Angular Interview Questions Ebook

By

www.questpond.com

Contents

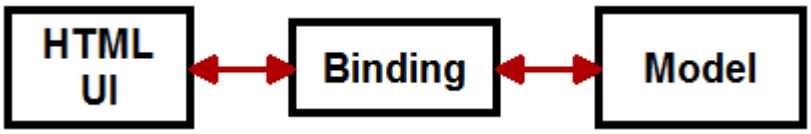
1. What is the whole goal of Angular ?

Note :- Now lot of candidates answer this question by saying it’s for SPA , or some say its a Javascript UI framework and so on. I would suggest to answer in a more detailed way and in a more holistic way so that you send a message to the interviewer that you really know angular and you are the right candidate.

Angular is a JavaScript Binding framework which binds the HTML UI and Javascript Model. This helps you to reduce your effort on writing those lengthy lines of code for binding.

Adding to it,it also helps you to build SPA by using the concept of routing.
It also has lot of other features like HTTP , DI , Input output because of which you do not need other frameworks.

So when you are answering question please answer in a more detailed way and not just one liners.



2. What are directives in Angular and how many types of directives exists?

Directives help you to attach behaviour in the HTML DOM. For example in the below code "ngModel" is a directive which when attached to the textbox binds the text box value to "myvariable".

```
<input [(ngModel)]="myvariable" type="text" value=""><br>
```

There are three types of directives in Angular :-

- Structural directives: - Change the DOM layout by adding and removing elements.

  For example in the below code we have the "*ngFor" loop which is a example of structural directive. How many HTML table "tr" will be added in the HTML DOM layout depends on rows in the "coll" collection.

```
 <tr *ngFor="let temp of coll">
        <td>{{temp.CustomerName}}</td>
  </tr>
```

- Attribute directives: - Change the appearance and behaviour of HTML elements.

```
<div  [hidden]="Hide()">Some text</div>
```

- Component directives: - Directives with templates. It's like a user control.

```
<my-grid [grid-data]="SalesModelObjs"
</my-grid>
```

3. Explain data bindings and different types of the same?

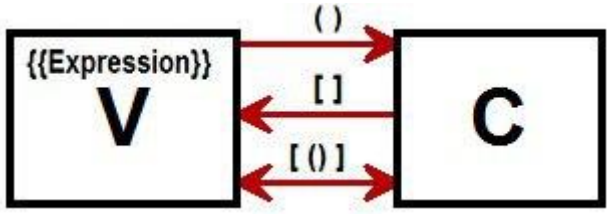
Data binding defines how the view and component communicate with each other. There are four types of bindings in Angular as shown below.

Expression / Interpolation {{}}: - Data flows from component to the view and we can mix the same with HTML tags.

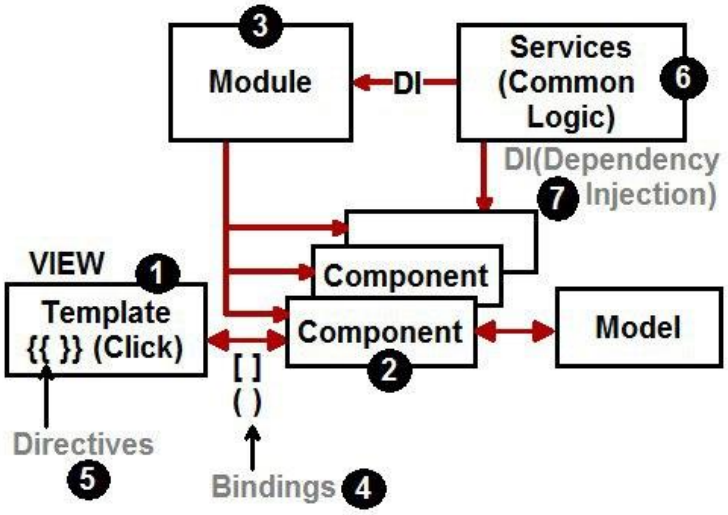
Property binding []: - Data flows from component to the view.

Event Binding (): - When you want to send event from the view to the component.

Two-way binding [()] :- Data flows from component to the view and vice versa.



4. Explain the basic components involved in Angular ?



There are 7 important pillars in Angular Architecture.

1. Template:- The HTML view of Angular.
2. Component:- Binds the View and Model.
3. Modules:- Groups components logically.
4. Bindings :- Defines how view and component communicate.
5. Directive :- Changes the HTML DOM behaviour.
6. Services :- Helps to share common logic across the project.
7. DI :- Dependency injection helps to inject instance across constructor.

5. Difference between AngularJS vs Angular?

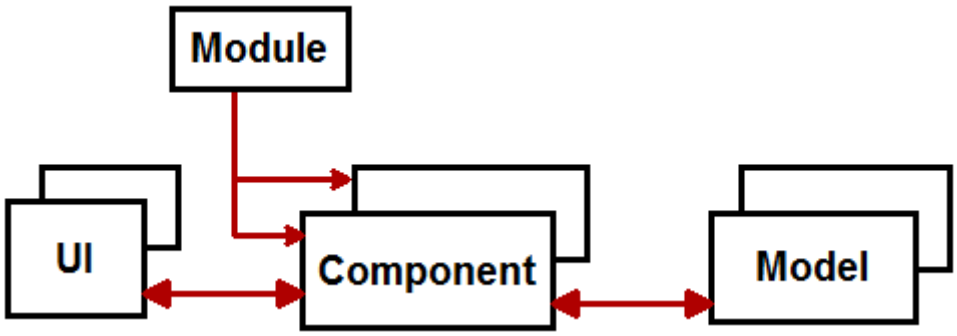
	AngularJS (1.x)	**Angular** 2 , 4 , 5 , 6 , 7 ,8 and 9
Language	JavaScript	TypeScript
Architecture	Controller	Component
Mobile compliant	No	Yes
CLI	No	Yes
Lazy loading	No	Yes
SEO	No	Yes
Server side	No	Yes

6. What are component and modules in Angular?

As said in one of the previous question angular is a binding framework it binds the UI and model.

