* **What is the difference between AngularJS and Angular?**

|  |  |
| --- | --- |
| **AngularJS** | **Angular** |
| It is based on MVC architecture | This is based on Service/Controller |
| This uses use JavaScript to build the application | Introduced the typescript to write the application |
| Based on controllers concept | This is a component based UI approach |
| Not a mobile friendly framework | Developed considering mobile platform |
| Difficulty in SEO friendly application development | Ease to create SEO friendly applications |

* **What are the key components of Angular?**

Angular has the below key components,

* **Component:** These are the basic building blocks of angular application to control HTML views.
* **Modules:** An angular module is set of angular basic building blocks like component, directives, services etc. An application is divided into logical pieces and each piece of code is called as "module" which perform a single task.
* **Templates:** This represent the views of an Angular application.
* **Services:** It is used to create components which can be shared across the entire application.
* **Metadata:** This can be used to add more data to an Angular class.
* **What are directives?**

Directives add behaviour to an existing DOM element or an existing component instance.

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

@Directive({ selector: '[myHighlight]' })

export class HighlightDirective {

constructor(el: ElementRef) {

el.nativeElement.style.backgroundColor = 'yellow';

}

}

Now this directive extends HTML element behavior with a yellow background as below

<p myHighlight>Highlight me!</p>

* **What are components?**

Components are the most basic UI building block of an Angular app which formed a tree of Angular components. These components are subset of directives. Unlike directives, components always have a template and only one component can be instantiated per an element in a template. Let's see a simple example of Angular component

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

@Component ({

selector: 'my-app',

template: ` <div>

<h1>{{title}}</h1>

<div>Learn Angular6 with examples</div>

</div> `,

})

export class AppComponent {

title: string = 'Welcome to Angular world';

}

* **What are the differences between Component and Directive?**

In a short note, A component(@component) is a directive-with-a-template.

Some of the major differences are mentioned in a tabular form

|  |  |
| --- | --- |
| **Component** | **Directive** |
| To register a component we use @Component meta-data annotation | To register directives we use @Directive meta-data annotation |
| Components are typically used to create UI widgets | Directive is used to add behavior to an existing DOM element |
| Component is used to break up the application into smaller components | Directive is use to design re-usable components |
| Only one component can be present per DOM element | Many directives can be used per DOM element |

* **What is angular CLI?**

Angular CLI(**Command Line Interface**) is a command line interface to scaffold and build angular apps using nodejs style (commonJs) modules. You need to install using below npm command,

npm install @angular/cli@latest

Below are the list of few commands, which will come handy while creating angular projects

* **Creating New Project:** ng new
* **Generating Components, Directives & Services:** ng generate/g The different types of commands would be,
* ng generate class my-new-class: add a class to your application
* ng generate component my-new-component: add a component to your application
* ng generate directive my-new-directive: add a directive to your application
* ng generate enum my-new-enum: add an enum to your application
* ng generate module my-new-module: add a module to your application
* ng generate pipe my-new-pipe: add a pipe to your application
* ng generate service my-new-service: add a service to your application
* **Running the Project:** ng serve
* **What is dependency injection in Angular?**

Dependency injection (DI), is an important application design pattern in which a class asks for dependencies from external sources rather than creating them itself. Angular comes with its own dependency injection framework for resolving dependencies( services or objects that a class needs to perform its function).So you can have your services depend on other services throughout your application.

* **What is the option to choose between inline and external template file?**

You can store your component's template in one of two places. You can define it inline using the **template** property, or you can define the template in a separate HTML file and link to it in the component metadata using the **@Component** decorator's **templateUrl** property. The choice between inline and separate HTML is a matter of taste, circumstances, and organization policy. But normally we use inline template for small portion of code and external template file for bigger views. By default, the Angular CLI generates components with a template file. But you can override that with the below command,

ng generate component hero -it

* **How do you categorize data binding types?**

Binding types can be grouped into three categories distinguished by the direction of data flow. They are listed as below,

