p>greetings works!</p>

<div>

Enter Name :<inputtype="text"value = {{name}}/><br/>

<button(click)="greeting()">Greeting</button>

</div>

**Bootstrapping component**

We have to load the component but before that we have to load the root component this is called bootstrapping.

Add greeting component in the app module.ts.

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

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

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

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

import { GreetingsComponent } from './greetings/greetings.component';

@NgModule({

declarations: [

AppComponent,

GreetingsComponent

],

imports: [

BrowserModule,

AppRoutingModule

],

providers: [],

bootstrap: [AppComponent]

})

exportclass AppModule { }

Add the greeting component tag in the root html template

App.component.html

<div><app-greetings></app-greetings></div>

**Html Template**

--------- **Inline Template**

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

@Component({

selector:'app-greetings',

template:" Enter your name: <input value ={{name}}>"

})

exportclass GreetingsComponent implements OnInit{

constructor() { }

ngOnInit():void { }

name:string= "rani";

greeting():void

{

alert("Hello " + this.name);

};

}

**Event Handling**

Create a component event binding component

Eventbindingcomponent.ts

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

@Component({

selector:'app-event-bindingcomponent',

template:`<button (click)="show()">click</button>`

})

exportclass EventBindingcomponentComponentimplements OnInit {

constructor() { }

ngOnInit():void {

}

show()

{

alert("this is event demo");

}

}

appcomponent.html

<div><app-event-bindingcomponent></app-event-bindingcomponent></div>

**Developing Web Frontend with Angular**

**Introduction to Angular**

Angular is a popular **application-design framework and development platform** for creating efficient and sophisticated single-page applications (SPAs). [It’s built on **TypeScript** and maintained by a dedicated team at Google*1*](https://angular.dev/overview/). [Angular provides a broad suite of tools, APIs, and libraries to simplify and streamline your development workflow*1*](https://angular.dev/overview/).

**What is Angular?**

Angular is a **component-based framework** that helps in building scalable web applications. It includes a collection of well-integrated libraries that cover a wide variety of features, such as:

* **Routing**
* **Forms management**
* **Client-server communication**

**Key Features**

* **Component Architecture**: Angular organizes your project into manageable, well-organized parts with clear responsibilities.
* **TypeScript**: Most Angular code is written using TypeScript, which provides improved tooling and enhanced maintainability.
* **Developer Tools**: Angular comes with a suite of developer tools to help you develop, build, test, and update your code.

**TypeScript** is a powerful language that is commonly used in Angular applications. It’s a superset of JavaScript, which means any valid JavaScript code is also valid TypeScript code. [TypeScript provides static typing, which can help catch errors early in the development process and improve code quality and maintainability*1*](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/).

Angular’s architecture is designed to facilitate the development of complex client-side applications. It’s composed of several core concepts:

* **Modules**: These are the building blocks of an Angular app. An Angular application is a set of NgModules, which provide a compilation context for components. [Modules collect related code into functional sets; an Angular app is defined by a set of NgModules](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*2*](https://v2.angular.io/docs/ts/latest/guide/architecture.html).
* [**Components**: They are the main way we build and specify elements and logic on the page, through which we can display data and respond to user actions](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*3*](https://angular.io/guide/architecture).
* **Templates**: These define views in an Angular application. [They are a form of HTML that tells Angular how to render the component](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*3*](https://angular.io/guide/architecture).
* **Metadata**: This tells Angular how to process a class. [For example, what is its template, what selectors it responds to, etc](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*2*](https://v2.angular.io/docs/ts/latest/guide/architecture.html).
* **Data binding**: It’s the mechanism to coordinate parts of a template with parts of a component. [Angular supports two-way data binding, meaning that changes in the DOM, such as user choices, are also reflected in the component’s state](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*3*](https://angular.io/guide/architecture).
* [**Directives**: These are classes that add additional behavior to elements in your Angular applications](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*2*](https://v2.angular.io/docs/ts/latest/guide/architecture.html).
* **Services**: They are a wide category encompassing any value, function, or feature that an app needs. A service is typically a class with a narrow, well-defined purpose. [It should do something specific and do it well](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*3*](https://angular.io/guide/architecture).
* **Dependency Injection (DI)**: It’s a design pattern and mechanism for creating and delivering parts of an application to other parts of an application that request them. [Angular’s DI framework provides dependencies to a class upon instantiation](https://www.geeksforgeeks.org/what-is-typescript-and-why-is-it-used-in-angular/)[*3*](https://angular.io/guide/architecture).