Component is where you write your binding code. Component Binds the UI and Model.

Module groups components. In a big project we can have lot of UI, Component and Models, you can logically group them by using Modules.



7. What are decorators in Angular ?

Decorator are meta-data which is applied / decorated on the class level as shown below. If you decorate "@Component" on the class the class is treated as Angular component.

```
@Component()
export class AppComponent {}
```

If you decorate "@NgModule" on the class it becomes a Angular Module.

```
@NgModule({})
export class MyModule { }
```

8. What is metadata or annotations in Angular ?

Meta-data or annotations are also termed as decorators which is already answered in the previous question.

9. What are templates in Angular ?

An Angular template is an HTML view where you can use Angular Directives. You can write template in two ways one is **inline templates** or you can have **separate HTML file** and link the same.

Below is a simple code of inline template where in the HTML is written in the code itself.

```
@Component({
template: '<h1>Test</h1>'
})
export class AppComponent {}
```

Below is code where we have template as separate HTML. To define separate template we need to use "templateUrl".

```
@Component({
templateUrl: './app.component.html'
})
export class AppComponent {}
```

10. What is SPA and how to implement in Angular ?

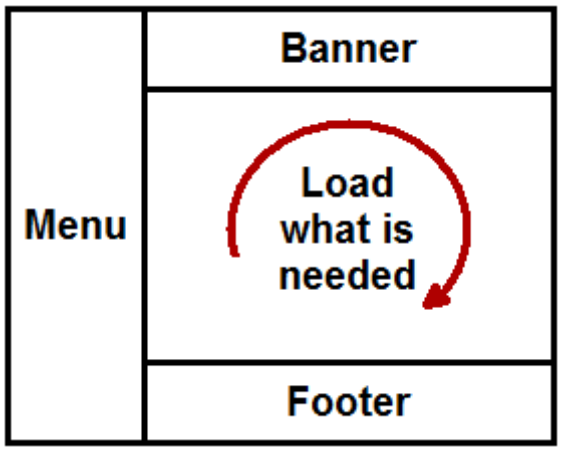
SPA stands for Single page application. Single page application are applications where the main UI gets loaded once and then the needed UI or data is loaded rather than making a full post back.

For example, in a typical website we have master pages some even term them as template.

So, a typical master page has left menu, footer, top banner and so on. So, the master page gets loaded once and then on demand, depending on which link the user clicks other pages are loaded, the main master page is not loaded again and again.

In Angular it is implemented by using lazy routing concept.

SPA or single page application are those applications where the main page is loaded once and the other pages get loaded on demand.



11. Explain the importance of routing in Angular & how to implement?

Routing helps you to define the navigation for your angular application. So if you want to move from one screen to other screen and you want to respect SPA that means not loading and refreshing the whole UI routing is needed.

In order to implement routing we need to use the routing module of angular. So we need to do the following steps :-

1. Create routing collection which defines the URL path and which component to load for that URL.

```
export const PatientRoutes = [
    { path: '', component: PatientHomeComponent},
    { path: 'Login', component: PatientLoginPageComponent}
];
```

2. Later this routing collection is loaded using router module using “forroot” or “forchild” as shown in the code below.

```
RouterModule.forRoot(PatientRoutes),
```

3. Then we can use router-link directive on html anchor tag or in TS we can navigate using router.navigate.

```
<a [routerLink]="['/Home']">Home</a><br>
<a [routerLink]="['/Login']">Login</a><br>
```

When the navigation happens, the UI gets loaded in to router-outlet tag. Router-outlet is a reserved tag in which the UI loads when routing happens.

```
<router-outlet>
</router-outlet>
```

12. What is Lazy loading concept in Angular ?

Lazy loading means on demand loading. Loading only the necessary HTML , CSS and JavaScript files so that you have better performance. So for example let’s say you have a

hospital management system , so when the front desk user logs in we just load appointment module and billing module, when doctors logs in we load modules like treatment , operation theatre and so on.

So with lazy loading we have better UI experience and performance because we load only the necessary HTML , JS and CSS files thats is needed by the end user.

13. How to implement lazy loading in Angular?

In order to implement lazy loading we need to do three steps: -

1. The first thing is dividing your project in to modules.
2. Create separate routing files for each module and use “loadChildren” and specify which modules to be loaded on demand.
3. In routing navigation use “forRoot” to load the routes of the main module and “forChild” to load the child modules.

14. What is node ?

Its a javascript runtime which helps to run Javascript outside the browser . It does that by using V8 chrome engine . Chrome V8 engine compiles javascript fully rather than interepting it.

15. What is NPM ?

Its a package manager which makes installation of Javascript framework easy.

16. What is the importance of node_modules folder ?

“node_modules” is the folder where all the packages are installed.

17. What is package.json ?

It has all the Javascript references needed for a project. So rather than installing one package at a time we can install all packages in one go.

What is the importance of ^ and ~ in package.json?

^=>> Latest Minor version+ revision
~ ==> Latest revision

What is package.lock ?
It has the exact version which is installed after analysis of package.json

Explain the Flow of Angular ?
Index → Main → Module → Coponent ==> HTML
pushed the HTML in the selector of Index

What is the use of templateURL ?