* From the source-to-view
* From view-to-source
* View-to-source-to-view

The possible binding syntax can be tabularized as below,

|  |  |  |
| --- | --- | --- |
| **Data direction** | **Syntax** | **Type** |
| From the source-to-view(One-way) | 1. {{expression}} 2. [target]="expression" 3. bind-target="expression" | Interpolation, Property, Attribute, Class, Style |
| From view-to-source(One-way) | 1. (target)="statement" 2. on-target="statement" | Event |
| View-to-source-to-view(Two-way) | 1. [(target)]="expression" 2. bindon-target="expression" | Two-way |

* **How do you chain pipes?**

You can chain pipes together in potentially useful combinations as per the needs. Let's take a birthday property which uses date pipe(along with parameter) and uppercase pipes as below

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

@Component({

selector: 'app-birthday',

template: `<p>Birthday is {{ birthday | date:'fullDate' | uppercase}} </p>` // THURSDAY, JUNE 18, 1987

})

export class BirthdayComponent {

birthday = new Date(1987, 6, 18);

}

* **What is a bootstrapping module?**

Every application has at least one Angular module, the root module that you bootstrap to launch the application is called as bootstrapping module. It is commonly known as AppModule. The default structure of AppModule generated by AngularCLI would be as follows,

/\* JavaScript imports \*/

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

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

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

import { HttpClientModule } from '@angular/common/http';

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

/\* the AppModule class with the @NgModule decorator \*/

@NgModule({

declarations: [ //register all the components,directives,pipes

AppComponent

],

imports: [

BrowserModule,

FormsModule,

HttpClientModule

],

providers: [], // declare pipes,services

bootstrap: [AppComponent] // the main component of the application i.e entry point of the application.

})

export class AppModule { }

* **What are Observables?**

Observables are declarative which provide support for passing messages between publishers and subscribers in your application. They are mainly used for event handling, asynchronous programming, and handling multiple values. In this case, you define a function for publishing values, but it is not executed until a consumer subscribes to it. The subscribed consumer then receives notifications until the function completes, or until they unsubscribe.

* **What is the short hand notation for subscribe method?**

The subscribe() method can accept callback function definitions in line, for next, error, and complete handlers is known as short hand notation or Subscribe method with positional arguments. For example, you can define subscribe method as below,

myObservable.subscribe(

x => console.log('Observer got a next value: ' + x),

err => console.error('Observer got an error: ' + err),

() => console.log('Observer got a complete notification')

);

**\* What is dynamic components?**

Dynamic components are the components in which components location in the application is not defined at build time.i.e, They are not used in any angular template. But the component is instantiated and placed in the application at runtime.

***General Questions:***

* What did you learn about Angular yesterday/this week?
* What are some of the reasons you would choose to use Angular in your project?
* What did you like about working with Angular?
* How do you keep your Angular code more readable and maintainable?
* What does testable code mean to you in context of Angular?
* What does reusable code mean to you in context of Angular?

***Animations Questions:***

* How do you define transition between two states in Angular?
* How do you define a wildcard state?

***Architecture Questions:***

* What is a good use case for ngrx/store?
* What is a good use case for ngrx/entity?
* Can you talk about a bug related to a race condition, how to solve it and how to test it?
* What is the difference between a smart/container component and dumb/presentational component? What is a good use case example? What are the advantages?

***API Questions:***

* What does this code do:

@HostBinding('class.valid') isValid;

<div \*ngIf='someObservableData | async as data; else loading'>{{data}}</div>

<ng-template #loading>

Loading Data...

</ng-template>

* Why would you use renderer methods instead of using native element methods?
* How would you control size of an element on resize of the window in a component? Ans. Use @HostListener
* What would be a good use for NgZone service?
* What are the bootstrap options for NgZone? Why would you use them? (Angular 5+)
* How would you protect a component being activated through the router?
* How would you insert an embedded view from a prepared TemplateRef?
* What is the difference between @ViewChild() and @ContentChild()

***Template Syntax Questions:***

* How can you add an active class to a selected element in a list component?
* What is a template variable. How would you use it?
* What is the difference of using a property binding verses a function binding on a template?
* What happens if you subscribe to a data source multiple times with async pipe?
* What is the difference between ng-content, ng-container and ng- template?
* When you create a data-binding in Angular, are you working with attributes or properties? What is the difference anyway?
* When can you omit the brackets in template binding?

