**What is Angular 8?**

**Ans:** Angular 8 is the latest version which is released on May 28, 2019, by the Angular community. It is the most prominent client-side TypeScript based framework that is mainly used to create dynamic web applications. Angular 8 is very much similar to the previous version of Angular but has some more added features. You can even upgrade your Angular CLI to Angular version 8 very easily.

Angular 8 is a client-side [TypeScript](https://www.javatpoint.com/typescript-tutorial) based framework used to create dynamic web applications. It is very similar to its previous versions except having some extensive features. It was developed by the same team that develops [AngularJS](https://www.javatpoint.com/angularjs-tutorial) and is a complete rewrite of its first version. Now, this is led and maintained by the Angular Team at Google and also by a community of individuals and corporations.

**List the key features of Angular 8?**

**Ans:** The key features of Angular 8 are as follows:

* Differential Loading to create two different production bundles of your app.
* New Dynamic Lazy Loading modules for imports.
* Supports Web Workers.
* Supports TypeScript version 3.4.
* Availability of New Workspace APIs and Builder.
* Bazel Support.
* Opt-In usage sharing.
* Ivy Rendering Engine: The new compiler for Angular version 8.
* ngUpgrade improvements.

Angular 8 is a very popular Typescript development framework used to develop Web Applications. Following is the list of most prominent features of Angular 8 that makes this framework powerful.

**Angular 8 support Cross-Platform Features**

* The Angular 8 platform can be used to develop Progressive Web Apps. By using Angular 8, you can develop an extremely fast website that can provide app-like experiences with zero-step installation and high-quality performance.
* With Angular 8, you can create native mobile apps by using strategies like Ionic, Cordova, or NativeScript.
* With Angular 8, by using native OS APIs, the developers can create Desktop-Installed apps for Windows, Mac, and Linux.

**Better Speed & Performance**

* Angular 8 supports code generation facility that covert the templates into highly optimized code for JavaScript virtual machines. That's why Angular 8 is super fast.
* Angular 8 apps and websites are SEO friendly.
* Angular 8 uses a new Component Router to provide a code-splitting facility that makes apps load faster.

**High Productivity**

* Angular 8 provides easy and powerful template syntax to create UI views.
* It provides Angular CLI, which is a command-line tool. This tool is used to start building fast, add components and tests, and then instantly deploy it.
* Angular's IDEs make development easy and help you to find errors instantly. You can also see the other feedback in editors and IDEs.

**Provide Full Development Support**

* Angular 8 supports animation. The developers can create high-performance animations with intuitive API and a very little code of Angular.
* The angular framework helps to create accessible applications with ARIA-enabled mechanism and developer guides.

**Do Angular 8 support the TypeScript version 3.4?**

**Ans:** Yes, Angular version 8 supports TypeScript 3.4 because it is mainly used to run the Angular 8 projects. Everything in Angular version 8 is written in TypeScript. So you need to upgrade your TypeScript version to 3.4.

 How Performance Improvements on the core in Angular 8?

Angular 8 has advanced level features which ensure systematic workflow and performance improvements. It has apparent features such as differential loading, CLI workflow improvements, Dynamic imports for lazy routes, Ivy rendering engine, Bazel, etc.

Can we upgrade the older version of Angular that we have installed on our system to the Angular 8 version? If yes, then how?

* First, check the older version of Angular that you are using. Run the **ng --version** command on the Node.js command prompt.
* Now, you have to uninstall the older version of Angular by using the following command.

1. npm uninstall -g angular cli // For Windows Open Powershell on Administrator Mode
2. sudo npm uninstall -g angular-cli // For Mac

* Now, verify and clear the cache by using the following commands.

1. npm cache verify
2. npm cache clean

* Now, install the latest Angular CLI (Angular 8) by using the following command.

1. npm install -g @angular/cli@latest

* You can now verify that you are using the correct version by using the following command.

1. ng --version

**What is Ivy in Angular 8?**

**Ans:** Ivy in Angular version 8 is considered as the Rendering Engine. It was released in Angular 8 as Opt-in. It has opted as the code name for Angular’s next-generation rendering pipeline and compilation. By default, Ivy is intended to be the rendering engine in Angular 9

**What is the main purpose of a Bazel in Angular 8?**

Bazel is a new feature of Angular 8. It is a new build system available for a short period and provides a platform to make your back-end and front-end with a similar tool.

**Following is the list of main features of Bazel:**

* Bazel is an internal build tool that is used to customize the application.
* Bazel is used to test the action and performance of multiple machines.
* Bazel facilitates you to construct a graph through which you can identify the useful information.
* Bazel also supports customization.

**Ans:** Bazel is one of the key features present in Angular version 8. It always allows you to build CLI applications quickly. Bazel is considered a built tool that is developed and mostly used by Google as it can build applications in any language. The entire Angular framework is built with Bazel. Moreover, Bazel allows you to break an application into different build units which are defined at the NgModule level.

**What is the purpose of Codelyzer in Angular 8?**

**Ans:** Codelyzer in Angular version 8 is the open-source tool that is present on the top of the TSLint. The main purpose of Codelyzer is to verify whether the Angular TypeScript 3.4 projects are following the set of linting rules or not. It mainly focuses on the static code in Angular TypeScript. In simple words, we can say that the main purpose of the Codelyzer in Angular version 8 is to check the quality and correctness of the program.

It also checks the errors in codes, not following pre-defined rules. It contains more than 50 rules for checking if an Angular application follows best practices. It runs on the tslint.json file and checks only the static code in Angular 8. Codelyzer is by default inbuilt with the projects set up with the Angular command-line interface (CLI).

### Which command do you use to run and load the Angular App?

Use the **ng serve** command to run and load the Angular App.

### Which command is used to install the latest version of Angular CLI?

The following command is used to install the latest version of Angular CLI:

1. npm install -g @angular/cli@latest

**What is the purpose of Wildcard Route in Angular 8?**

**Ans:** The main purpose of the Wildcard Router in Angular version 8 is to match every URL as an instruction to get a clear client-generated view. This Wildcard route always comes last as it needs to perform its task at the end only. It is mainly used to define the route of the pages in Angular 8.

**The wildcard** is the last route in the appRoutes array of routes specified by the **\*\*path**. This route will be selected by the router if the requested URL doesn’t match any of the paths defined in the earlier configuration. It is mainly used in displaying a **“404 - Not Found”** page or to **redirect** to another route.

In Angular 8, the Wildcard Router is used to set a route when the requested URL doesn't match any router paths. After using the Wildcard Router, the set route matches to every [URL](https://www.javatpoint.com/url-full-form) as an instruction to get a clear client-generated view. This Wildcard route always comes last as it needs to perform its task at the end only. So, the Wildcard Router is mainly used to define the route of the pages in Angular 8.

In Angular 8, the Wildcard route is used to define the route of pages. You can make specific changes/updates while defining the route using Wildcard.

### How Angular 8 has enhanced the performance over its previous versions?

Angular 8 consists of advanced level features such as differential loading, CLI workflow improvements, Dynamic imports for lazy routes, Ivy rendering engine, Bazel, etc. which ensure systematic workflow and performance improvements.

**Which command is used to check the current version of Angular CLI?**

**Ans:** The ng -version command is used to check the current version of Angular CLI.

### What are the differences between promises and observables in Angular 8?

Following is the list of differences between promises and observables in Angular 8:

|  |  |
| --- | --- |
| **Promises** | **Observables** |
| Promises can deal with a single asynchronous event at a time. | The observables can handle a sequence of asynchronous events over a period of time. |
| Promises are always asynchronous. | Observables are both synchronous as well as asynchronous. |
| Promises generally provide only a single value. | Observables can emit multiple values. |
| Promises are not lazy. They can execute immediately after creation. | Observables are very lazy. They can't be executed until you subscribe to them using the subscribe() method. |
| Promises are not cancellable. | Observables have subscriptions that can be canceled using the unsubscribe() method. After that, they stop the listener from receiving further values. |
| Promises don't provide any operations. | Observables provide the map for forEach, filter, reduce, retry, and retryWhen operators. |
| Promises push errors to the child promises. | Observables deliver errors to subscribers. |

Consider the following Observable:

const observable = rxjs.Observable.create(observer => {

console.log('Text inside an observable');

observer.next('Hello world!');

observer.complete();

});

console.log('Before subscribing an Observable');

observable.subscribe((message)=> console.log(message));

When you run the above Observable, you can see messages being displayed in the following order:

Before subscribing an Observable  
Text inside an observable  
Hello world!

As you can see, observables are lazy. Observable runs only when someone subscribes to them hence, the message “Before subscribing…” is displayed ahead of the message inside the observable.  
  
Now let’s consider a Promise:

const promise = new Promise((resolve, reject) => {

console.log('Text inside promise');

resolve('Hello world!');

});

console.log('Before calling then method on Promise');

greetingPoster.then(message => console.log(message));

Running the above promise, the messages will be displayed in the following order:

Text inside promise  
Before calling then method on Promise  
Hello world!

As you can see the message inside Promise is displayed first. This means that a promise runs before the **then** method is called. Therefore, promises are **eager**.  
  
The next difference is that Promises are always **asynchronous**. Even when the promise is immediately resolved. Whereas an Observable, can be both **synchronous** and **asynchronous**.  
  
The above example of an observable is the case to show that an observable is synchronous. Let’s see the case where an observable can be asynchronous:

const observable = rxjs.Observable.create(observer => {

setTimeout(()=>{

observer.next('Hello world');

observer.complete();

},3000)

});

console.log('Before calling subscribe on an Observable');

observable.subscribe((data)=> console.log(data));

console.log('After calling subscribe on an Observable');

The messages will be displayed in the following order:

Before calling subscribe on an Observable  
After calling subscribe on an Observable  
Hello world!

You can see in this case, observable runs asynchronously.  
  
The next difference is that Observables can emit **multiple** values whereas Promises can emit only one value.  
  
The biggest feature of using observables is the use of **operators**. We can use multiple operators on an observable whereas, there is no such feature in a promise.

**Angular by default, uses client-side rendering for its applications.**

Can one make an angular application to render on the server-side?  
Yes, angular provides a technology called **Angular Universal**, which can be used to render applications on the server-side.  
  
The advantages of using Angular Universal are :

 First time users can instantly see a view of the application. This benefits in providing **better user experience**.

 Many search engines expect pages in plain HTML, thus, Universal can make sure that your content is available on every search engine, which leads to **better SEO**.

 Any server-side rendered application **loads faster** since rendered pages are available to the browser sooner.

**12. What are directives in Angular?**

A directive is a class in Angular that is declared with a **@Directive** decorator.  
Every directive has its own behaviour and can be imported into various components of an application.  
  
**When to use a directive?**  
Consider an application, where multiple components need to have similar functionalities. The norm thing to do is by adding this functionality individually to every component but, this task is tedious to perform. In such a situation, one can create a **directive** having the required functionality and then, import the directive to components which require this functionality.  
  
**Types of directives**  
**Component directives**  
These form the main class in directives. **Instead** of @Directive decorator we use **@Component** decorator to declare these directives. These directives have a view, a stylesheet and a selector property.  
  
**Structural directives**  
These directives are generally used to manipulate DOM elements.  
Every structural directive has a ‘ \* ’ sign before them.  
We can apply these directives to any DOM element.  
  
Let’s see some built-in structural directives in action:

<div \*ngIf="isReady" class="display\_name">

{{name}}

</div>

<div class="details" \*ngFor="let x of details" >

<p>{{x.name}}</p>

<p> {{x.address}}</p>

<p>{{x.age}}</p>

</div>

In the above example, we can \*ngIf and \*ngFor directives being used.  
  
\*ngIf is used to check a boolean value and if it’s truthy,the div element will be displayed.  
  
\*ngFor is used to iterate over a list and display each item of the list.  
  
**Attribute Directives**  
  
These directives are used to change the look and behaviour of a DOM element. Let’s understand attribute directives by creating one:  
  
How to create a custom directive?  
  