TemplateURL connect the component with the UI

18. What is typescript?

Typescript superset of JavaScript. It added types to JavaScript.
It gives a nice Object-oriented programming environment which transpiles / converts to JavaScript.
So as its strongly typed we will have less errors and because we can do OOP with JavaScript our productivity and quality also increases.

19. What is the need of Angular CLI ?

Angular CLI is a command line interface by which we can create initial Angular project template. So rather than starting from scratch we have some boiler plate code.

20. What are services in Angular ?

Services helps you to share common logic across Angular projects.

Dependency Injection is an application design pattern where rather than creating object instances from within the component , Angular injects it via the constructor.

By using the "@NgModule" provider meta-data we can specify which instance to be injected

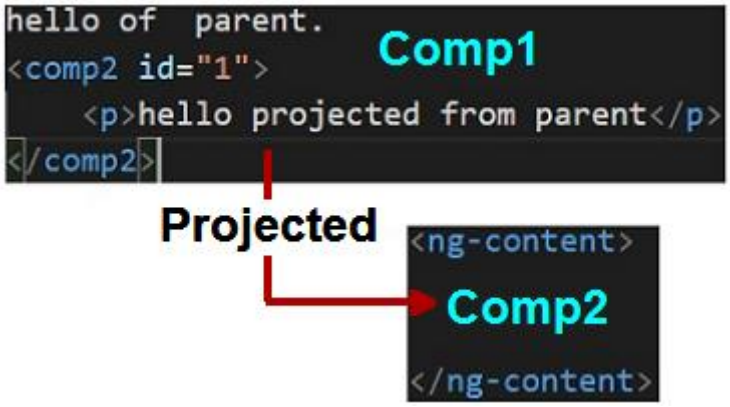
Dependency injection helps to decouple class dependencies , so that when you add new dependencies you do not have change everywhere.

"ng serve" builds inmemory while "ng build" builds on the hard disk. So when you want to go for production "ng build" command is used.

Ng build –prod flag compresses your JS file , removes comments , creates GUIDs of your JS files and make your application ready for production.

21. In What scenarios will we use content projection ?

Content Projection is used when we want to project contents like HTML or Components from parent component to child component. These contents get projected in to a reserved tag by name "ng-content".

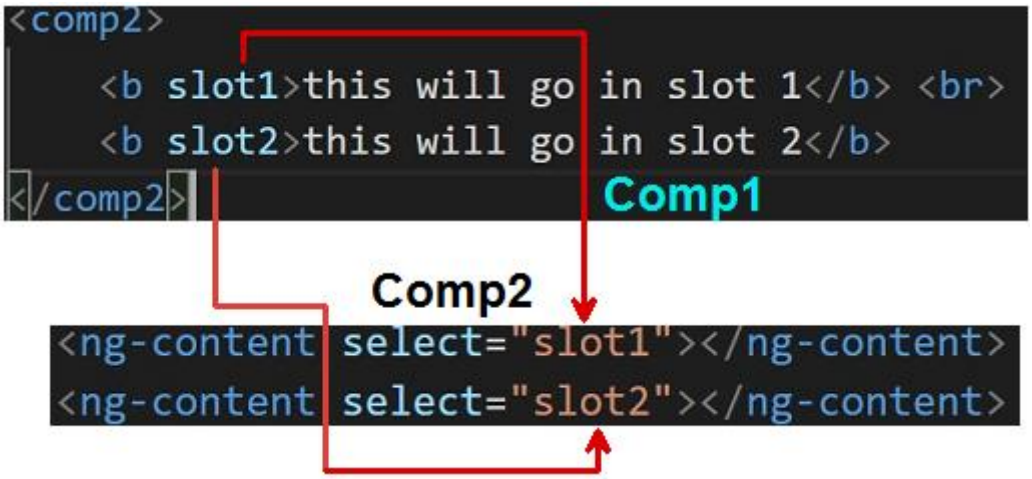


22. Explain Content Projection Slots in Angular?

Content projection slot helps to project specific content to specific "<ng-content>" tag section..So if we have two “ng-content” sections in the contained component and from the parent component we want to project specific content to specific “ng-content” tags then we need to define Content projection slots.

Defining content projection slot is a two-step process: -

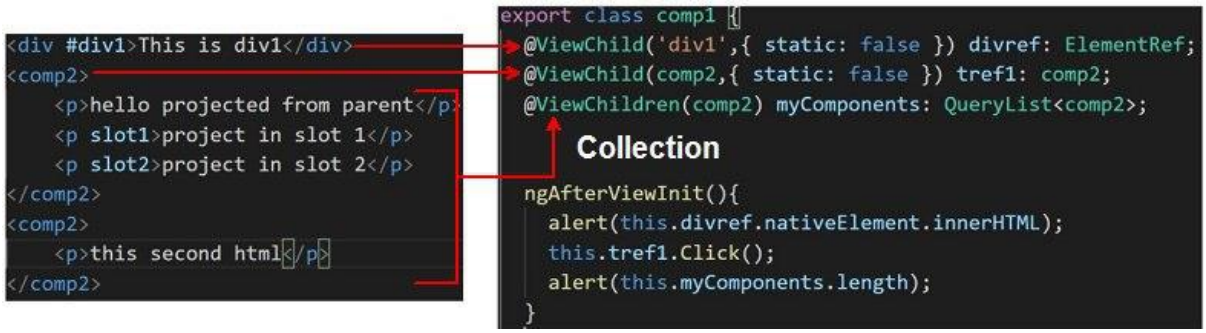
- In the contained component we will use the “select” attribute to define the slot name in “ng-content” tag.
- Then in the parent component we will provide the slot name to project in the respective “ng-content” tags. Below is a pictorial representation of the same.