***Component Questions:***

* What is the minimum definition of a component?
* What is the difference between a component and a directive?
* How do components communicate with each other?
* How do you create two way data binding in Angular?
* How would you create a component to display error messages throughout your application?
* What does a lean component mean to you?

***Component Interaction & State Management Questions:***

* How would you pass data from a parent component to a child component?
* How would you pass data from a child component to a parent component?
* Which components will be notified when an event is emitted?
* Tell me about the different ways how you would get data to your components from a service and talk about why would you use one way vs the other?
* How would you use cached data?

***Forms Questions:***

* When do you use template driven vs model driven forms? Why?
* How do you submit a form?
* What's the difference between NgForm, FormGroup, and FormControl? How do they work together?
* What's the advantage of using FormBuilder?
* How do you add form validation to a form built with FormBuilder?
* What's the difference between dirty, touched, and pristine on a form element?
* How can you access validation errors in the template to display error messages?
* What is async validation and how is it done?
* What is the correct form control class name which is set to true when value is modified?

***NgModules Questions:***

* What is the purpose of NgModule?
* How do you decide to create a new NgModule?
* What are the attributes that you can define in an NgModule annotation?
* What is the difference between a module's forRoot() and forChild() methods and why do you need it?
* What is providedIn property used for in an NgModule?
* What would you have in a shared module?
* What would you not put shared module?
* What module would you put a singleton service whose instance will be shared throughout the application (e.g. ExceptionService andLoggerService)?
* What is the purpose of exports in a NgModule?
* What is the difference between exports and declarations in NgModule?
* Why is it bad if SharedModule provides a service to a lazy loaded module?

***Services Questions:***

* What is the use case of services?
* How are the services injected to your application?
* How do you unit test a service with a dependency?
* Why is it a bad idea to create a new service in a component like the one below?

let service = new DataService();

***Structural Directives Questions:***

* What is a structural directive?
* How do you identify a structural directive in html?
* When creating your own structural directives, how would you decide on hiding or removing an element? What would be the advantages or disadvantages of choosing one method rather than the other?

***Style Guide Questions:***

* What are some of the Angular Style Guide suggestions you follow on your code? Why?
* Is it important to have a style guide? Why/not?

***Styling Questions:***

* How would you select a custom component to style it.
* What pseudo-class selector targets styles in the element that hosts the component?
* How would you select all the child components' elements?
* How would you select a css class in any ancestor of the component host element, all the way up to the document root?
* What selector force a style down through the child component tree into all the child component views?
* What does :host-context() pseudo-class selector targets?
* What does the following css do?

:host-context(.theme-light) h2 {

background-color: red;

}

***Lifecycle Hooks Questions:***

* What is the possible order of lifecycle hooks.
* When will ngOnInit be called?
* How would you make use of ngOnInit()?
* What would you consider a thing you should be careful doing on ngOnInit()?
* What is the difference between ngOnInit() and constructor() of a component?
* What is a good use case for ngOnChanges()?

***Observables RxJS Questions:***

* What is the difference between an observable and a promise?
* What is the difference between an observable and a subject?
* What are some of the angular apis that are using observables?
* How would you cache an observable data?
* How would you implement a multiple api calls that needs to happen in order using rxjs?
* What is the difference between switchMap, concatMap and mergeMap?
* How would you make sure an api call that needs to be called only once but with multiple conditions. Example: if you need to get some data in multiple routes but, once you get it, you can reuse it in the routes that needs it, therefor no need to make another call to your backend apis.
* How would you implement a [brush HYPERLINK "https://bl.ocks.org/mbostock/34f08d5e11952a80609169b7917d4172"behavior](https://bl.ocks.org/mbostock/34f08d5e11952a80609169b7917d4172) using rxjs?
* How would you implement a color picker with rxjs?
* If you need to respond to two different Observable/Subject with one callback function, how would you do it?(ex: if you need to change the url through route parameters and with prev/next buttons).
* What is the difference between scan() vs reduce() ?