We’re going to create an attribute directive:  
  
In the command terminal, navigate to the directory of the angular app and type the following command to generate a directive:

ng g directive blueBackground

The following directive will be generated. Manipulate the directive to look like this:

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

@Directive({

selector: '[appBlueBackground]'

})

export class BlueBackgroundDirective {

constructor(el:ElementRef) {

el.nativeElement.style.backgroundColor = "blue";

}

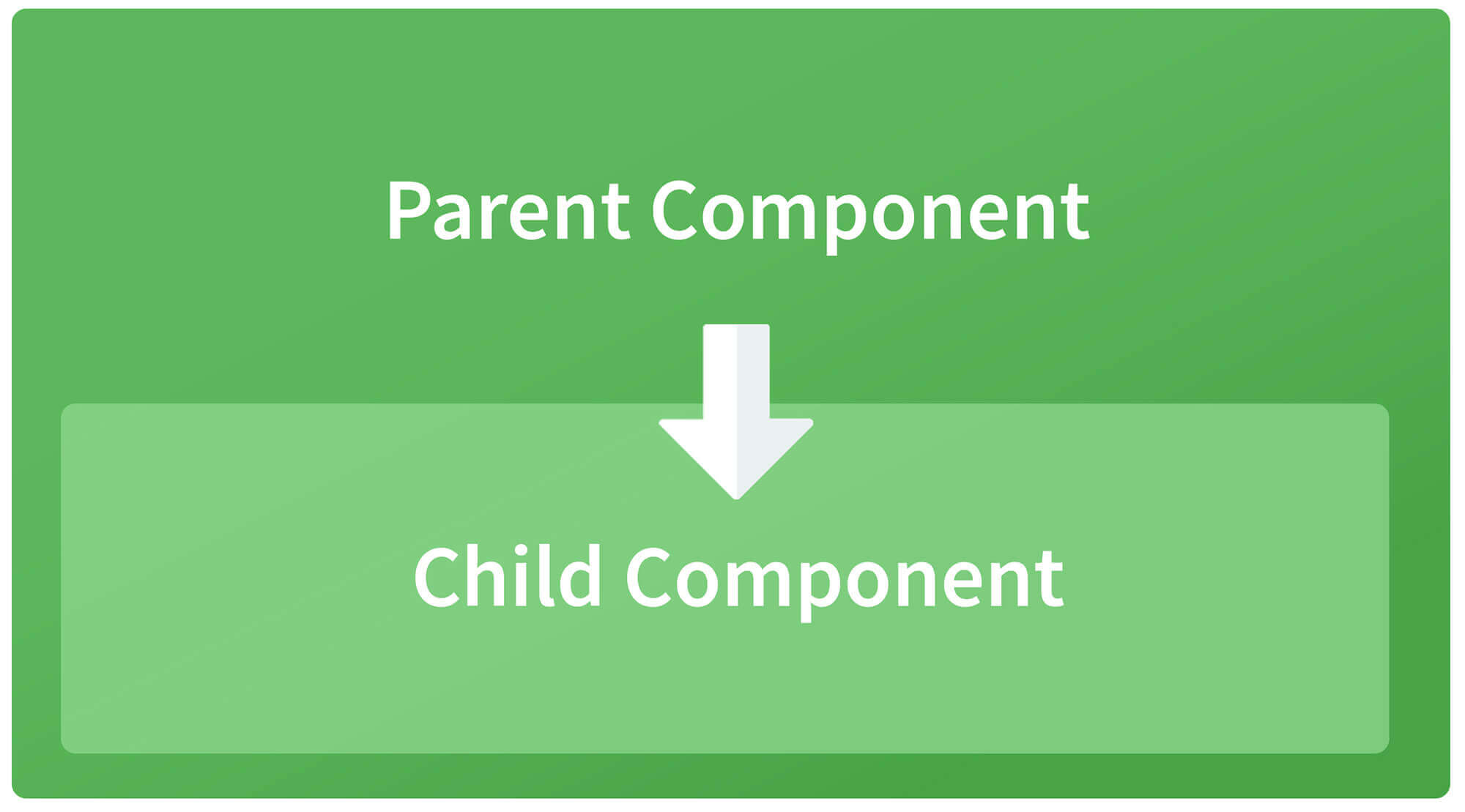
}

Now we can apply the above directive to any DOM element:

<p appBlueBackground>Hello World!</p>

**How does one share data between components in Angular?**

Following are the commonly used methods by which one can pass data between components in angular:



**Parent to child using @Input decorator**  
  
Consider the following parent component:

@Component({

selector: 'app-parent',

template: `

<app-child [data]=data></app-child>

` ,

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

})

export class ParentComponent{

data:string = "Message from parent";

constructor() { }

}

In the above parent component, we are passing “data” property to the following child component:

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

@Component({

selector: 'app-child',

template:`

<p>{{data}}</p>

`,

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

})

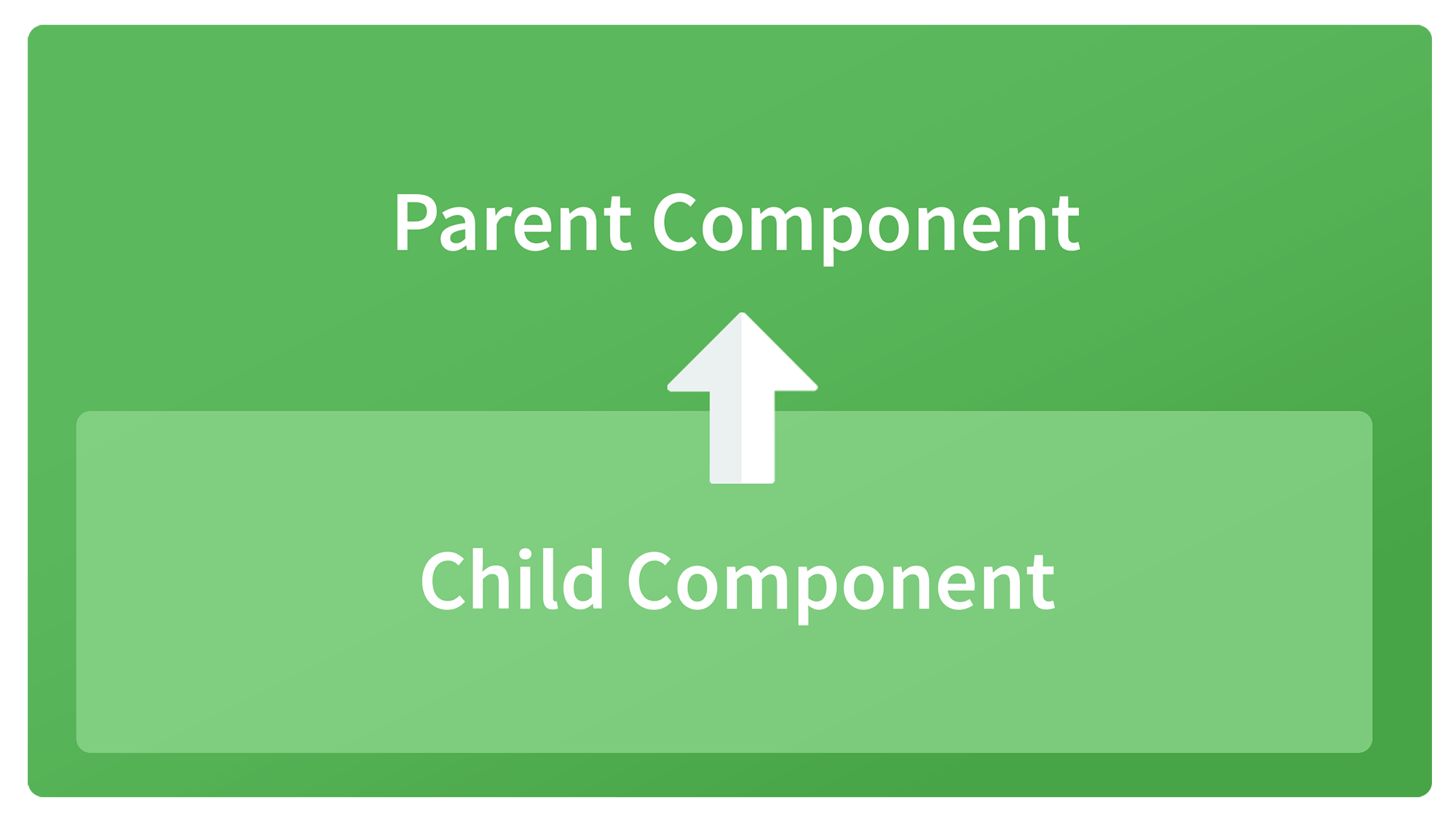
export class ChildComponent {

@Input() data:string

constructor() { }

}

In the child component, we are using @Input decorator to capture data coming from a parent component and using it inside the child component’s template.



**Child to parent using @ViewChild decorator**  
  
Child component:

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

@Component({

selector: 'app-child',

template:`

<p>{{data}}</p>

`,

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

})

export class ChildComponent {

data:string = "Message from child to parent";

constructor() { }

}

Parent Component

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

import { ChildComponent } from './../child/child.component';

@Component({

selector: 'app-parent',

template: `

<p>{{dataFromChild}}</p>

` ,

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

})

export class ParentComponent implements AfterViewInit {

dataFromChild: string;

@ViewChild(ChildComponent,{static:false}) child;

ngAfterViewInit(){

this.dataFromChild = this.child.data;

}

constructor() { }

}

In the above example, a property named “data” is passed from the child component to the parent component.  
**@ViewChild** decorator is used to reference the child component as “child” property.  
Using the **ngAfterViewInit** hook, we assign the child’s data property to the messageFromChild property and use it in the parent component’s template.  
  
**Child to parent using @Output and EventEmitter**  
  
In this method, we bind a DOM element inside the child component, to an event ( **click** event for example ) and using this event we emit data that will captured by the parent component:  
  
Child Component:

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

@Component({

selector: 'app-child',

template:`

<button (click)="emitData()">Click to emit data</button>

`,

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

})

export class ChildComponent {

data:string = "Message from child to parent";

@Output() dataEvent = new EventEmitter<string>();

constructor() { }

emitData(){

this.dataEvent.emit(this.data);

}

}

As you can see in the child component, we have used **@Output** property to bind an **EventEmitter**. This event emitter emits data when the button in the template is clicked.  
  
In the parent component’s template we can capture the emitted data like this:

<app-child (dataEvent)="receiveData($event)"></app-child>

Then inside the receiveData function we can handle the emitted data:

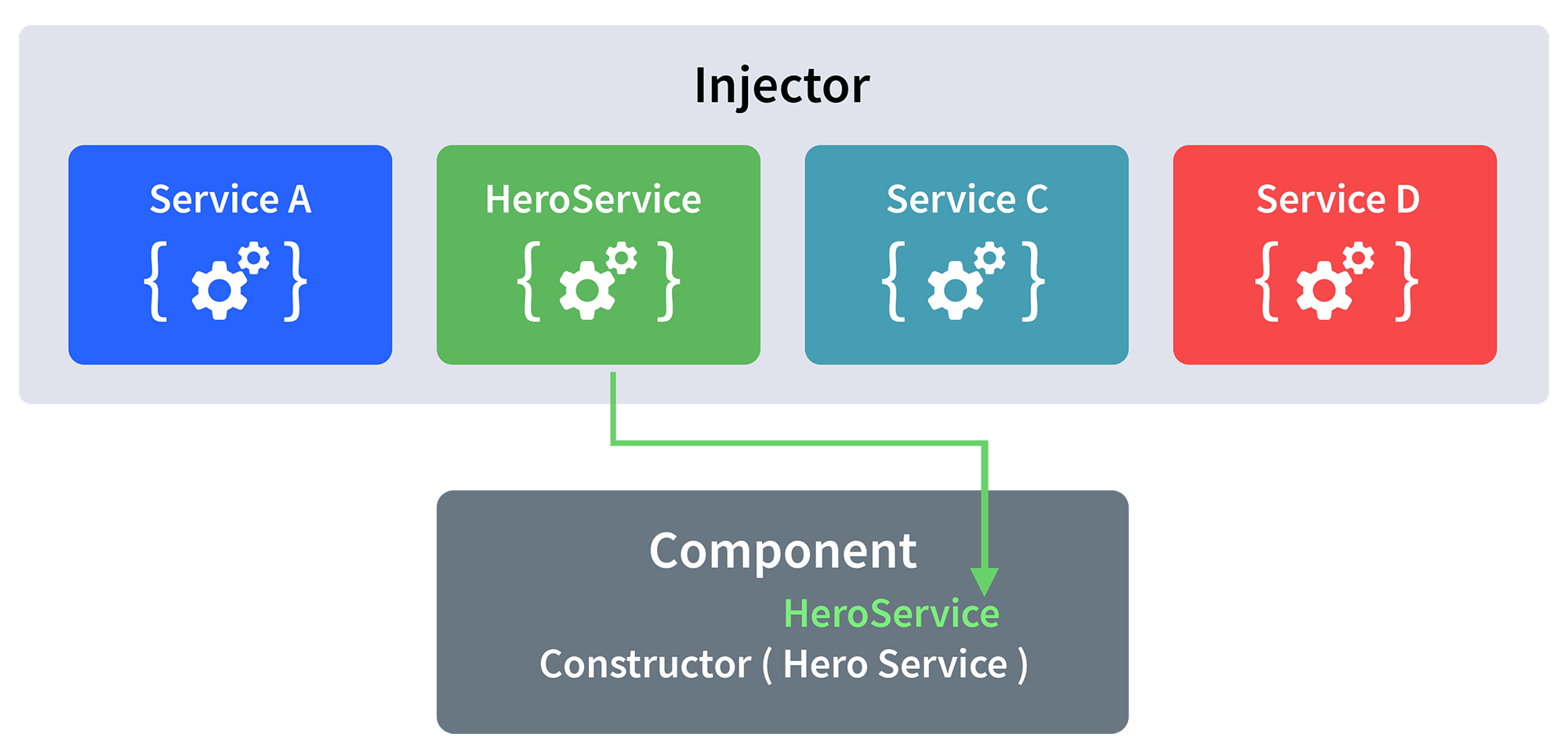
receiveData($event){

this.dataFromChild = $event;

}

**14. Explain the concept of Dependency Injection?**

Dependency injection is an application design pattern which is implemented by Angular.  
It also forms one of the core concepts of Angular.  
  