23. Why do we need “ViewChild” and “ViewChildren” in Angular?

Angular have two parts one is the view and the other is component or the code which handles the view data and events. Now in the component many times we would like to refer instance of view elements , that’s where “ViewChild” helps.

“ViewChild” helps to reference view objects in the component to which it is connected.“ViewChild” references one object while “ViewChildren” references collection.



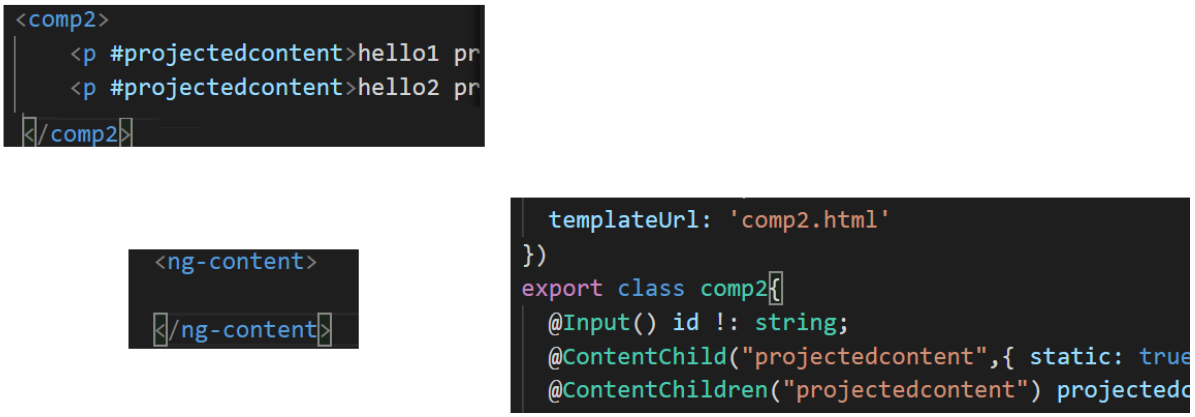
24. What’s Template reference variable?

A template reference variable is used to give reference to a DOM Element, a component, directive, or a web component within a template. This variable can then be used anywhere inside the template or inside the component to reference the same.

<< write more>>

25. Explain “ContentChild” and “ContentChildren” ?

“ContentChild” and “ContentChildren” helps to access projected contents from the parent component.
“ContentChild” references a single projected content while “ContentChildren” references collection.



26. Differentiate between ViewChild , ViewChildren , ContentChild and ContentChildren ?

“ViewChild” and “ViewChildren” helps to reference view elements which belongs to HIS OWN VIEWS.

“ContentChild” and “ContentChildren” helps to access view elements which is PROJECTED BY THE PARENT.

```
<comp2>
    <p #projectedcontent>hello1 pr
    <p #projectedcontent>hello2 pr
</comp2>
```

```
export class comp1 {
  @ViewChild(comp2,{ static: false }) tref1: comp2;
  @ViewChildren(comp2) myComponents: QueryList<comp2>;
```

```
<ng-content>

</ng-content>
```

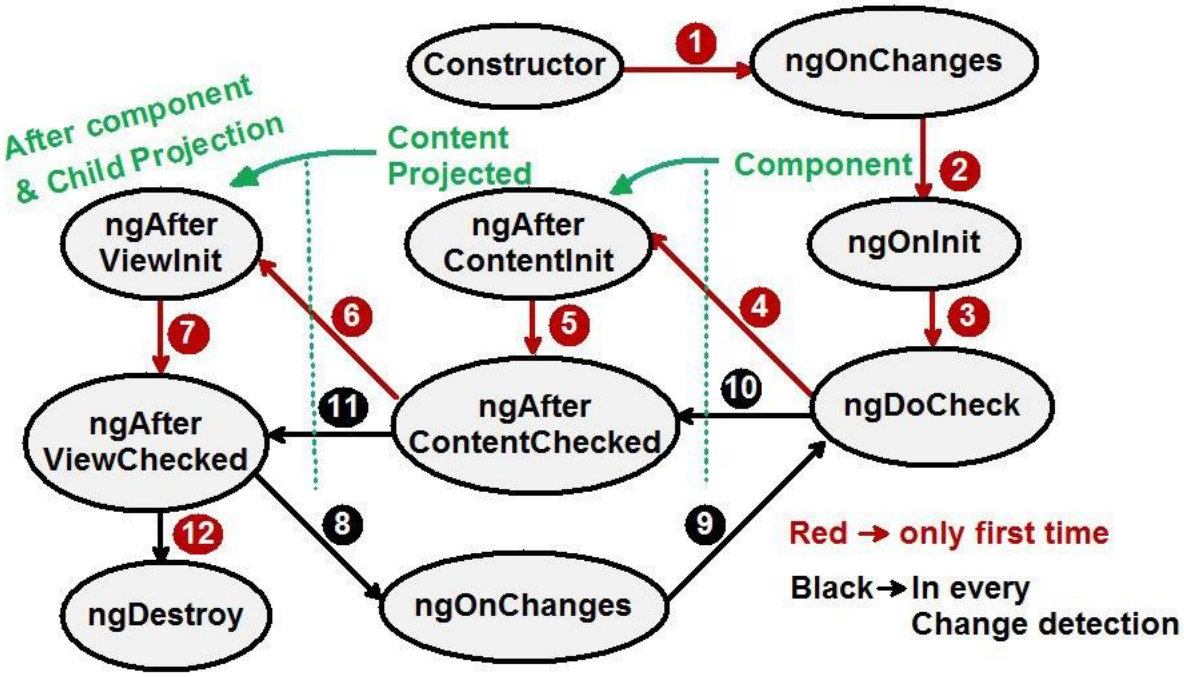
```
  templateUrl: 'comp2.html'
})
export class comp2{
  @Input() id !: string;
  @ContentChild("projectedcontent",{ static: true
  @ContentChildren("projectedcontent") projectedc
```

27. What is { static: true } in ViewChild ?

28. What’s the importance Angular component hooks / life cycle?

Component life cycle a.k.a angular component hooks are events which developers can tap in and write custom logic in the same. For instance if you want to write initialization code when the component starts first time then we can tap in to the “ngOninit()” function . If you want to write clean up code when the component is unloading then we have the “ngDestroy()” function .