1. **Installing Node.js and npm**:
   * Node.js is a JavaScript runtime required to run Angular applications.
   * npm (Node Package Manager) is used to install Angular CLI and other Angular packages.
   * You can download and install Node.js and npm from the [Node.js official website](https://docs.npmjs.com/downloading-and-installing-node-js-and-npm/?ref=sfeir.dev). It’s recommended to use a Node version manager like nvm for easier management of Node.js versions.
2. **Installing Angular CLI**:
   * Angular CLI is a command-line interface tool that helps you create projects, generate application and library code, and perform a variety of ongoing development tasks.
   * Install Angular CLI globally using npm with the command:
   * npm install -g @angular/cli
   * This will allow you to use Angular CLI commands from any directory in your system.
3. **Creating a New Angular Project**:
   * Once Angular CLI is installed, you can create a new project using the ng new command followed by your project name, like so:
   * ng new my-angular-app
   * This command will create a new directory with all the necessary files and dependencies for starting an Angular application.
4. **Running Your Angular Project**:
   * Navigate into your project directory and start the development server using:
   * cd my-angular-app
   * ng serve --open
   * The ng serve command will compile the application and serve it on http://localhost:4200.
   * The --open (or -o) flag will automatically open your default web browser to this URL.

**Update NPM**

Angular requires NPM v6.11 or later. Check the NPM version on your local machine by opening a terminal/command window and type the following command:

>npm -v  
 6.14.4

This will display the version number of NPM installed on your local machine. If you don't have the latest version of NPM, then update it using the following command on Windows.

npm install -g npm

If you are on MAC, use sudo npm install -g npm command.

After installing Node.js and NPM, install the Angular CLI.

After installing Node.js and NPM, install the Angular CLI.

**Install Angular CLI**

Angular provides many libraries and packages for application development. You can install libraries required for your application using Angular CLI (Command Line Interface). Angular CLI is also used to generate, build, run, and deploy Angular application.

To install the Angular CLI globally using NPM, open a terminal/command window, and enter the following command:

npm install -g @angular/cli@latest

In the above command -g indicates global, so that you can use angular CLI anywhere on your local machine. @latest is specifies to install the latest verion of angular CLI.

**The following creates a new angular application named "FirstAngularApp**" in the AngularApps folder.

D:\AngularApps> ng new FirstAngularApp

The ng new command prompts you for information about features to include in the initial app project. You can accept the defaults by pressing an Enter key,

**Angular Component**

Angular Component = Html +component class+ metadata

Component class – Type script class ( properties and methods)

Metadata – some extra data about the component ( about Html files,css,,selectors,providers)

**Generate component**

**ng g component greetings**

**CREATE src/app/greetings/greetings.component.html (25 bytes)**

**CREATE src/app/greetings/greetings.component.spec.ts (636 bytes)**

**CREATE src/app/greetings/greetings.component.ts (258 bytes)**

**CREATE src/app/greetings/greetings.component.css (0 bytes)**

greetings component.ts

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

@Component({

selector:'app-greetings',

standalone:true,

imports: [],

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

styleUrl:'./greetings.component.css'

})

exportclass GreetingsComponent

{

constructor() { }

ngOnInit():void {

}

name:string= "rani";

greeting():void

{

alert("Hello "+this.name);

};

}

Greetings component.html

<p>greetings works!</p>

<div>

Enter Name :<inputtype="text"value = {{name}}/><br/>

<button(click)="greeting()">Greeting</button>

</div>

**Bootstrapping component**

We have to load the component but before that we have to load the root component this is called bootstrapping.

Add greeting component in the app module.ts.

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

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

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

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

import { GreetingsComponent } from './greetings/greetings.component';

@NgModule({

declarations: [

AppComponent,

GreetingsComponent

],

imports: [

BrowserModule,

AppRoutingModule

],

providers: [],

bootstrap: [AppComponent]

})

exportclass AppModule { }

Add the greeting component tag in the root html template

App.component.html

<div><app-greetings></app-greetings></div>

**Html Template**

--------- **Inline Template**

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

@Component({

selector:'app-greetings',

template:" Enter your name: <input value ={{name}}>"

})

exportclass GreetingsComponent implements OnInit{

constructor() { }

ngOnInit():void { }

name:string= "rani";

greeting():void

{

alert("Hello " + this.name);

};

}

**Event Handling**

Create a component event binding component

Eventbindingcomponent.ts

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

@Component({

selector:'app-event-bindingcomponent',

template:`<button (click)="show()">click</button>`

})

exportclass EventBindingcomponentComponentimplements OnInit {

constructor() { }

ngOnInit():void {

}

show()

{

alert("this is event demo");