**So what is dependency injection in simple terms?**  
Let’s break it down, dependencies in angular are nothing but services which have a functionality. Functionality of a service, can be needed by various components and directives in an application. Angular provides a smooth mechanism by which we can inject these dependencies in our components and directives.  
So basically, we are just making dependencies which are injectable across all components of an application.



Let’s understand how DI (Dependency Injection) works:  
  
Consider the following service, which can be generated using:

ng g service test

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

@Injectable({

providedIn: 'root'

})

export class TestService {

importantValue:number = 42;

constructor() { }

returnImportantValue(){

return this.importantValue;

}

}

As one can notice, we can create injectable dependencies by adding the **@Injectable** decorator to a class.  
  
We inject the above dependency inside the following component:

import { TestService } from './../test.service';

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

@Component({

selector: 'app-test',

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

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

})

export class TestComponent implements OnInit {

value:number;

constructor(private testService:TestService) { }

ngOnInit() {

this.value = this.testService.returnImportantValue();

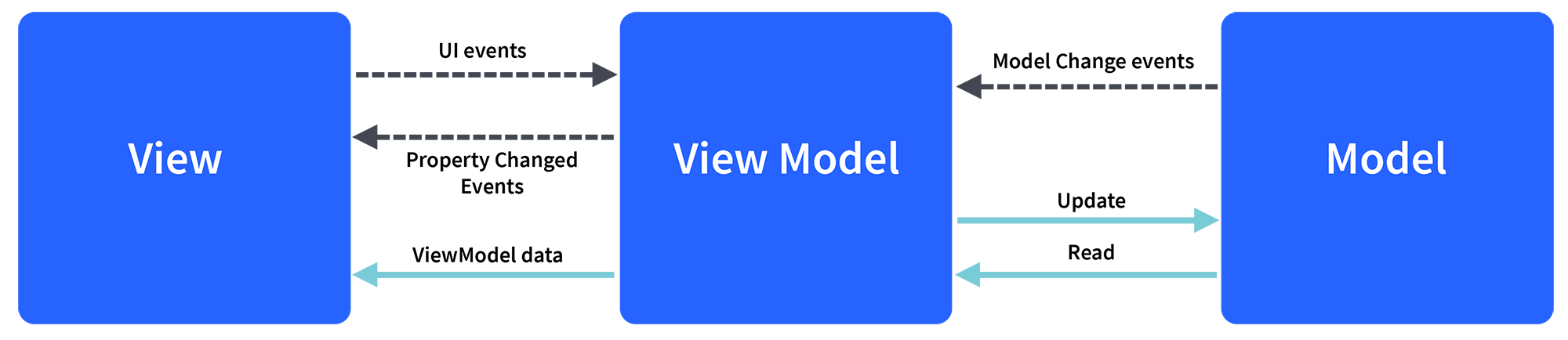
}

}

One can see we have imported our TestService at the top of the page. Then, we have created an instance inside the constructor of the component and implemented the **returnImportantValue** function of the service.  
  
From the above example, we can observe how angular provides a smooth way to inject dependencies in any component.

**15. Explain MVVM architecture**

MVVM architecture consists of three parts:  
  
1. Model  
2. View  
3. ViewModel



Model contains the structure of an entity. In simple terms it contains data of an object.  
View is the visual layer of the application. It displays the data contained inside the Model. In angular terms, this will be the HTML template of a component.  
  
ViewModel is an abstract layer of the application. A viewmodel handles the logic of the application. It manages the data of a model and displays it in the view.  
View and ViewModel are connected with data-binding (two-way data-binding in this case). Any change in the view, the viewmodel takes a note and changes the appropriate data inside the model.

## **What are the utility functions provided by RxJS?**

**Ans:**

1. Converting existing code for async operations into observables
2. Iterating through the values in a stream
3. Mapping values to different types
4. Filtering streams
5. Composing multiple streams

**What are the advantages of Bazel?**

**Ans:** The following are the key advantages of Bazel, and they are as follows:

* It provides you the possibility to make both frontends and backends with the same tool.
* Availability of incremental build and test options.
* In Bazel, you have the possibility for cache and remote builds on the build farm.

**What are the limitations of web workers in Angular 8?**

**Ans:** The web workers in Angular version 8 are mainly used to speed up the work of the things in your application during the working of CPU. These web workers mainly allow you to run the CPU computations in a background thread. This, in turn, helps the main thread to free up and then update the user interface.

##### Here are the limitations of a Web Worker:

* A web worker cannot directly manipulate the DOM
* It has limited access to methods and properties of the window object.
* It cannot be run directly from the file system. A web worker needs a server to run.

Why we used Service Workers in Angular?

A Service Worker is used in Angular 8 to build the basic steps of converting an application into a Progressive Web App (PWA). Service workers function as network proxies and intercepting all outgoing HTTP requests made by the application and how to respond.

How can you hide an HTML element just by a button click in Angular?

Ng-hide command is used to hide HTML elements if an expression is true.

##### Here's an example:

<div ng-app="DemoApp" ng-controller="DemoController">  
   <input type="button" value="Hide Angular" ng-click="ShowHide()"/>  
   <div ng-hide="IsVisible">Angular 8</div>  
</div>  
<script type="text/javascript">  
   var app = angular.module('DemoApp',[]);  
   app.controller('DemoController',function($scope){  
   $scope.IsVisible = false;  
   $scope.ShowHide = function(){  
     $scope.IsVisible = $scope.IsVisible = true;  
   }  
});  
</script>

**What is the main difference between Angular 8 and Angular 7?**

**Ans:** The main difference between Angular 8 and Angular 7 are as follows:

|  |  |
| --- | --- |
| Angular 8 | Angular 7 |
| Angular 8 was released on May 28, 2019. | Angular 7 was released on October 18, 2018. |
| Angular 8 supports TypeScript version 3.4. Added support for SASS. | Angular 7 supports TypeScript version 2.9. |
| Ivy is the newly added compiler in Angular 8. | Compatibility Compiler is the newly added compiler in Angular 7. |
| This version makes Angular developer life easier and it is very easier and faster to use. | It is the major release in Angular and expands the entire platform with CLI, Angular material, etc. |
| Added Navigation type available during the Navigation in the Router. | Added a new interface known as a Bootstrap interface. |
| Added ng-command to build new projects using Bazel. | Introducing a new Pipe called a KeyValuePipe. Added new mapping elements. |

|  |  |
| --- | --- |
| **Angular 7** | **Angular 8** |
| Angular 7 is slightly difficult to use because it was expanded to the entire platform, including Core framework, Angular Material, CLI, etc. | Angular 8 is comparatively smaller, faster, and easier to use. It has made Angular developer's life easier by providing a lot of advanced features. |
| Angular 7 has some features such as Angular Material, CLI prompts, Drag, and Virtual and Drop & Component Dev Kit (CDK), Virtual scrolling, Application performance, Bundle budget, Angular compiler, Angular elements, NativeScript, Better error handling, etc. | Angular 8 has some advanced level unique features such as Ivy rendering Engine, Bazel support, Differential Loading, API builders, Support for $location, Router backward compatibility, Opt-In Usage sharing, web- workers, etc. along with updated angular core framework, Angular Material, and the Command Line Interface of Angular 7. |
| The most significant changes in Angular 7 were Component Dev Kit (CKD), Material design library, and virtual scrolling. | The most significant changes in Angular 8 were the Core framework, Angular, material library, and CLI. |
| Till the Angular 7 version, Angular was supported by all types of Node.js version. | Angular 8 supports the Node.js version 12 or later. |
| Angular 7 supports a lower version of the typescript language. | Angular 8 supports the updated Typescript t version 3.4.x |
| The Angular 7 command-line interface (CLI) prompts supported the older version of CLI v7.0.2. | Angular 8 provided the stable CLI Builder API to developers who want to customize the Angular CLI by adding or modifying commands. |

**Which command is used to run and load the Angular App?**

**Ans:** The command ng serve is used to run and load the Angular App.

**What are the key parts of the Angular 8 Architecture?**

**Ans:** The key parts of the Angular 8 Architecture are as follows:

* Modules
* Components
* Templates
* MetaData
* Data-Binding
* Directives
* Services
* Dependency Injection

**What is the need for Angular 8 components?**

**Ans:** The components in Angular 8 are the list of classes with decorators that mainly mark their own types and provide metadata to guide Angular to do things. Every application in Angular has at least one component called a root component. This root component is mainly used to connect page hierarchy with page DOM.

**What are NgModules in Angular 8?**

**Ans:** NgModules in Angular version 8 varies from other JavaScript modules. Each and every Angular app has at least one module called AppModule. The NgModule provides a bootstrap mechanism to launch different applications. Commonly, every Angular App contains many functional modules to do things. The key features of Angular 8 modules are as follows:

* Own functionality of the NgModule can be exported and can also be used by other NgModules.
* Angular 8 NgModule can import functionalities from other NgModules.

**What is NgUpgrade in Angular 8?**

**Ans:** NgUpgrade is the library present in the Angular version 8. This upgrade library is mainly used to integrate both AngularJS and Angular components in the application. Moreover, NgUpgrade also helps you in bridging the gap between the Dependency Injection Systems in both AngularJS and Angular.

NgUpgrade is the library present in the Angular version 8. This upgrade library is mainly used to integrate both AngularJS and Angular components in the application. Moreover, NgUpgrade also helps you in bridging the gap between the Dependency Injection Systems in both AngularJS and Angular.

**Which command is used to install the latest version of Angular CLI?**

**Ans:** The npm install -g @angular/cli@latest is the command which is used to install the latest version of Angular CLI.

**What is the use of nglf directive in Angular 8?**

**Ans:** The nglf directive is a structural directive present in Angular Version 8. This directive is mainly used to add or remove HTML elements based on the expression statement. The functioning of the nglf directive is observed when the expression is true the element is added and when the expression is false the element is removed using nglf directive.

**What is typeofchecks in Angular 8?**

**Ans:** The typeofchecks plays a key role in Angular 8, it is mainly used to check the type of value assigned to the variable the same as that of JavaScript. You can even test the value assigned to the object by using typeofchecks in version 8.

What do you mean by typeofchecks in Angular 8?

In Angular 8, the typeofchecks is used to check the type of value assigned to the variable. It is used same we used in [JavaScript](https://www.javatpoint.com/javascript-tutorial). In Angular version 8, you can also check the value assigned to the object by using typeofchecks.

19) What are the most important parts of the Angular 8 Architecture?

The architecture of an Angular 8 application follows some fundamental concepts. The basic building blocks are NgModules that are used to provide compilation context for components and collect related code into functional sets. A set of NgModules are used to define an Angular app.

Following is the list of most important parts of the Angular 8 architecture:

* Modules
* Components
* Templates
* MetaData
* Data-Binding
* Directives
* Services
* Dependency Injection

### What is the requirement and usage of Angular 8 components?

The Angular 8 components are the list of classes with decorators that mainly mark their own types and provide metadata that guide Angular to do things.

Every Angular application always has at least one component known as a root component that connects a page hierarchy with page DOM. Each component defines a class that contains application data and logic and is associated with an [HTML](https://www.javatpoint.com/html-tutorial) template that defines a view to be displayed in a target environment.

### 21) What is the usage of NgUpgrade in Angular 8?

The NgModules in Angular 8 is used for the following things:

* The NgModule is a class marked by the @NgModule decorator where the @NgModule decorator is used to take a metadata object that describes how to compile a component's template and how to create an injector at runtime.
* The NgModules is used to configure the injector and the compiler and also help to organize the related things together.
* The NgModule is used to identify the module's own components, directives, and pipes. It can also make some of them public, through the exports property, to facilitate external components to use them.

## **What are NgModules in Angular8?**

**Ans**: The NgModule provides a bootstrap mechanism to launch different applications. Commonly, every Angular App contains many functional modules to do things. The key features of Angular 8 modules are as follows:

* Own functionality of the NgModule can be exported and can also be used by other NgModules.
* Angular 8 NgModule can import functionalities from other NgModules.