29. Explain in detail Angular life cycle hooks?



The complete Angular life cycle events are divided in to two parts: -

- Sequence of events which occurs when the component is loaded first time.
- Second sequence of events which fires on every change detection which occurs on the component.

The top image shows the first-time sequence in RED color and change detection sequence in BLACK color.

Below is the First time Sequence of events which fires when the component is loaded first time.

1. Constructor: - This is not really an event of angular it's an event of typescript class. When you create a object of a typescript class constructor fires first. And this will fire irrespective we have angular or not. But still it does have lot of significance as it comes as the first event before any angular component event fires.

2. ngOnChanges: Called when data bound input property changes. This event is called before ngOnInit().

3. ngOnInit: Called when first time the data-bound properties are displayed and here we set the @input property values.

4. ngDoCheck: Called whenever angular change detection runs.

5. ngAfterContentInit: Called after Angular projects external content first into the component's view.

6. ngAfterContentChecked: After Angular checks the content projected into the component. This is the change detection check for the contents projected.

7. ngAfterViewInit: After Angular initializes the component's views , child views and projected content this event fires.

8. ngAfterViewChecked: Once the default change detection run and content projected change detection run this event fires.

Once the component is initialized in every change detection below is how the events will fire. If there are no changed to input “npOnChanges” event will fire.

9. ngOnChanges.
10. ngDoCheck.
11. ngAfterContentChecked.
12. ngAfterViewChecked.

Finally when the component is unloaded Destroy event fires.

1. ngOnDestroy: This is the clean-up phase just before Angular destroys the directive/component.

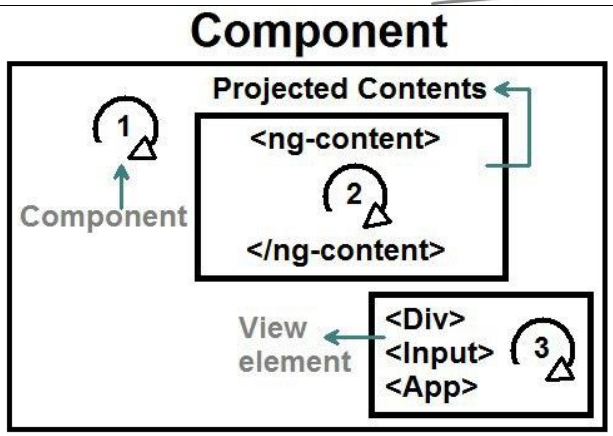
While answering this question many candidates try to mug up or byheart the answer which can make the interview suspicious. So rather than by hearting if you can understand the overall sequence of how events fire you will not forget the same.

If you see a typical view of Angular it has two parts one is its own HTML/ Content view and the other is the projected HTML / Content inside ng-content. Now whenever events fire they first fire on the components projected contents and then they fire on the components view.

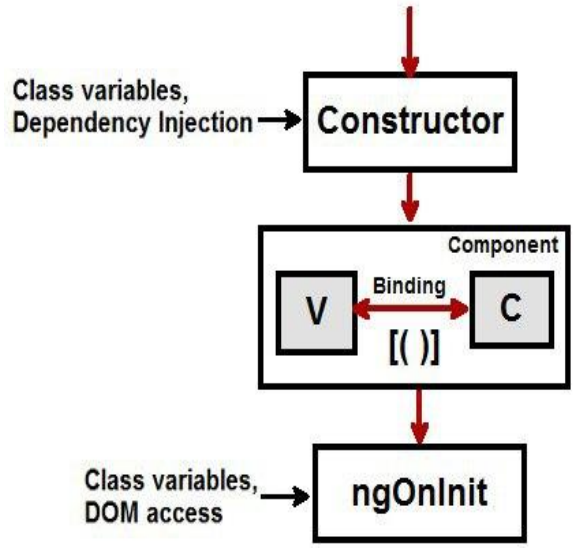
Also there is one event which fires before them which is like a **kick start event** to say that both the events should start processing.

So, there are basically three event sequence: -

- Kick start event :- ngOnInit , ngDoCheck
- After that events fire during content projection ngAfterContentInit , ngAfterContentChecked
- And then on the final view components ngAfterViewInit ,ngAfterViewChecked

So when the first time component is loaded sequence is ngOnInit , ngDoCheck , ngAfterContentInit , ngAfterContentChecked, ngAfterViewInit and ngAfterViewChecked. When component is running change detection fires for every change and the sequence is ngDoCheck , ngAfterContentChecked & ngAfterViewChecked.	

30. Differentiate between “constructor” and “ngOnInit()” ?

	“Constructor” is a concept or event of typescript class while “ngOnInit()” is a concept of angular. “Constructor” can be used to initialize class variables and do dependency injection. It should NOT be used to access any UI elements /DOM or Angular objects because they will not be available during this event. “ngOnInit()” event fires after your UI is binded with the component and at this stage the component is initialized , so you have access to DOM elements and you can write any logic of initialization.