***Performance Questions:***

* What are some of the things that you pay attention to, to make sure your angular application is performant?
* What tools would you use to find a performance issue in your code?
* What tools have you used to improve the performance of your application?
* What are some ways you may improve your website's scrolling performance?
* Explain the difference between layout, painting and compositing.
* Have you seen Jeff Cross's NgCruise talk on performance?

***Pipes Questions:***

* What is a pure pipe?
* What is an async pipe?
* What kind of data can be used with async pipe?
* How do you create a custom pipe?
* How does async pipe prevents memory leeks?
* What is the difference between pure and impure pipes?

***Router Questions:***

* What is the difference between RouterModule.forRoot() vs RouterModule.forChild()? Why is it important?
* How does loadChildren property work?
* Do you need a Routing Module? Why/not?
* When does a lazy loaded module is loaded?
* Below link doesn't work. Why? How do I fix it?

<div routerLink='product.id'></div>

* Can you explain the difference between ActivatedRoute and RouterState?
* How do you debug router?
* Why do we need route guards?
* What is a RouterOutlet?

***Security Questions:***

***Testing Questions:***

* What are some of the different tests types you can write?
* How do you mock a service to inject in an integration test?
* How do you mock a module in an integration test?
* How do you test a component that has a dependency to an async service?
* What is the difference between 'async()' and 'fakeAsync()'?

***TypeScript Questions:***

* Why do you need type definitions?
* How would you define a custom type?
* What is the difference between an Interface and a Class?
* First line below gives compile error, second line doesn't. Why?

someService.someMethod(x);

someService['someMethod'](x);

* What are Discriminated union types?
* How do you define Object of Objects type in typescript?
* How can you capture the 'type' the user provides (e.g. number), so that we can use that information later.

***JavaScript Questions:***

* Explain the difference between var, let and const key words.
* Could you make sure a const value is garbage collected?
* Explain Object.assign and possible use cases.
* Explain Object.freeze and possible use cases.
* Explain the code below. How many times the createVal function is called?

function createVal(){

return Math.random();

};

function fun( val = createVal()){

// Do something with val...

}

fun();

fun(5);

* What is the spread operator doing in this function call? Seriously!

doStuff(...args);

* What is destructuring assignment?
* Explain why the below stand-alone syntax is not valid?

{a, b} = {a: 1, b: 2}

***Coding Questions:***

* What would these components render?

import { Component, ContentChildren, Directive, Input, QueryList } from '@angular/core';

@Directive({selector: 'pane'})

export class Pane {

@Input() id: string;

}

@Component({

selector: 'tab',

template: `

<div>panes: {{serializedPanes}}</div>

`

})

export class Tab {

@ContentChildren(Pane) panes: QueryList<Pane>;

get serializedPanes(): string { return this.panes ? this.panes.map(p => p.id).join(', ') : ''; }

}

@Component({

selector: 'example-app',

template: `

<tab>

<pane id="1"></pane>

<pane id="2"></pane>

<pane id="3" \*ngIf="shouldShow"></pane>

</tab>

<button (click)="show()">Show 3</button>

`,

})

export class ContentChildrenComp {

shouldShow = false;

show() { this.shouldShow = true; }

}

* What's a cool project that you've recently worked on?
* What are some things you like about the developer tools you use?
* Who inspires you in the angular community?
* Do you have any pet projects? What kind?
* How did you design the architecture of your project?
* What's your favorite feature of Angular?
* If you could decide on a new feature for angular, what would it be?

**RealTIme Development FAQs**

* **What is the use of EventEmitter**.