## **Q17. What modules should you import in Angular to use [(ngModel)] and reactive forms?**

**Ans**: FormsModule and ReactiveformsModule.

## **What are the basic parts of an Angular application?**

**Ans**:

1. Modules,
2. Component,
3. Data Binding,
4. Template,
5. Directives,
6. Dependency Injection,
7. Services,
8. Routing.
9. **How would you approach solving (some problem) on a high level?**
10. **Ans**: The problem should be directly related to the work the candidate will actually be doing. You aren‘t looking for a perfect answer or even necessarily a correct answer. Instead, listen to how they approach solving a problem, their ability to break a problem down into parts, and if they can anticipate problems.

22) What is the usage of NgUpgrade in Angular 8?

The NgUpgrade is an Angular 8 library mainly used to integrate both AngularJS and Angular components in the application. The NgUpgrade library is also used to bridge the gap between the Dependency Injection Systems in both AngularJS and Angular.

23) What is the usage of Angular 8 ngFor Directive?

The Angular 8 ngFor directive is used to repeat a portion of the HTML template once per each item from an iterable list (Collection). The ngFor is an Angular structural directive and is similar to ngRepeat in AngularJS. Some local variables like Index, First, Last, odd, and even are exported by ngFor directive.

**Syntax for ngFor Directive**

1. <li \*ngFor="let item of items;"> .... </li>

Read more information about ngFor directive: <https://www.javatpoint.com/angular-8-ngfor-directive>

24) What is the usage of Angular 8 ngIf Directive?

The Angular 8 ngIf directive is a structural directive that is used to add or remove HTML elements according to the expression. The expression must return a Boolean value true or false. You can see the functioning of nglf directive clearly. If the expression is false, then the element is removed. Otherwise, the element is inserted. It is similar to the ng-if directive of AngularJS.

**Syntax for ngIf Directive**

1. <p \*ngIf="condition">
2. condition is **true** and ngIf is **true**.
3. </p>
4. <p \*ngIf="!condition">
5. condition is **false** and ngIf is **false**.
6. </p>

Read more information about ngIf directive: <https://www.javatpoint.com/angular-8-ngif-directive>

25) What is data binding in Angular 8?

Data Binding is one of the key concepts of Angular 8. It is the most eminent technique which is used to link your data to the view layer. It is used to make a communication between the DOM and the TypeScript code of your component. In simple words, you can say that data binding is a communication between the typescript code of your component and your template, which the user sees. It makes it easy to define interactive applications without worrying about pushing and pulling data.

There are two types of data binding i.e., one-way data binding, two-way data binding.

26) What is String Interpolation in Angular 8, and why is it used?

String Interpolation is a one-way data-binding technique in Angular 8. It is used to extract the output data from a TypeScript code to the HTML template view layer. It shows the data from the component to view layer in the form of curly braces. This interpolation technique adds the value of property to the component.

**String Interpolation Example:**

1. {{data}}

27) What is the Purpose of Event Binding in Angular 8?

Event binding is a technique in Angular 8 used to handle the events raised from the DOM like button click, mouse move etc. When the DOM event happens (eg. click, change, keyup, keydown), it calls the specified method in the component.

See an example of event binding. In this example, the playMusic() method from the component will be called when you will click the button:

**For example:**

1. <button (click)="playMusic()"></button>

Read more information about Angular 8 Event Binding: <https://www.javatpoint.com/event-binding-in-angular-8>

### 28) How can you create an app in Angular 8?

You can easily create an Angular 8 web app using Angular CLI. Type the following command to create the Angular 8 web app.

**Syntax:**

1. ng **new** angular\_app\_name

**Example:**

1. ng **new** myapp

The above command will create an Angular 8 app with the name "**myapp**." Then, the Angular CLI will automatically install the required NPM modules. Before this, you must ensure that you have [Node.js](https://www.javatpoint.com/nodejs-tutorial), Angular 8, and installed Angular CLI using the following command:

1. npm install -g @angular/cli

You can also manually create an app folder and install ng dependencies.

### What is the main purpose of Angular 8 forms?

The main purpose of Angular 8 forms is to handle the user's input. You can also use these Angular forms in your application to enable users to log in, update profiles, enter information, or perform many other data-entry tasks.

There are two approaches to handle the user's input through forms in Angular 8:

* Reactive forms
* Template-driven forms

Both approaches are used to collect user input events from the view, validate the user input, create a form model and data model to update, and provide a way to track changes.

### 30) What is the use of reactive forms in Angular 8?

Reactive forms use a model-driven approach to handle form inputs where values constantly change over time. It uses an explicit and immutable approach to manage the state of a form at a specific time. Every time the changes occur to the form state, it returns a new state. The reactive forms are used mainly for the following features:

* Reactive forms are more robust than template-driven forms.
* Reactive forms are more scalable, reusable, and testable.
* Reactive forms are preferred to use when the forms are a key part of the application, or the application is already built using reactive patterns. In both cases, reactive forms are best to use.

### 31) What are the benefits of using Template-driven forms in Angular 8?

Following are the benefits of using Template-driven forms:

* You should use Template-driven forms if you want to add a simple form to your application because template-driven forms use two-way data binding to update the data model in the component. By using this, you can see the changes instantly as you make changes in the template and vice versa.
* Template-driven forms are easy to use in the application, but they are not as scalable as Reactive forms, so they are preferred to use in test applications for learning purposes.
* Template-driven forms are mainly used if your application requires a very basic form and logic. It can easily be managed in a template.

### 32) What is the difference between reactive forms and Template-driven forms in Angular 8?

Both Reactive forms and Template-driven forms are used to manage and process data differently. Each form offers different advantages. Here, we have compared both forms to see the differences:

|  |  |
| --- | --- |
| **Reactive Forms** | **Template-driven forms** |
| Reactive forms are more robust, so they are best to use. | Template-driven forms are best if you want to add a simple form to your application. For example email list signup form. |
| Reactive forms are more scalable, reusable, and testable, so they are preferred to use in professional websites for the industry. | Template-driven forms are easy to use in the application, but they are not as scalable as Reactive forms, so they are preferred to use in test applications for learning purposes. |
| Reactive forms are most preferred to use if forms are a key part of your application, or your application is already built using reactive patterns. In both cases, reactive forms are best to use. | Template-driven forms are mainly used if your application requires a very basic form and logic. It can easily be managed in a template. |

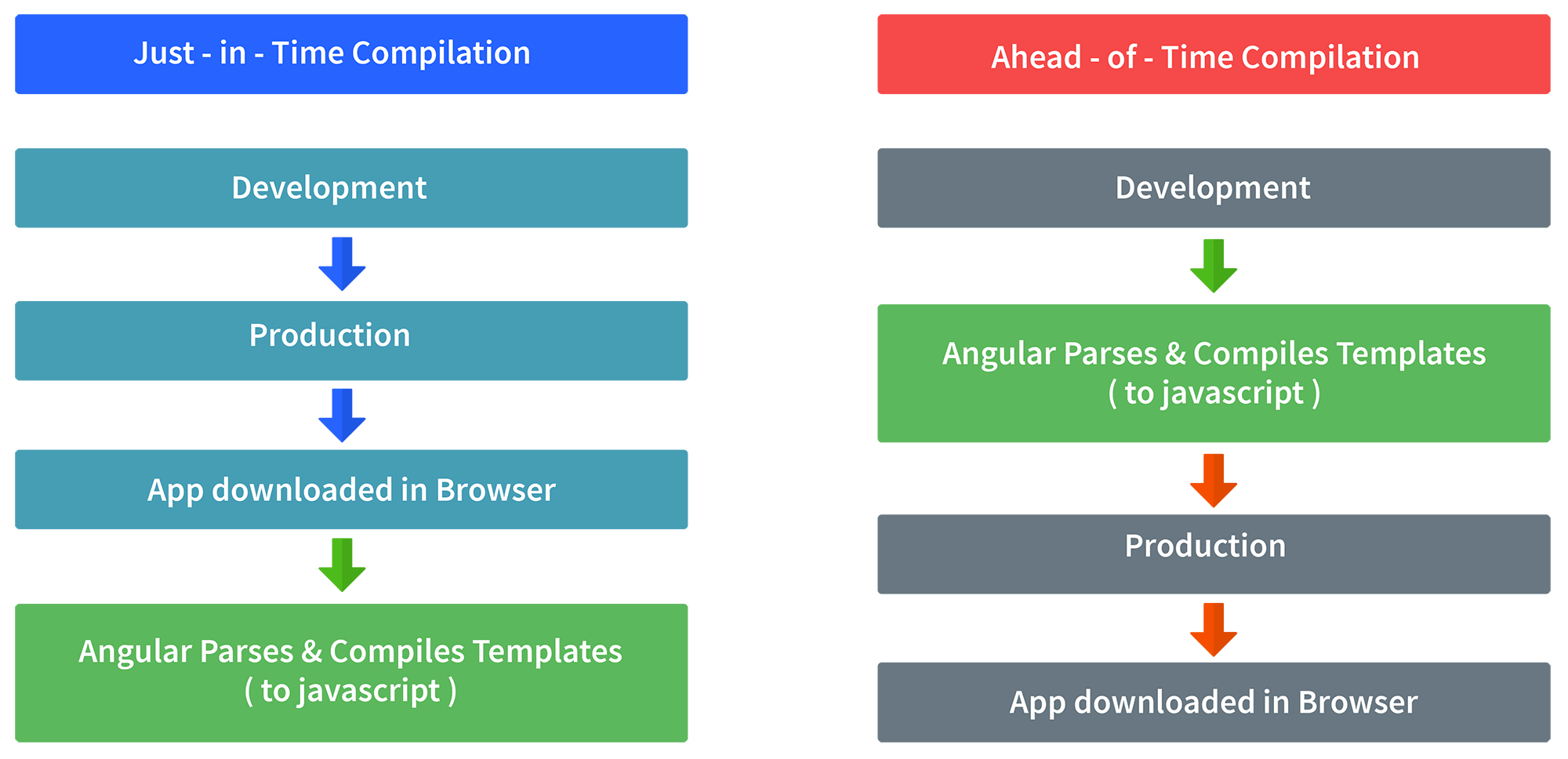
### 33) What do you mean by Angular Universal?

Angular Universal is a technology that is used to render Angular applications on the server. This process is called SSR (server-side rendering). A simple Angular application is executed on the browser as all the Angular applications are single-page applications, so the rendering always occurs on the browser. This process of rendering single-page applications is called the client-side rendering process (CSR). On the other hand, the Angular Universal is executed on the server, generating static application pages that later get bootstrapped on the client. That's why Angular Universal application generally renders more quickly, giving users a chance to view the application layout before it becomes fully interactive.

### 34) What is AOT in Angular 8?

In Angular 8, AOT stands for Ahead-of-Time compiler. It pre-compiles the application components and their templates during the build process.

There are several reasons why apps compiled with AOT launch faster:

* The application components that are compiled with AOT execute immediately, without client-side compilation.
* Here, templates are embedded as code within their components, so there is no client-side request for template files. That's why it is fast.
* The compiler doesn't entertain the unused Angular directives. It's also a reason for its fast response.
* Every Angular application consists of components and templates which the browser cannot understand. Therefore, all the Angular applications need to be compiled first before running inside the browser.  
    
  Angular provides two types of compilation:
*  JIT(Just-in-Time) compilation
*  AOT(Ahead-of-Time) compilation
* 
* In JIT compilation, the application compiles inside the browser during runtime.  
  Whereas in the AOT compilation, the application compiles during the build time.  
    
  The advantages of using AOT compilation are:
*  Since the application compiles before running inside the browser, the browser loads the executable code and renders the application immediately, which leads to **faster rendering**.
*  In AOT compilation, the compiler sends the external HTML and CSS files along with the application, eliminating separate AJAX requests for those source files, which leads to **fewer ajax requests**.
*  Developers can detect and handle errors during the building phase, which helps in **minimizing errors**.
*  The AOT compiler adds HTML and templates into the JS files before they run inside the browser. Due to this, there are no extra HTML files to be read, which provide **better security** to the application.
* By default, angular builds and serves the application using JIT compiler:
* ng build  
  ng serve
* For using AOT compiler following changes should be made:
* ng build --aot  
  ng serve --aot
* **Explain Components, Modules and Services in Angular**
* For better understanding, I would like you to create an Angular application by running the following inside the command terminal:
* ng new angularApp
* The above command will create an angular application in the directory.  
  Next, let's move on to understand Components, Modules, and Services.  
    