	Constructor	**ngOnInit**
Concept of	It’s a concept of Typescript. It’s called by JavaScript framework.	It’s a concept of Angular. It’s an angular event. It’s called by Angular framework.
Binding state	Binding has not happened till this moment so only class variables are accessible.	Binding with the UI has been done so the class variables have values set from the UI.
DOM	Component is not initialized. So, DOM is not accessible.	Component is initialized, DOM is ready and accessible.

Initialization logic	Initialization of class Variables and dependency injection logic can be put here.	All type of Initialization logics can be written here as right from class to component is initialized.

31. How to implement lazy loading in Angular?

32. How to implement HTTP in Angular ?

- Import “HttpClientModule” at Module level.
- Import “HttpClient” from “@angular/common/http” at component level.
- Create object of “HttpClient” using Dependency injection.
- You can then make “post” , “get” calls using “HttpClient”.
- Using subscribe function to catch success and error response.

33. How to pass data between components?

This question is a very common question and there are many ways of passing data between components. Depending on scenarios each one of these methods / ways should be used. So when you answer this question in the interview must be you can also emphasize best practices.

These are the kind of questions where in you get an opportunity to prove that not only you know angular but you also know the best practices when to use what.

Scenario	Method
Parent child	Input, Output & Event emitters. ViewChild.
Navigating	Routing
Global data	Services
	Local storage, Temp storage

34. What are pipes ?

Pipes help you transform data on a Angular UI expression from one format to some other format. For example

35. Can you give some examples of inbuilt Angular pipes ?

AsyncPipe :-Used to read the object from an asynchronous source

CurrencyPipe:-Used to format the currencies

DatePipe:-Used to format the dates

DecimalPipe:- Used to transform the decimal numbers

I18nPluralPipe:- Converts value to string that pluralizes value according to locale.

I18nSelectPipe:- Used to display values according to the selection criteria

JsonPipe:- Converts an object into a JSON string

KeyValuePipe:- Converts an Object or Map into an array of key value pairs.

LowerCasePipe:- Converts a string or text to lowercase

PercentPipe:-Used to display percentage numbers

SlicePipe:-Used to slice an array

TitleCasePipe:-Converts a string or text to title case

UpperCasePipe:-Converts a string or text to uppercase

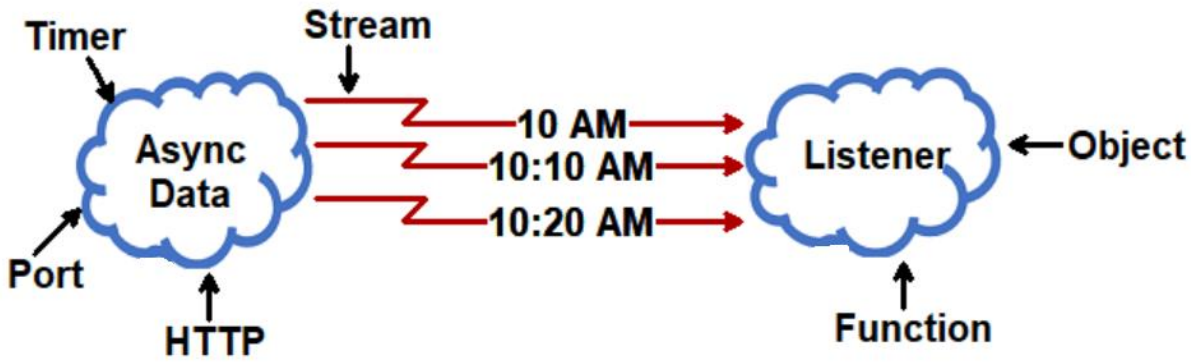
36. How can we write a custom pipe ?

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'square'
})
export class SquarePipe implements PipeTransform {

  transform(value: number): number {
    return value * value;
  }

}
```

37. What is RxJs and why do we need it ?

RxJs stands for Reactive extensions for JavaScript. RxJs helps to **handle asynchronous data stream with ease**.

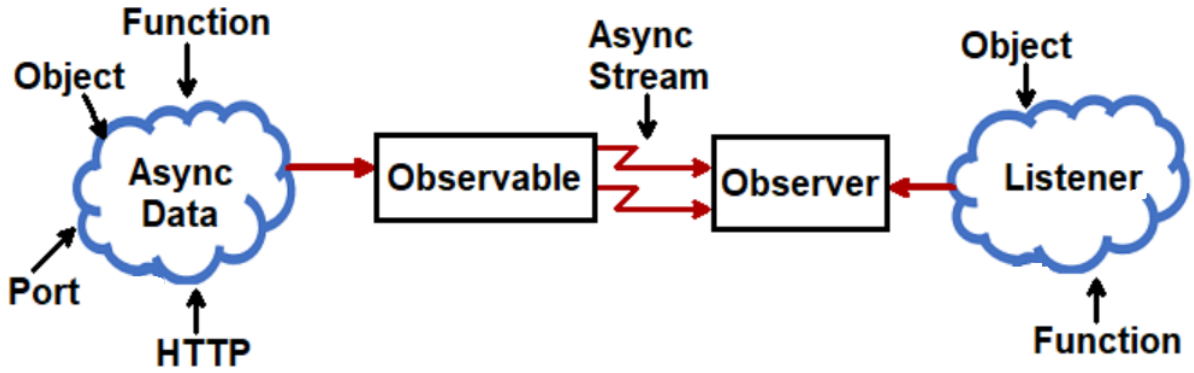
Assume you have a entity which is streaming async data, now this entity which is streaming async data can be HTTP response , Port which is streaming data , Timer emitting data and so on. This Async stream is coming in undecided intervals like stream of data. As a developer you would like to listen to this stream , run some logic on these stream and so on.RxJs makes this task easy.



38. What are observables and observers?

RxJs library helps to handle async data stream easily. But in order to access the async stream it has to be exposed as a rxJs **Observable** object. The listener who is interested in accessing the Observable stream is exposed as an **observer**.

In simple word observable represents async stream of data and observer subscribes to the observable to receive the stream.



39. What is stream in RxJs ?

Stream in RxJs is asynchronous data emitted from observables.

40. What is the use of subscribe in RxJs ?

“Subscribe” function starts the stream from observable to the observer function. You can see in the below code how the subscribe function takes a function as reference. So when data is streamed from Observable its received by the Listener function.