**EventEmitter is used in Angular to create custom events that allow a child**

**component to send data or signals to its parent component.**

* **How to open one component as a popup of current component.**

**constructor(public modalService: BsModalService){}  
 public modalRef?: BsModalRef;**

**this.modelref = this.modalService.show(StatusComponent, {**

**class: 'fullwidth-dialog',**

**backdrop: 'static',**

**keyboard: false,**

**initialState: { // data to be sent from current component to status**

**component.**

**applicationid: this.applicationIds,**

**data: this.data,**

**flag: 'editInquiry',**

**webFlag: 'InqSave',**

**Type: this.data.type,**

**salesUserFlag:true**

**}**

**})**

* **What is JSON.Stringify() method.**  
  **This method is used to convert Javascript object value to JSON string.**
* **In Realtime why JSON.stringify() is not used for POST Http method type**.

**Because in case of GET type Http method then it is JS/typescript object format, so we need to convert it as a JSON string, But in case of POST/save it is already stringify(JSON string), so don’t need to use JSON.stringify() method. If we will use then it will work in postman only but not inside the project.**

**In case of POST they handle the JSON serialization automatically.**

* **What do you mean by --save at the time of installing npm or other packages.**

**--save means the package will be added as a dependency of your project.**

* **How to pass query params from one component to another and how to access it inside the target component.**

**Source Component:**

***This.router.navigate([‘maintenance/security-roles’],{***

***queryParams: {***

***id: this.id,***

***roleName: ‘TestRole23’,***

***sortOrder: this.sortorder***

***}***

**Target Component:**

**Way:1:**

***constructor(private route:ActivatedRoute){}***

***let id = this.router.snapshot.queryParamMap.get(‘id’)***

***let roleName = this.router.snapshot.queryParamMap.get(‘roleName’)***

***let sortOrder = this.router.snapshot.queryParamMap.get(‘sortOrder’)***

**Way:2:**

***this.router.queryParams.subscribe(res=>{***

***const id = res[‘id];***

**const *roleName* = res[‘*roleName’*];**

**const *sortOrder* = res[‘*sortOrder’*];**

**1. What is pathMatch='full' and pathMatch='prefix'.**

**Ans. pathMatch='full' it means the exact match of an url it is used with redirect.**

**Ex: {path: '', redirectTo: "/login", pathMatch: 'full' },**

**pathMatch='exact' it means the partial match of an URL, default behaviour.**

**2. What is loadChildren.**

**Ans. It handles the lazy loading for a route i.e dynamically load a module when the user visits the page. It improves the performance of an application throgh reducing the initial bundle size.**

**Ex: {path : "sales",canLoad: [AuthGuard] ,loadChildren: ()=>(import('./maintenance/maintenance.module')).then(m=>MaintenanceModule)},**

**3. What is the difference between href and routerLink.**

**routerLink is an angular specific directive where as href is a html attribute.**

**routerLink is handled by Angular router where as href is handled bw browser.**

**routerLink maintains SPA where as href does not, it entired load the page.**

**4. What is Route Guard.**

**It is used to prevent the unauthorized users to access the particular routes.**

**Here is several route guards,  
i. CanActivate() : This guard checks the login before the route loads i.e checks if the application is login then dashboard component wil be loaded.**

**Ex: { path: '\*\*', canActivate: [AuthGuard], redirectTo: '' }**

**ii. CanActivateChild() : This guard checks the login before the child component is loaded. applying a guard logic to all child routes.  
Ex: {**

**path: 'admin',**

**component: AdminLayoutComponent,**

**canActivateChild: [AuthGuard],**

**children: [**

**{ path: 'users', component: UserListComponent },**

**{ path: 'settings', component: SettingsComponent }**

**]**

**}**

**iii. canDeactivate(): It checks if the user leaves from the current route component.**

**iv. canLoad(): This guard prevents the lazy loading module after the login of the application, when the corresponding route will be accessed it will load at that time.**

**5. What is the difference between canActivate and CanLoad.**

**canActivate prevents the navigation where as canLoad controls the loading of modules.**

**6. What is Safe navigation and nullish coalescing operator.**

**?. is known as safe navigation operator which avoids the error if the value if null or undefined.**

**?? is known as nullish coalescing operator which provides the default value if the left side operand is null or undefined.**