  **Components**  
  In Angular, components are the basic building blocks, which control a part of the UI for any application.  
  A component is defined using the **@Component** decorator. Every component consists of three parts, the template which loads the view for the component, a stylesheet which defines the look and feel for the component, and a class that contains the business logic for the component.  
  For creating a component, inside the command terminal, navigate to the directory of the application created, and run the following command:
* ng generate component test
* Or
* ng g c test
* One can see the generated component inside src/app/test folder. The component will be defined inside test.component.ts and this is how it looks:

* import { Component, OnInit } from '@angular/core';
* @Component({
* selector: 'app-test',
* templateUrl: './test.component.html',
* styleUrls: ['./test.component.css']
* })
* export lass TestComponent implements OnInit {
* constructor() {}
* ngOnInit() {
* }
* }

* As we can see in the above image, our component is defined with **@Component** decorator.
* **Modules**  
  A module is a place where we can group components, directives, services, and pipes. Module decides whether the components, directives, etc can be used by other modules, by exporting or hiding these elements. Every module is defined with a @NgModule decorator.  
  By default, modules are of two types:
*  Root Module
*  Feature ModuleEvery application can have only one root module whereas, it can have one or more feature modules.  
  A root module imports **BrowserModule**, whereas a feature module imports **CommonModule**.  
    
  In the application that we created before, one can see that the root module is defined inside **app.module.ts** and this is how it looks:
* import { BrowserModule } from '@angular/platform-browser';
* import { NgModule } from '@angular/core';
* import { AppComponent } from './app.component';
* import { TestComponent } from './test/text.component';
* @NgModule({
* declarations: [
* AppComponent,
* TestComponent
* ],
* imports: [
* BrowserModule
* ],
* providers: [],
* bootstrap: [AppComponent]
* })
* export class AppModule { }

* We can see in the above image that the component we created earlier is already imported in the declarations array.  
    
  To create a feature module, run the following command:
* ng g m test-module
* The module is created inside the src/app/test-module/test-module.module.ts file:
* import { NgModule } from '@angular/core';
* import { CommonModule } from '@angular/common';
* @NgModule({
* declarations: [],
* imports: [
* CommonModule
* ]
* })
* export class TestModuleModule { }

* As one can see, **CommonModule** is imported since this is a feature module.
* **Services** Services are objects which get instantiated only once during the lifetime of an application. The main objective of a service is to share data, functions with different components of an Angular application.  
  A service is defined using a **@Injectable** decorator. A function defined inside a service can be invoked from any component or directive.  
    
  To create a service, run the following command:
* ng g s test-service
* The service will be created inside src/app/test-service.service.ts:
* import { Injectable } from '@angular/core';
* @Injectable({
* providedIn: 'root'
* })
* export class TestServiceService {
* constructor() { }
* }

* Any method/function defined inside the TestServiceService class can be directly used inside any component by just importing the service.

**What is the use of the ngSwitch directive in Angular 8?**

**Ans:** It is also a Structural directive present in Angular Version 8. The ngSwitch directive is mainly used to add or remove HTML DOM elements. It is much similar to the switch statement of C#. This directive is applied to a container element with the help of a switch expression.

**What is Data Binding in Angular 8?**

**Ans:** Data Binding in Angular version 8 is considered as the key concept, that is mainly used to build the communication between the DOM and the TypeScript code of your component. It is the most prominent technique used to link your data to the view layer. Data Binding also defines interactive applications very quickly and easily. It can be achieved either by one-way binding or two-way binding.

**What is the Purpose of String Interpolation in Angular 8?**

**Ans:** String Interpolation in Angular version 8 comes under the one-way Data Binding technique, it is mainly used to output or extract the data from a TypeScript coding to HTML template view layer. It represents the data from the component to view layer in the form of curly braces. This interpolation technique adds the value of property to the component.

Example for String Interpolation is {{data}}

**What is the use of Event Binding in Angular 8?**

**Ans:** Event Binding technique is mainly used to handle the events that are raised from DOM like mouse move, button click, and many more in Angular version 8. When the DOM event occurs, at that moment only the specified method is called in the component.

**What is the main purpose of Angular 8 forms?**

**Ans:** The main use of Angular 8 forms is to handle the user’s input. These forms can also be used in your applications to enable the user to login, enter personal info, update profile, perform various tasks related to data entry, and many more. Basically, there are two approaches to handle the user’s input by using forms. The two approaches are used to collect the user inputs from the view layer and they also provide the way to track different changes.

**Why are the reactive forms used in Angular 8?**

**Ans:** The reactive forms are used in Angular 8 because of the following features. They are:

* Reactive forms are testable, scalable, and reusable.
* They are more robust.
* They are mostly preferred when the forms are considered as the key part of your application.

**Why do we need Template-driven forms in Angular 8?**

**Ans:** In the following cases the template-driven forms are needed in Angular 8.

* If you want to add just a simple form to your application you need template-driven forms.
* They are very easy to use in the applications but they are not scalable.
* If any of your applications require a very basic logic and form then you must make use of template-driven forms.

**What is the syntax to create an app in Angular 8?**

Ans: The syntax to create an application in Angular version 8 is

ng new app\_name

**Ans:** The npm install -g @angular/cli is the command used to install Angular CLI.

**What is an Angular Universal?**

**Ans:** The Angular Universal is defined as the process of SSR (server-side rendering) of your particular application to HTML present on the Server. Basically, all the Angular applications are single-page applications so the rendering always occurs on the browser. The entire process of rendering single-page applications is known as the client-side rendering process (CSR).

**What are the advantages of Angular 8?**

**Ans:** The following are the advantages of Angular 8, and they are as follows:

* Higher performance
* Effective clean code development
* debugging templates
* Angular 8 supports multiple apps in a single domain.
* easier implementation.

#### **Question: What is Angular?**

**Answer:**Angular is a TypeScript-based [open-source web application framework](https://hackr.io/blog/top-10-web-development-frameworks-in-2020), developed and maintained by Google. It offers an easy and powerful way of building front end web-based applications.

Angular integrates a range of features like declarative templates, dependency injection, end-to-end tooling, etc. that facilitates web application development.

#### **Question: Why was Angular introduced as a client-side framework?**

**Answer:**Traditionally, VanillaJS and jQuery were used by developers to develop dynamic websites. As the websites became more complex with added features and functionality, it was hard for the developers to maintain the code. Moreover, there was no provision of data handling facilities across the views by jQuery. So, Angular was built to address these issues, thus, making it easier for the developers by dividing code into smaller bits of information that are known as Components in Angular.

Client-side frameworks permit the development of advanced web applications like SPAs which, if developed by VanillaJS, is a slower process.

Back in the day, web developers used VanillaJS and jQuery to develop dynamic websites but, as the logic of one's website grew, the code became more and more tedious to maintain. For applications that use complex logic, developers had to put in extra effort to maintain separation of concerns for the app. Also, jQuery did not provide facilities for data handling across views.  
For tackling the above problems, client-side frameworks like Angular came into the picture, which made life easier for the developers by handling separation of concerns and dividing code into smaller bits of information (In the case of Angular, called Components).  
Client-side frameworks allow one to develop advanced web applications like Single-Page-Application. Not that we cannot develop SPAs using VanillaJS, but by doing so, the development process becomes slower.

#### **Question: Define the ng-content Directive?**

**Answer**: Conventional HTML elements have some content between the tags. For instance:

<**p**>Put your paragraph here</**p**>

Now consider the following example of having custom text between angular tags:

<**app-work**>This won’t work like HTML until you use ng-content Directive</**app-work**>

However, doing so won’t work the way it worked for HTML elements. In order to make it work just like the HTML example mentioned above, we need to use the ng-content Directive. Moreover, it is helpful in building reusable components.

Know more about the [ng-content directive](https://blog.angular-university.io/angular-ng-content/).

#### **Question: Please explain the various features of Angular.**

**Answer:** There are several features of Angular that make it an ideal front end JavaScript framework. Most important of them are described as follows:

* **Accessibility Applications**

Angular allows creating accessible applications using ARIA-enabled components, built-in a11y test infrastructure, and developer guides.

* **Angular CLI**

Angular provides support for command-line interface tools. These tools can be used for adding components, testing, instant deploying, etc.

* **Animation Support**

Angular’s intuitive API allows the creation of high-performance, complex animation timelines with very little code.

* **Cross-Platform App Development**

Angular can be used for building an efficient and powerful desktop, native, and progressive web apps. Angular provides support for building native mobile applications using Cordova, Ionic, or NativeScript.

Angular allows creating high performance, offline, and zero-step installation progressive web apps using modern web platform capabilities. The popular JS framework can also be used for building desktop apps for Linux, macOS, and Windows.

* **Code Generation**

Angular is able to convert templates into highly-optimized code for modern JavaScript virtual machines.

* **Code Splitting**

With the new Component Router, Angular apps load quickly. The Component Router offers automatic code-splitting so that only the code required to render the view that is requested by a user is loaded.

* **Synergy with Popular Code Editors and IDEs**

Angular offers code completion, instant errors, etc. with popular source code editors and IDEs.

* **Templates**

Allows creating UI views with a simple and powerful template syntax.

* **Testing**

Angular lets you carry out frequent unit tests using Karma. The Protractor allows running faster scenario tests in a stable way.

#### **Question: State some advantages of Angular over other frameworks.**

**Answer:**

**Out of box Features:**Several built-in features like routing, state management, rxjs library, and HTTP services are straight out of the box services provided by Angular. So, one does not need to look for the above-stated features separately.

**Declarative UI:**Angular uses HTML to render the UI of an application as it is a declarative language and is much easier to use than JavaScript.

**Long-term Google Support:**Google plans to stick with Angular and further scale up its ecosystem as it has offered its long term support to Angular.

#### **Question: What is the difference between Angular and AngularJS.**

**Answer:**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **AngularJS** | **Angular** |
| **Architecture** | MVC or  Model-View-Controller architecture facilitates the AngularJS framework, where the Model contains the business logic and Controllers processes information, lastly, View shows the information present in the Model. | Angular replaces controllers with Components that are directives with a predefined template. |
| **Language** | AngularJS uses JavaScript language, which is a dynamically typed language. | Angular uses TypeScript language, a statically typed language, and a superset of JavaScript. Angular provides better performance while developing larger applications. |
| **Mobile Support** | Does not support mobile support. | Supported by all popular mobile browsers. |
| **Structure** | The process of maintaining code becomes tedious in the case of larger applications. | It is easier to maintain code for larger applications as it provides a better structure. |
| **Expression Syntax** | A developer needs to remember the correct ng-directive for binding an event or a property. | Property binding is done using "[ ]" attribute and event binding is done using "( )" attribute. |
| **Routing** | AngularJS uses $routeprovider.when() | Angular uses @RouteConfig{(…)} |
| **Speed** | The development effort and time are reduced significantly because of the two-way data binding | Angular is faster due to upgraded features. |
| **Dependency** **Injection** | AngularJS doesn’t support DI. | Angular supports a hierarchical Dependency Injection with unidirectional tree-based change detection. |
| **Support** | No official support or updates are available for AngularJS. | Angular has active support with updates rolling out every now and then. |

#### **Question: What are Lifecycle hooks in Angular? Explain some life cycles hooks**

**Answer:**Angular components enter its lifecycle from the time it is created to the time it is destroyed. Angular hooks provide ways to tap into these phases and trigger changes at specific phases in a lifecycle.

**ngOnChanges( ):** This method is called whenever one or more input properties of the component changes. The hook receives a SimpleChanges object containing the previous and current values of the property.

**ngOnInit( ):** This hook gets called once, after the ngOnChanges hook.

It initializes the component and sets the input properties of the component.

**ngDoCheck( ):** It gets called after ngOnChanges and ngOnInit and is used to detect and act on changes that cannot be detected by Angular.

We can implement our change detection algorithm in this hook.

**ngAfterContentInit( ):** It gets called after the first ngDoCheck hook. This hook responds after the content gets projected inside the component.

**ngAfterContentChecked( ):** It gets called after ngAfterContentInit and every subsequent ngDoCheck. It responds after the projected content is checked.

**ngAfterViewInit( )**: It responds after a component's view, or a child component's view is initialized.

**ngAfterViewChecked( ):** It gets called after ngAfterViewInit, and it responds after the component's view, or the child component's view is checked.

**ngOnDestroy( ):**It gets called just before Angular destroys the component. This hook can be used to clean up the code and detach event handlers.

**ngOnChanges( )** This hook/method is called before **ngOnInit** and whenever one or more input properties of the component changes.  
This method/hook receives a SimpleChanges object which contains the previous and current values of the property.  
  
**ngOnInit( )** This hook gets called once, after the **ngOnChanges** hook.  
It initializes the component and sets the input properties of the component.  
  
**ngDoCheck( )** It gets called after **ngOnChanges** and **ngOnInit** and is used to detect and act on changes that cannot be detected by Angular.  
We can implement our change detection algorithm in this hook. **ngAfterContentInit( )** It gets called after the first **ngDoCheck** hook. This hook responds after the content gets projected inside the component.  
  
**ngAfterContentChecked( )** It gets called after **ngAfterContentInit** and every subsequent **ngDoCheck**. It responds after the projected content is checked.  
  