```
var observ = Observable.create(AsyncStream);
observ.subscribe(res=>Listener(res));


function Listener(res){
    console.log(res);
}
```

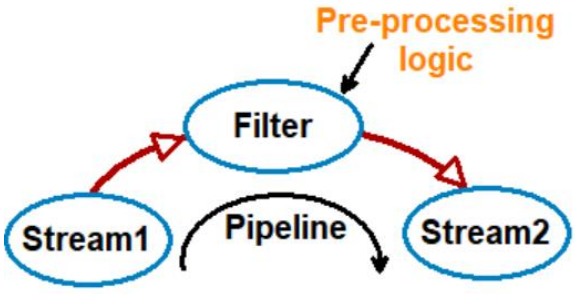
41. How to unsubscribe from the stream ?

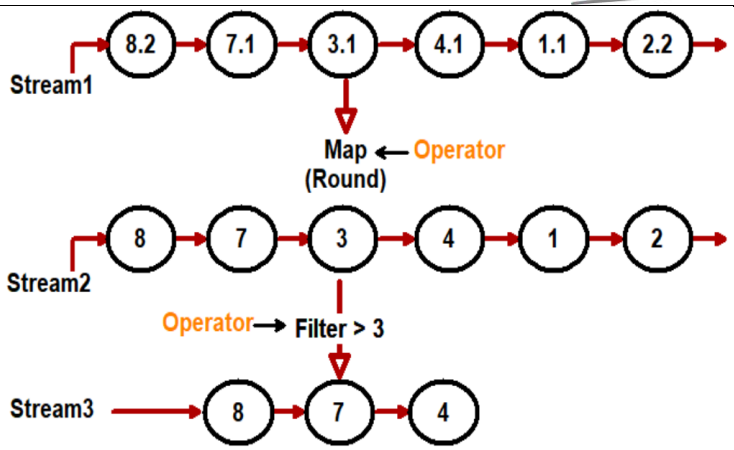
We need to get reference of the “subscription” object. This subscription object is returned when we call the “subscribe” function. To unsubscribe we can call “unsubscribe” on the “sunscription” object.

```
var subscription = observ.subscribe(res=>Listener(res));
subscription.unsubscribe();
```

42. What are operators in RxJs?

Operators are logics which manipulate an observable stream and create new observable streams.



For example you can have a stream which emits decimal numbers. You can then apply a Map operator and round the stream, which generates a new stream without decimals. You can again apply filter operator on the stream and say give values which are only greater than 3.	

Below is how the source code looks like. We have stream1 on which rounding operator is applied and stream2 is created and filter applied to create stream3.

```
var stream1 = Observable.create(AsyncStream);
var stream2 = stream1.pipe(map(x=>Math.round(x)));
var stream3 = stream2.pipe(filter(x=>x>3));
stream3.subscribe(res=>Listener(res));

function Listener(res){
    console.log(res);

}
```

43. Where did you use RxJs in Angular ?

Most of the times rxJs is used in http calls with angular. As http streams asynchronous data we can subscribe , apply filters to the http streams.

Below is a simple sample code of how RxJs can be used with HTTP calls.

```
var  stream1 = httpc.get("https://www.myapi.com/somedata");
var stream2 = stream1.pipe(filter(x=>x>3));
stream2.subscribe(res=>this.Success(res),res=>this.Error(res));
```

44. Differentiate between RxJs and Promises?

RxJs	Promise
Observable return stream of data.	Promise return single value.
You can subscribe and unsubscribe stream.	You cannot cancel a promise.

45. How to install rxJs ?

npm install rxjs

46. Why is rxjs called Push/Reactive not pull/imperative ?

Imperative programming means listener code is responsible to pull stream of data. Reactive programming means you register a callback and the stream is responsible to push data. Some devs also visualize it as publisher and subscriber model as well.

47. Name some rxJs Operators ?

- Map :- Transforms data in a observable in to a different format.
- Filter :- Allows data which meets conditions.
- Merge :- This operator will combine multiple Observables into one. So if one of the observables emit a value the combined one will emit as well.
- Concat :- only when observable completes, it will start with the next observable.
- From :- This operator will turn array, promise or iterable into an observable.
- debouncetime :- discard emitted values if a certain time didn't pass between the last input
- distinctuntilchanged :- only emits a value if it is different than the last one.
- pluck :- select a property to emit.
- delay :- emits a value with a delay.

48. What are interceptors in Angular?

Interceptors help to execute pre-processing logic before any HTTP call is made from angular application. So, for example you when any HTTP call is made any application you want to log it, you can create an interceptor and write logging logic in the same.

49. How to implement interceptors?

Implementing interceptor is a two-step process: -

Step 1 :- First create the class which has the pre-processing logic and this class should implement "HttpInterceptor" interface. In the "intercept" method we need to write pre-processing logic which will fire before HTTP call is executed.

```
export class JwtInterceptor implements HttpInterceptor {
    constructor(private tokenObj: Token) { }

intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<
any>> {
        // pre processing logic written here
            });
        return next.handle(request);
    }
}
```

Step 2 :- Dependency inject the pre-processing class at module level using providers as shown in the below code.