}

}

appcomponent.html

<div><app-event-bindingcomponent></app-event-bindingcomponent></div>

**ngModel directive**

**Syntax:**

**[()] = []+()**

**[] – binds attribute**

**() – binds an event**

**Used for Two-way binding**

**One-way binding --- <input type = “name” [ngModel]=”value”>**

**Two-way binding --- <input type = “name” [(ngModel)]= “value”>**

Example

ngmodeldemocomponent.ts

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

@Component({

selector:'app-ngmodeledemo',

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

styleUrl:'./ngmodeledemo.component.css'

})

exportclass NgmodeledemoComponentimplements OnInit{

constructor() { }

ngOnInit():void {

}

isChecked:boolean=true;

mycolor:string= "red";

}

Ngmodeldemo.component.html

<inputtype = "checkbox"[(ngModel)]=" isChecked"> check status

<p>check status is : {{isChecked ? "true" : "false"}}</p>

<input[(ngModel)] ="mycolor" type = "text"placeholder="Enter color">

<p> my color is: {{mycolor}}</p>

App.module.ts

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

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

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

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

import { GreetingsComponent } from './greetings/greetings.component';

import { EventBindingcomponentComponent } from './event-bindingcomponent/event-bindingcomponent.component';

import { NgmodeledemoComponent } from './ngmodeledemo/ngmodeledemo.component';

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

@NgModule({

declarations: [

AppComponent,

GreetingsComponent,

EventBindingcomponentComponent,

NgmodeledemoComponent

],

imports: [

BrowserModule,

AppRoutingModule,

FormsModule

],

providers: [],

bootstrap: [AppComponent]

})

exportclass AppModule { }

**Interpolation**

{{}}

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

@Component({

selector:'app-interpolationdemo',

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

styleUrl:'./interpolationdemo.component.css'

})

exportclass InterpolationdemoComponentimplements OnInit {

constructor() { }

ngOnInit():void {

}

publictext:string="Hello";

randomNums:number[]=[1,2,3,4,5];

getCurrentTime():any{

returnnew Date().getDate();

}

}

Interpolationdemocomponent.html

<div><app-ngmodeledemo></app-ngmodeledemo></div>

<div><app-interpolationdemo></app-interpolationdemo></div>

**Module**

**Module** in Angular refers to a place where you can group the components, directives, pipes, and services, which are related to the application.

In case you are developing a website, the header, footer, left, center and the right section become part of a module.

**Usage:**

**Reusability**

* **Create project**
* **ng new projectname –no -standalone**
* **cd projectname**
* **ng generate module orders**
* **ng generate component orders/customerOrder --module=orders**

**customer.order.component.html**

<p>customer-order works!</p>

<h2> Customer order component is working</h2>

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

**customer.order.component.ts**

@Component({

selector:'app-customer-order',

templateUrl:'./customer-order.component.html',

styleUrl:'./customer-order.component.css'

})

exportclass CustomerOrderComponent {

}

Orders.modules.ts

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

import { CommonModule } from'@angular/common';

import { CustomerOrderComponent } from './customer-order/customer-order.component';

@NgModule({

declarations: [

CustomerOrderComponent

],

imports: [

CommonModule

],

exports: [

CustomerOrderComponent

]

})

exportclass OrdersModule { }

app.module.ts

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

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

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

import { ProductsModule } from'./products/products.module';

import { OrdersModule } from'./orders/orders.module';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

ProductsModule,

OrdersModule

],

providers: [],

bootstrap: [AppComponent]

})

exportclass AppModule { }

appcomponent.html

<app-product-item></app-product-item> //from product module

<app-customer-order></app-customer-order> // from order module

**Pipes**

Angular is a framework that lets us create interactive web frontends for users in an organized way. One entity that it has are pipes.

Pipes let us render items in component templates in the way we want. Angular comes with a few built-in pipes to let us format data for display.

**Built-in Pipes**

Angular comes with the following pipes:

* The AsyncPipe lets us take the resolve value of promises or observables and display them on the screen.
* CurrencyPipe lets us format numbers into a currency value for displaying on the screen.
* DatePipe lets us format dates.
* I18PluralPipe lets us get the plural version of a word for a given locale and display it.
* I18nSelectPipe returns a value that corresponds to a given string and shows that.
* JsonPipe lets us show JSON objects on the screen, which is helpful for debugging.
* KeyValuePipe lets us get the keys and the corresponding values in a new object so that we can get them and display them on the screen.
* LowerCasePipe returns a string converted to all lower case.
* PercentPipe converts a decimal number into a locale specific percentage value.
* SlicePipe returns a subset of the elements in an array or string.
* TitleCasePipe converts a string into title case.
* UpperCasePipe converts a string into upper case.