**ngAfterViewInit( )** It responds after a component's view, or a child component's view is initialized.  
  
**ngAfterViewChecked( )** It gets called after **ngAfterViewInit**, and it responds after the component's view, or the child component's view is checked.  
  
**ngOnDestroy( )** It gets called just before Angular destroys the component. This hook can be used to clean up the code and detach event handlers.  
  
Let’s understand how to use **ngOnInit** hook, since it’s the most oftenly used hook. If one has to process lot of data during component creation, it’s better to do it inside **ngOnInit** hook rather than the constructor:

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() {

this.processData();

}

processData(){

// Do something..

}

}

As you can see we have imported OnInit but we have used **ngOnInit** function. This principle should be used with the rest of the hooks as well.

#### **Question: Could we make an angular application to render on the server-side?**

**Answer:**Yes, we can, with Angular Universal, a technology provided by Angular capable of rendering applications on the server-side.

The benefits of  using Angular Universal are:

* **Better User Experience:**Allows users to see the view of the application instantly.
* **Better SEO:**Universal ensures that the content is available on every search engine leading to better SEO.
* **Loads Faster:** Render pages are available to the browsers sooner, so the server-side application loads faster.

**Question: Explain Dependency Injection?**

**Answer:**Dependency injection is an application design pattern that is implemented by Angular and forms the core concepts of Angular.

Let us understand in a detailed manner. Dependencies in Angular are services which have a functionality. Various components and directives in an application can need these functionalities of the service. Angular provides a smooth mechanism by which these dependencies are injected into components and directives.

**Question: Describe the MVVM architecture.**

**Answer:**MVVM architecture removes tight coupling between each component. The MVVM architecture comprises of three parts:

* Model
* View
* ViewModel

The architecture allows the children to have reference through observables and not directly to their parents.

* **Model:** It represents the data and the business logic of an application, or we may say it contains the structure of an entity. It consists of the business logic - local and remote data source, model classes, repository.
* **View:**View is a visual layer of the application, and so consists of the UI Code(in Angular- HTML template of a component.). It sends the user action to the ViewModel but does not get the response back directly. It has to subscribe to the observables which ViewModel exposes to it to get the response.
* **ViewModel:** It is an abstract layer of the application and acts as a bridge between the View and Model(business logic). It does not have any clue which View has to use it as it does not have a direct reference to the View. View and ViewModel are connected with data-binding so, any change in the View the ViewModel takes note and changes the data inside the Model. It interacts with the Model and exposes the observable that can be observed by the View.

#### **Question: Demonstrate navigating between different routes in an Angular application.**

**Answer**: Following code demonstrates how to navigate between different routes in an Angular app dubbed “Some Search App”:

import **from** "@angular/router";  
.  
.  
.  
@Component({  
  selector: 'app-header',  
  template: `  
<nav **class**="navbar navbar-light bg-faded">  
  <a **class**="navbar-brand" (click)="goHome()">Some Search App</a>   
  <ul **class**="nav navbar-nav">  
    <li **class**="nav-item">  
      <a **class**="nav-link" (click)="goHome()">Home</a>   
    </li>  
    <li **class**="nav-item">  
      <a **class**="nav-link" (click)="goSearch()">Search</a>   
    </li>  
  </ul>  
</nav>  
 `  
})  
**class** **HeaderComponent** {  
  constructor(**private** router: Router) {}   
  goHome() {  
    **this**.router.navigate(['']);   
  }  
  goSearch() {  
    **this**.router.navigate(['search']);   
  }  
}

#### **Question: What is the AOT (Ahead-Of-Time) Compilation? What are its advantages?**

**Answer:**  An angular application consists of components and templates which a browser cannot understand. Therefore, every Angular application needs to be compiled before running inside the browser. The Angular compiler takes in the JS code, compiles it, and then produces some JS code. It is known as AOT compilation and happens only once per occasion per user.

There are two kinds of compilation that Angular provides:

JIT(Just-in-Time) compilation: the application compiles inside the browser during runtime

AOT(Ahead-of-Time) compilation: the application compiles during the build time.

**Advantages of AOT compilation:**

* **Fast Rendering:** The browser loads the executable code and renders it immediately as the application is compiled before running inside the browser.
* **Fewer Ajax Requests:** The compiler sends the external HTML and CSS files along with the application, eliminating AJAX requests for those source files.
* **Minimizing Errors:** Easy to detect and handle errors during the building phase.
* **Better Security:**Before an application runs inside the browser, the AOT compiler adds HTML and templates into the JS files, so there are no extra HTML files to be read, thus providing better security for the application.

#### **Question: Could you explain services in Angular?**

**Answer:** Singleton objects in Angular that get instantiated only once during the lifetime of an application are called services. An Angular service contains methods that maintain the data throughout the life of an application.

The primary intent of an Angular service is to organize as well as share business logic, models, or data and functions with various components of an Angular application.

The functions offered by an Angular service can be invoked from any Angular component, such as a controller or directive.

#### **Question: Discuss the advantages and disadvantages of using Angular?**

**Answer:** Following are the various advantages of using Angular:

* Ability to add a custom directive
* Exceptional community support
* Facilitates client and server communication
* Features strong features, such as Animation and Event Handlers
* Follows the MVC pattern architecture
* Offers support for static template and Angular template
* Support for two-way data-binding
* Supports dependency injection, RESTful services, and validations

Disadvantages of using Angular are enumerated as follows:

* Complex SPAs can be inconvenient and laggy to use due to their size
* Dynamic applications do not always perform well
* Learning Angular requires a decent effort and time

#### **Question: Enumerate some salient features of Angular 7.**

**Answer**: Unlike the previous versions of Angular, the 7th major release comes with splitting in @angular/core. This is done in order to reduce the size of the same. Typically, not each and every module is required by an Angular developer. Therefore, in Angular 7 each split of the @angular/core will have no more than 418 modules.

Also, Angular 7 brings drag-and-drop and virtual scrolling into play. The latter enables loading as well as unloading elements from the DOM. For virtual scrolling, the latest version of Angular comes with the package. Furthermore, Angular 7 comes with a new and enhanced version of the ng-compiler.

#### **Question: What is string interpolation in Angular?**

**Answer:** Also referred to as moustache syntax, string interpolation in Angular refers to a special type of syntax that makes use of template expressions in order to display the component data. These template expressions are enclosed within double curly braces i.e. {{ }}.

The JavaScript expressions that are to be executed by Angular are added within the curly braces and the corresponding output is embedded into the HTML code. Typically, these expressions are updated and registered like watches as a part of the digest cycle.

#### **Question: Explain Angular Authentication and Authorization.**

**Answer**: The user login credentials are passed to an authenticate API, which is present on the server. Post server-side validation of the credentials, a JWT (JSON Web Token) is returned. The JWT has information or attributes regarding the current user. The user is then identified with the given JWT. This is called authentication.

Post logging-in successfully, different users have a different level of access. While some may access everything, access for others might be restricted to only some resources. The level of access is authorization.

Here is a detailed post on Angular 7 - JWT Authentication Example & Tutorial: http://jasonwatmore.com/post/2018/11/16/angular-7-jwt-authentication-example-tutorial

#### **Question: Can you explain the concept of scope hierarchy in Angular?**

**Answer:** Angular organizes the $scope objects into a hierarchy that is typically used by views. This is known as the scope hierarchy in Angular. It has a root scope that can further contain one or several scopes called child scopes.

In a scope hierarchy, each view has its own $scope. Hence, the variables set by a view’s view controller will remain hidden to other view controllers. Following is a typical representation of a Scope Hierarchy:

* Root $scope
  + $scope for Controller 1
  + $scope for Controller 2
  + …
  + ..
  + .
  + $scope for Controller n

#### **Question: How to generate a class in Angular 7 using CLI?**

**Answer**:

ng generate **class** **Dummy** [**options**]

This will generate a class named Dummy.

#### **Question: Explain what is the difference between Angular and backbone.js?**

**Answer:** Following are the various notable differences between Angular and Backbone.js

* **Architecture**

Backbone.js makes use of the MVP architecture and doesn’t offer any data binding process. Angular, on the contrary, works on the MVC architecture and makes use of two-way data binding for driving application activity.

* **Community Support**

Being backed by Google greatly ups the community support received by the Angular framework. Also, extensive documentation is available. Although Backbone.js has a good level of community support, it only documents on Underscore.js templates, not much else.

* **Data Binding**

Angular uses two-way data binding process and thus is a bit complex. Backbone.js, on the contrary, doesn’t have any data binding process and thus, has a simplistic API.

* **DOM**

The prime focus of Angular JS is upon valid HTML and dynamic elements that imitate the underlying data for rebuilding the DOM as per the specified rules and then works on the updated data records.

Backbone.js follows the direct DOM manipulation approach for representing data and application architecture changes.

* **Performance**

Thanks to its two-way data binding functionality, Angular offers an impactful performance for both small and large projects.

Backbone.js has a significant upper hand in performance over Angular in small data sets or small webpages. However, it is not recommended for larger webpages or large data sets due to the absence of any data binding process.

* **Templating**

Angular supports templating via dynamic HTML attributes. These are added to the document to develop an easy to understand application at a functional level. Unlike Angular, Backbone.js uses [Underscore.js](https://en.wikipedia.org/wiki/Underscore.js) templates that aren’t fully-featured as Angular templates.

* **The Approach to Testing**

The approach to testing varies greatly between Angular and Backbone.js due to the fact that while the former is preferred for building large applications the latter is ideal for developing smaller webpages or applications.

For Angular, unit testing is preferred and the testing process is smoother through the framework. In the case of Backbone.js, the absence of a data binding process allows for a swift testing experience for a single page and small applications.

* **Type**

Angular is an open-source JS-based front-end web application framework that extends HTML with new attributes. On the other hand, Backbone.js is a lightweight JavaScript library featuring a RESTful JSON interface and MVP framework.

#### **Question: How do Observables differ from Promises?**

**Answer**: As soon as a [promise](http://andyshora.com/promises-angularjs-explained-as-cartoon.html) is made, the execution takes place. However, this is not the case with observables because they are lazy. This means that nothing happens until a subscription is made. While promises handle a single event, observable is a stream that allows passing of more than one event. A callback is made for each event in an observable.

#### **Question: Please explain the difference between Angular and AngularJS?**

**Answer:** Various differences between Angular and AngularJS are stated as follows:

* **Architecture - AngularJS supports the MVC design model. Angular relies on components and directives instead**
* **Dependency Injection (DI) - Angular supports a hierarchical Dependency Injection with unidirectional tree-based change detection. AngularJS doesn’t support DI**
* **Expression Syntax -** In AngularJS, a specific ng directive is required for the image or property and an event. Angular, on the other hand, use () and [] for blinding an event and accomplishing property binding, respectively
* **Mobile Support -** AngularJS doesn’t have mobile support while Angular does have
* **Recommended Language -** While JavaScript is the recommended language for AngularJS, TypeScript is the recommended language for Angular
* **Routing -** For routing, AngularJS uses $routeprovider.when() whereas Angular uses @RouteConfig{(…)}
* **Speed -** The development effort and time are reduced significantly thanks to support for two-way data binding in AngularJS. Nonetheless, Angular is faster thanks to upgraded features
* **Structure -** With a simplified structure, Angular makes the development and maintenance of large applications easier. Comparatively, AngularJS has a less manageable structure
* **Support -** No official support or updates are available for the AngularJS. On the contrary, Angular has active support with updates rolling out every now and then

#### **Question: Observe the following image:**



**What should replace the “?”?**

**Answer**: Directives. The image represents the types of directives in Angular; Attribute, structural, and custom.

#### **Question: Could you explain the concept of templates in Angular?**

**Answer:** Written with HTML, templates in Angular contains Angular-specific attributes and elements. Combined with information coming from the controller and model, templates are then further rendered to cater the user with the dynamic view.

#### **Question: Explain the difference between an Annotation and a Decorator in Angular?**

**Answer:** In Angular, annotations are used for creating an annotation array. They are only metadata set of the class using the Reflect Metadata library.

Decorators in Angular are design patterns used for separating decoration or modification of some class without changing the original source code.