```
providers: [
    { provide: HTTP_INTERCEPTORS, useClass: JwtInterceptor , multi:true  }
  ]
```

50. Give some use of Interceptors?

Below are some uses of interceptors: -

Authentication: - In every call you would want to attach JWT token. This can be done by attaching token in the headers before call is made.

Logging: - Would like to log / audit calls before every HTTP call is made.

Caching: - Take data from cache rather than going to server. So before HTTP call is executed you can look up in the local cache, get data and cancel expensive HTTP call to server.

Fake backend: - If you want to fake HTTP for unit testing.

Modifying headers: - Before executing any HTTP call you would like to

51. Can we provide multi-interceptors?

Yes,you can fire multiple interceptor in pre-processing by defining multiple interceptors in the providers and by specifying “multi” value to “true”.

As per the sequence provided in the providers interceptor will fire in that sequence.

```
providers: [
   { provide: HTTP_INTERCEPTORS, useClass: JwtInterceptor , multi:true  },
   { provide: HTTP_INTERCEPTORS, useClass: JwtInterceptor1 , multi:true  }
 ]
```

52. Notes by author for further book expansion

URL structure → map to Component , define navigation for your project
Creation collection of routing
ROuter-Link and Routeroutlet for navigation and loading

validation :- formsgroup → formcontrol → validations
Valid,HasError,dirty,pristine , formsgroup.controls[name].haserror('required')

Input output event emiiter:- Resuable component which can communicate with parent component @input("grid-data") ,

Depedency injection :- Change at one place—— it should reflect in many places..Good for decoupling purpose

providers: [

{

provide: Logger,

useClass: FileLogger

}

pipes :- Transform data on UI some other format , Create custom pipes

jquery :-

npm install jquery

npm install @types/jquery

typings t.ds , the method names of all the framework
import * as $ from "…"

ViewChild -> Component to component communication

Caching , Constant :- DI / Service

Jasmine and Karma :- Unit testing , Describe and IT

Integration with visual studio.

1. ng build angular and visual studio MVC
2. ng serve can notbe use ng buld—deployurl
3. index.html page copied in index.cshtml

Angular component life cyecl constructor , init , check , onchange

Angular 2 vs Angular 4

Up to 60% less code, generated by compiler;
Separate Animation package;
Angular 4 now allows developers to use an else clause after an if condition;
TitleCase pipe
Official support of Angular Universal
Better TypeScript Compatibility
Template Source Maps
Flat ES Modules

Angular 4 vs Angular 5
a) Build Optimizer: This is a tool which was included in the CLI to help the developers in creating a smaller bundle for the application. Apart from decreasing the users' bundle size, the feature also helps in increasing the boot speed of the application for the users.
b) Compiler Improvements: To enhance faster rebuilds for production and AOT (Ahead of Time) builds, Angular 5 supports incremental compilation.
c) New Router Lifecycle Events: This new feature was added to enable the developers in tracking the cycle of the router, starting from running guards to the completion of activation.
d) HttpClient: This feature has been recommended for all the application as HTTPClient was highly appreciated. The framework developer is not suggesting anymore, to use the previous @angular/HTTP library. Developers can update the HTTPClient in 3 easy steps:

Angular 5 vs Angular 6

The Angular Material Design Library
A new Tree component is now added in the Angular Material Design Package and the Component Dev Kit. It allows you to visualize tree structures in a more hierarchical order, like a list of files, for example. These new tree components come in both styled and un-styled versions, (Material's mat-tree) and (CDK's cdk-tree) respectively.

Angular Elements
Remember the Elements package? Angular 6 fully supports it now. What it did was allow us to use Angular components outside of Angular like in JQuery or VueJS apps.

This package primarily focuses on taking an advantage of web components that are supported by all modern web browsers (except Edge). Using the Elements Package, you can create Angular components and publish them as Web Components, which can then be used in any HTML page.

Turning a component into a custom element gives you an easy path for creating dynamic HTML content for your Angular app, and, using the Angular Elements package, it is even easier to create native custom elements.

Component Dev Kit (CDK)
The CDK was released in December of 2017, but the Angular Team has made some really neat improvements to it for the 6th version.

With the CDK you can now build your own library of UI components without using the Angular Material library. It also supports Responsive Web Design layouts so you don't have to use other libraries like Flex Layout or even learn using the CSS Grid. It covers them all.

Another brilliant improvement in the CDK includes the @angular/cdk/overlay package. This one has a new positioning logic that makes your pop-ups stay on screen very brilliantly.

Command Line Interface (CLI)
The Angular command-line interface is now equipped with new commands such as ng-update , which updates dependencies and code, and ng-add , which helps quickly add application features and also supports turning applications into progressive web apps.

Other than these new commands, the new CLI also allows developers to choose the ng-package for transpiling different libraries using the Bazel tool. Without the Bazel tool, you would have to build and package libraries yourself and trust me, the Bazel tool is a Godsend!

An Improved Service Worker
You can configure navigation URLS with the improved Service workers in Angular 6.

Web Pack Updated
Web pack module bundler has been updated to version4. By using the scope hosting technique, modules created will now be smaller.

Tree Shakable Services
You can apply Tree shaking on services as well. How great is that!

Multiple Validators For Your Forms
Those of you who had to fuss about passing more than one validator in your Formbuilders, your prayers have been answered because Angular 6 now allows you to pass multiple validators to the formBuilder.

Conclusion
Angular 6, in all its glory, demands that you test it yourself to fully realize the new adjustments and features. Some features click more to some developers. To me, these are my pain points resolved.