#### **Question: What are directives in Angular?**

**Answer:** Directives are one of the core features of Angular. They allow an Angular developer to write new, application-specific HTML syntax. In actual, directives are functions that are executed by the Angular compiler when the same finds them in the DOM. Directives are of three types:

* Attribute Directives
* Component Directives
* Structural Directives

#### **Question: What are the building blocks of Angular?**

**Answer**: There are essentially 9 building blocks of an Angular application. These are:

1. **Components –** A component controls one or more views. Each view is some specific section of the screen. Every Angular application has at least one component, known as the root component. It is bootstrapped inside the main module, known as the root module. A component contains application logic defined inside a class. This class is responsible for interacting with the view via an API of properties and methods.
2. **Data Binding –** The mechanism by which parts of a template coordinates with parts of a component is known as data binding. In order to let Angular know how to connect both sides (template and its component), the binding markup is added to the template HTML.
3. **Dependency Injection (DI) –** Angular makes use of DI to provide required dependencies to new components. Typically, dependencies required by a component are services. A component’s constructor parameters tell Angular about the services that a component requires. So, a dependency injection offers a way to supply fully-formed dependencies required by a new instance of a class.
4. **Directives –** The templates used by Angular are dynamic in nature. Directives are responsible for instructing Angular about how to transform the DOM when rendering a template. Actually, components are directives with a template. Other [types of directives](https://angular.io/guide/attribute-directives) are attribute and structural directives.
5. **Metadata –** In order to let Angular know how to process a class, metadata is attached to the class. For doing so decorators are used.
6. **Modules –** Also known as NgModules, a module is an organized block of code with a specific set of capabilities. It has a specific application domain or a workflow. Like components, any Angular application has at least one module. This is known as the root module. Typically, an Angular application has several modules.
7. **Routing –** An Angular router is responsible for interpreting a browser URL as an instruction to navigate to a client-generated view. The router is bound to links on a page to tell Angular to navigate the application view when a user clicks on it.
8. **Services –** A very broad category, a service can be anything ranging from a value and function to a feature that is required by an Angular app. Technically, a service is a class with a well-defined purpose.
9. **Template –** Each component’s view is associated with its companion template. A template in Angular is a form of HTML tags that lets Angular know that how it is meant to render the component.

#### **Question: Please explain the differences between Angular and jQuery?**

**Answer:** The single biggest difference between Angular and jQuery is that while the former is a JS frontend framework, the latter is a JS library. Despite this, there are some similarities between the two, such as both features DOM manipulation and provides support for animation.

Nonetheless, notable differences between Angular and jQuery are:

* Angular has two-way data binding, jQuery does not
* Angular provides support for RESTful API while jQuery doesn’t
* jQuery doesn’t offer deep linking routing though Angular supports it
* There is no form validation in jQuery whereas it is present in Angular

#### **Question: Could you explain the difference between Angular expressions and JavaScript expressions?**

**Answer:** Although both Angular expressions and JavaScript expressions can contain literals, operators, and variables, there are some notable dissimilarities between the two. Important differences between Angular expressions and JavaScript expressions are enlisted below:

* Angular expressions support filters while JavaScript expressions do not
* It is possible to write Angular expressions inside the HTML tags. JavaScript expressions, contrarily, can’t be written inside the HTML tags
* While JavaScript expressions support conditionals, exceptions, and loops, Angular expressions don’t
* The first and perhaps, the biggest difference is that Angular expressions allow us to write JavaScript in HTML which is not the case when it comes to JavaScript expressions.  
  Next, Angular expressions are evaluated against a **local** scope object whereas JavaScript expressions against **global** window object. Let's understand that better with an example :  
    
  Consider the following component named test:
* import { Component, OnInit } from '@angular/core';
* @Component({
* selector: 'app-test',
* template: `
* <h4>{{message}}</h4>
* `,
* styleUrls: ['./test.component.css']
* })
* export class TestComponent implements OnInit {
* message:string = “Hello world”;
* constructor() { }
* ngOnInit() {
* }
* }

* As one can see that Angular expression is used to display **message** property of a component. Since we are using Angular expressions, in the present template, we cannot access a property outside of its local scope, which in this case is **TestComponent**.  
  This proves that Angular expressions are always evaluated based on **scope** object rather than the global object.  
    
  Next difference is how Angular expressions handle **null** and **undefined**.  
  Consider the following JavaScript example:
* <!DOCTYPE html>
* <html lang="en">
* <head>
* <meta charset="UTF-8">
* <meta name="viewport" content="width=device-width, initial-scale=1.0">
* <title>JavaScript Test</title>
* </head>
* <body>
* <div id="foo"><div>
* </body>
* <script>
* 'use strict';
* let bar = {};
* document.getElementById('foo').innerHTML = bar.x;
* </script>
* </html>

* If you run the above code, you will see **undefined** displayed on the screen. Although it’s not ideal to leave any property undefined, the user does not need to see this.  
  Now consider the following Angular example:
* import { Component, OnInit } from '@angular/core';
* @Component({
* selector: 'app-new',
* template: `
* <h4>{{message}}</h4>
* `,
* styleUrls: ['./new.component.css']
* })
* export class NewComponent implements OnInit {
* message:object = {};
* constructor() { }
* ngOnInit() {
* }
* }

* If you render the above component, you will **not** see undefined being displayed on the screen.  
    
  Next, in Angular expressions one **cannot** use loops, conditionals and exceptions.  
    
  The difference which makes Angular expressions quite beneficial is the use of **pipes**. Angular uses pipes(called filters in AngularJS), which can be used to format data before displaying it. Let’s see one predefined pipe in action:
* import { Component, OnInit } from '@angular/core';
* @Component({
* selector: 'app-new',
* template: `
* <h4>{{message | lowercase}}</h4>
* `,
* styleUrls: ['./new.component.css']
* })
* export class NewComponent implements OnInit {
* message:string = "HELLO WORLD";
* constructor() { }
* ngOnInit() {
* }
* }

* In the above code we have used a predefined pipe called **lowercase**, which transforms all the letters in lowercase. Therefore, if you render the above component, you will see “hello world” being displayed.  
    
  In contrast, JavaScript does not have the concept of **pipes**.

#### **Question: Can you give us an overview of Angular architecture?**

**Answer**: You can draw some like this:



Here is Angular Architecture in detail: https://angular.io/guide/architecture

#### **Question: What is Angular Material?**

**Answer**: It is a UI component library. [Angular Material](https://material.angular.io/) helps in creating attractive, consistent, and fully functional web pages as well as web applications. It does so while following modern web design principles, including browser portability and graceful degradation.

#### **Question: What is AOT (Ahead-Of-Time) Compilation?**

**Answer**: Each Angular app gets compiled internally. The Angular compiler takes in the JS code, compiles it and then produces some JS code. This happens only once per occasion per user. It is known as AOT (Ahead-Of-Time) compilation.

#### **Question: What is Data Binding? How many ways it can be done?**

**Answer**: In order to connect application data with the DOM (Data Object Model), data binding is used. It happens between the template (HTML) and component (TypeScript). There are 3 ways to achieve data binding:

1. Event Binding – Enables the application to respond to user input in the target environment
2. Property Binding – Enables interpolation of values computed from application data into the HTML
3. Two-way Binding – Changes made in the application state gets automatically reflected in the view and vice-versa. The ngModel directive is used for achieving this type of data binding.

#### **Question: What is demonstrated by the arrow in the following image?**

  
**Answer**: This represents a dependency injection or DI.

#### **Question: Can you draw a comparison between the service() and the factory() functions?**

**Answer:** Used for the business layer of the application, the service() function operates as a constructor function. The function is invoked at runtime using the new keyword.

Although the factory() function works in pretty much the same way as the service() function does, the former is more flexible and powerful. In actual, the factory() function are design patterns that help in creating objects.

#### **Question: Please explain the digest cycle in Angular?**

**Answer:** The process of monitoring the watchlist in order to track changes in the value of the watch variable is termed the digest cycle in Angular. The previous and present versions of the scope model values are compared in each digest cycle.

Although the digest cycle process gets triggered implicitly, it is possible to start it manually by using the $apply() function.

#### **Question: Could you explain the various types of filters in AngularJS.**

**Answer:** In order to format the value of expression so that it can be displayed to the user, AngularJS has filters. It is possible to add these filters to the controllers, directives, services, or templates. AngularJS also provides support for creating custom filters.

Organizing data in such a way so that it is displayed only when certain criteria are fulfilled is made possible using filters. Filters are added to the expressions using the pipe ‘|’ character. Various types of AngularJS filters are enumerated as follows:

* currency – Formats a number to the currency format
* date – Formats a data to some specific format
* filter – Selects a subset of items from an array
* json – Formats an object to a JSON string
* limitTo – Limits an array or string into a specified number of characters or elements
* lowercase – Formats a string to lowercase
* number – Formats a number to a string
* orderBy – Orders an array by an expression

#### **Question: What is new in Angular 6?**

**Answer**: Here are some of the new aspects introduced in Angular 6:

* Angular Elements – It allows converting Angular components into web components and embeds the same in some non-Angular application
* Tree Shakeable Provider – Angular 6 introduces a new way of registering a provider directly inside the @Injectable() decorator. It is achieved by using the providedIn attribute
* RxJS 6 – Angular 6 makes use of RxJS 6 internally
* i18n (internationalization) – Without having to build the application once per locale, any Angular application can have “runtime i18n”

What is the use of RxJS in angular8?

Angular 8 uses observables which are implemented using RxJS libraries to push code. The main job of RxJS is to work with asynchronous events.

What is runGuardsAndResolvers in Angular 8?

Angular 8 introduced a number of new and unique options to runGuardsAndResolvers. In general, runGuardsAndResolvers is an option which is used for the Angular router configuration in order to control the resolvers and guards. The first option available in runGuardsAndResolvers is pathParamsChange. Through, this option router will re-run the guards and resolvers. Whenever you want to control over the resolvers and guards, use runGuardsAndResolvers option in Angular 8.

Why Incremental DOM is Tree Shakable?

In Angular 8, the framework does not interpret components in an incremental DOM. It uses component references instructions, and if it does not refer to a particular instruction, it shall be left unused. Now, VIrtual DOM requires an external interpreter. Hence, not knowing which components to display, everything is shifted to the browser, making the DOM shakeable.

What is authentication and authorization in Angular?

| **Authentication** | **Authorization** |
| --- | --- |
| Process of verifying the user | Process of verifying that you have relevant access to any procedure |
| Methods: Login form, HTTP Authentication, HTTP digest, X 509 Certificates, and Custom Authentication method. | Methods: Access controls for URL, secure objects and methods and Access Control Lists (ACL) |

#### **Question: What is ngOnInit()? How to define it?**

**Answer**: ngOnInit() is a lifecycle hook that is called after Angular has finished initializing all data-bound properties of a directive. It is defined as:

**Interface** **OnInit** {

ngOnInit() : void

}

**Question**: **What is SPA** **(Single Page Application) in Angular? Contrast SPA technology with traditional web technology?**

**Answer**: In the SPA technology, only a single page, which is index.HTML, is maintained although the URL keeps on changing. Unlike traditional web technology, SPA technology is faster and easy to develop as well.

In conventional web technology, as soon as a client requests a webpage, the server sends the resource. However, when again the client requests for another page, the server responds again with sending the requested resource. The problem with this technology is that it requires a lot of time.

#### **Question: What is the code for creating a decorator?**

**Answer**: We create a decorator called Dummy:

**function** **Dummy**(target) {

dummy.log('This decorator is Dummy', target);

}

#### **Question: What is the process called by which TypeScript code is converted into JavaScript code?**

**Answer**: It is called Transpiling. Even though TypeScript is used for writing code in Angular applications, it gets internally transpiled into equivalent JavaScript.

#### **Question: What is ViewEncapsulation and how many ways are there do to do it in Angular?**

**Answer**: To put simply, ViewEncapsulation determines whether the styles defined in a particular component will affect the entire application or not. Angular supports 3 types of ViewEncapsulation:

* Emulated – Styles used in other HTML spread to the component
* Native – Styles used in other HTML doesn’t spread to the component
* None – Styles defined in a component are visible to all components of the application

#### **Question: Why prioritize TypeScript over JavaScript in Angular?**

**Answer**: TypeScript is a superset of Javascript as it is Javascript + Types or extra features like typecasting for variables, annotations, variable scope and much more. The typescript is designed in a way to overcome Javascript shortcomings like typecasting of variables, classes, decorators, variable scope and many more. Moreover, Typescript is purely object-oriented programming that offers a "Compiler" that can convert to Javascript-equivalent code.

* It has a wide range of tools
* It’s a superset of Angular
* It makes abstractions explicit
* It makes the code easier to read and understand.
* It takes most of the usefulness within a language and brings it into a JS environment without forcing you out.

Which command is used to run static code analysis of angular application?

The ng lint command is used to run static code analysis within an Angular application.

#### **Question: Discuss the lifecycle designed for directive and components in Angular JS especially for the newly introduced version 6.0?**

**Answer:**

Components and directive of Angular JS follow the following typical lifecycle.

* nhOnInit
* ngDoCheck
* ngOnDestroy
* Constructor
* ngOnChanges
* ngAfterContentInit (only for components)
* ngAfterContentChecked (only for components)
* ngAfterViewInit (only for components)
* ngAfterViewChecked (only for components)

#### **Question: Write the features for Angular 6 over Angular 5.**

**Answer:**Following are the features:

**1. Added ng update**

CLI command updates angular project dependencies to their latest versions. The ng update is a normal package manager tool to identify and update in normal package manager tools to identify and update other dependencies.

**2. Uses RxJS6**

This is the third party library (RxJS) and introduces two important changes as compared to RxJS5.

1. Introduces a new internal package structure.
2. Operator concept

To update of RxJS6, run the following command:

npm **install** --save rxjs@6

To update your existing Angular Project, run the following:

npm **install** --save rxjs-compat

**3. The <ng-template>**

Angular 6 uses <ng-template> instead of <template>

**4. Service Level Changes**

If in an earlier version, the user wanted to provide a service to the entire application, the user was required to add it to providers in the AppModule but it is not required in the case of Angular6.

**5. Renamed Operators**

Angular 6 has renamed following operators:

* do() => tap()
* catch() => catchError()
* finally() => finalize()
* switch()=>switchAll()
* throw() => throwError

#### **Question: Why was Angular introduced as a client-side framework?**

**Answer:**Traditionally, VanillaJS and jQuery were used by developers to develop dynamic websites. As the websites became more complex with added features and functionality, it was hard for the developers to maintain the code. Moreover, there was no provision of data handling facilities across the views by jQuery. So, Angular was built to address these issues, thus, making it easier for the developers by dividing code into smaller bits of information that are known as Components in Angular. Client-side frameworks permit the development of advanced web applications like SPAs which, if developed by VanillaJS, is a slower process.

#### **Question: State some advantages of Angular over other frameworks.**

**Answer:**

* **Out of box Features:**Several built-in features like routing, state management, rxjs library, and HTTP services are straight out of the box services provided by Angular. So, one does not need to look for the above-stated features separately.
* **Declarative UI:**Angular uses HTML to render the UI of an application as it is a declarative language and is much easier to use than JavaScript.
* **Long-term Google Support:**Google plans to stick with Angular and further scale up its ecosystem as it has offered its long term support to Angular.

**How does an Angular application work?**

Every Angular app consists of a file named **angular.json**. This file will contain all the configurations of the app. While building the app, the builder looks at this file to find the entry point of the application. Following is an image of the angular.json file:

"build": {

"builder": "@angular-devkit/build-angular:browser",

"options": {

"outputPath": "dist/angular-starter",

"index": "src/index.html",

"main": "src/main.ts",

"polyfills": "src/polyfills.ts",

"tsConfig": "tsconfig.app.json",

"aot": false,

"assets": [

"src/favicon.ico",

"src/assets"

],

"styles": [

"./node\_modules/@angular/material/prebuilt-themes/deeppurple-amber.css",

"src/style.css"

]

}

}

Inside the build section, the main property of the options object defines the entry point of the application which in this case is **main.ts**.  
The main.ts file creates a browser environment for the application to run, and, along with this, it also calls a function called **bootstrapModule**, which bootstraps the application. These two steps are performed in the following order inside the main.ts file:

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

platformBrowserDynamic().bootstrapModule(AppModule)

In the above line of code, **AppModule** is getting bootstrapped.  
The AppModule is declared in the app.module.ts file. This module contains declarations of all the components.  
Below is an example of app.module.ts file:

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

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

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

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule

],

providers: [],

entryComponents: [],

bootstrap: [AppComponent]

})

export class AppModule { }

As one can see in the above file, **AppComponent** is getting bootstrapped.  
This component is defined in **app.component.ts** file. This file interacts with the webpage and serves data to it.  
Below is an example of app.component.ts file:

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

@Component({

selector: 'app-root',

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

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

})

export class AppComponent {

title = 'angular';

}

Each component is declared with three properties:  
1. **Selector** - used for accessing the component  
2. **Template/TemplateURL** - contains HTML of the component  
3. **StylesURL** - contains component-specific stylesheets  
  
After this, Angular calls the **index.html** file. This file consequently calls the root component that is **app-root**. The root component is defined in **app.component.ts**.  
This is how the index.html file looks:

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Angular</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

<app-root></app-root>

</body>

</html>

The HTML template of the root component is displayed inside the <app-root> tags.  
  
This is how every angular application works.

. How to install Angular 8?

##### Steps for the Installation of Angular 8 environmental setup

###### Step 1

Before installing Angular IDE using Angular CLI tool, make sure that Node.js has already installed in your system.

* 1. If Node.js is not installed in your system install it using the following steps.
  + The basic requirement of Angular 8 is Node.js version 110.9.0 or later.
  + Download it using [https//nodejs.org/en/](https://www.bestinterviewquestion.com/https/nodejs.org/en/)
  + Install it on your system
  + Open node.js command prompt
  + Check the version run command, node-v in the console window

###### Step 2

In order to install Angular CLI, use the following commands 2) npm install –g @angular/cli or npm install –g @angular/cli@latest

###### Step 3

To check the node and angular CLI version, run command ng –version on the console terminal

What is the difference between real Dom and virtual Dom?

| **Real DOM** | **Virtual DOM** |
| --- | --- |
| DOM is a language-neutral interface allowing programs and scripts to dynamically access and update multiple objects like content, structure, and style of a document. | Is a collection of modules designed to provide a declarative way to represent the DOM for an application. |
| The DOM represents the document as nodes and objects. | A virtual DOM object is a representation of a DOM object, like a lightweight copy. |
| It is an object-oriented representation of a web page, modified with a scripting language like JavaScript. | Virtual DOM is ideal for mobile-first applications. |

How to rollback the whole object when navigating back to a page managed by Angular Router in Angular 8?

The angular router in Angular 8 enables rollback the whole object to the next as users perform application work. To present a specific component view a given URL, Angular Router is the foremost service that presents. In order to import an Angular router, it is mandatory to install a library package, @angular/router.

### Ques. 1. Is the Angular and AngularJS different from each other?

Ans. The simple differences that you should keep in mind about Angular and AngularJS are that-

* Angular is the upgraded version of AngularJS.
* Also angular reduces the workload of the developer with its upgraded framework as well as, also faster. Whereas, AngularJS makes it more time to complete the task.
* Secondly, Angular use language- ‘TypeScript’ and AngularJS use language- ‘JavaScript’ to develop a program.
* Early, AngularJS doesn’t support any mobile support but now in Angular provides mobile support.

### Ques. 2. If you’re using Angular in your project, what are the advantages of that?

Ans. Some advantages of using Angular in a project:

* It supports two-way data-binding.
* It follows MVC pattern architecture.
* As well as Validations supported.
* It supports static template and Angular template.
* You can add a custom directive.
* Also, supports RESTful services.
* Support for dependency injection.
* Client and server communication facilitated.
* Has strong features like Event Handlers, Animation, etc.

### Ques. 3. For what, you mainly use the Angular in your application?

Ans.  Angular is a complete front-end **framework** that is really helpful in creating websites that are reliable, scalable, modular, easier to develop, and test. Secondly, Angular is a complete web-solution by itself from the initial stages of building to the final deployment of your app.  
  
Since Angular uses **TypeScript,** it’s easier to scale and prevent silly and unexpected errors. And also it’s backed by Google.

### Ques. 4. What are the key features of Angular?

Ans. Valiant features of the Angular:

1. Templates.
2. Model View Controller (MVC).
3. Dependency Injection (DI).
4. Directive.
5. Code splitting.
6. Validation.
7. Testing.
8. Child-Parent Relationship.
9. Data Binding.
10. Localization.

### Ques. 5. What are the benefits of TypeScript in Angular?

Ans. There are five major benefits of TypeScript in Angular:

#### **1. Consistency.**

Code consistency is an important goal to strive for in any code base. If you or your team have to support production applications then you understand how important consistency is and why it leads to better maintenance.

#### **2. Productivity.**

Consistency brings productivity into the picture as well. Secondly, developers don’t have to worry as much about if they’re doing it the “right way”.

#### **3. Maintainability.**

Easy to maintain, as well as it uses a framework backed by a full-time development team combined with a robust open source community is a key priority for most enterprises.

#### **4. Modularity.**

Angular is all about organizing code into “buckets”. Everything you create whether it’s components, services, pipes, or directives has to be organized into one or more buckets.

#### **5. Catch Errors Early.**

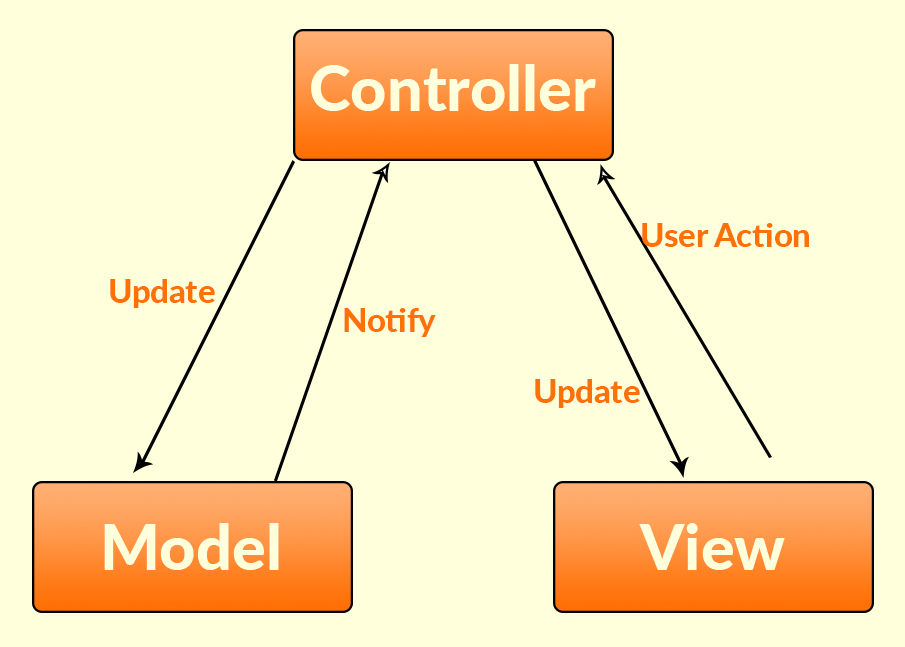
TypeScript in Angular, it quickly catches the errors.

### Ques. 6. What is a SPA?

Ans. Single Page Application (SPA) is a type of web application or website that dynamically reloads selected page elements in line with user interactions in order to avoid fetching entire new pages from a server. This can dramatically improve the speed and overall flow of a digital experience.  
  
Example of Single Page Application:

1. Netflix
2. Gmail
3. Facebook
4. Twitter

### Ques. 7. What is MVC Architecture?



Ans.  **MVC** Stands for “Model-View-Controller. The **MVC** model or “pattern” that commonly used for developing modern user interfaces. It provides fundamental pieces for designing programs for desktop or mobile, as well as web applications. Also, **MVC,** one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects.

### Ques. 8. What is the current version of angular?

Ans. The current version of Angular and its detail:  
-Version: Angular 8  
-Status: Active  
-Released: May 28, 2019

### Ques. 9. What’s the difference between an Angular component and module?

Ans. ***Components*** control views HTML. They also communicate with other components and services to bring functionality to your app.  
  
***Modules***consist of one or more components. As well as they do not control any HTML. As well as the modules declare which components can be used by components belonging to other modules, which classes will be injected by the dependency injector and which component gets bootstrapped. Secondly, Modules allow you to manage your components to bring modularity to your app.

### Ques. 10. What port does angular run on?

Ans. By default, Angular runs on port **4200** but it can configure as per demands.

### Ques. 11. Could you explain services in Angular?

Ans. Singleton objects in Angular that get instantiate only once during the lifetime of an application, called services. As well as, An Angular service contains methods that maintain the data throughout the life of an application.  
  
As well as, the primary intent of an Angular service is to organize it and also share business logic, models, as well as data and functions with different components of an Angular application. And the functions offered by it service can invoke from any component, such as a controller or directive.

### Ques.12. : What is ngOnInit ()? How to define it?

Ans. ngOnInit ()- lifecycle hook that called after Angular has finished initializing all data-bound properties of a directive. It defined as:

Interface OnInit {

ngOnInit () : void

}

### Ques. 13. Explain the difference between an Annotation and a Decorator in Angular?

Ans. In Angular, annotations use to creating an annotation array. They only metadata set of the class using the Reflect Metadata library. Decorators in Angular, design patterns used for separating decoration or modification of some class without changing the original source code.

### Ques. 14. What are directives in Angular?

Ans. Directives are one of the core features of Angular. Also, they allow an Angular developer to write new, application-specific HTML syntax. And in actual, directives are functions that execute by the Angular compiler when the same finds them in the DOM.  
Directives are of three types:  
  
Attribute Directives  
Component Directives  
Structural Directives

### Ques. 15. What is Angular Material?

Ans. It is a UI component library. Angular Material helps in creating attractive, consistent, and fully functional web pages as well as web applications. Also, it does so while following modern web design principles, including browser portability and graceful degradation.

### Ques. 16. What is the AOT (Ahead-Of-Time) Compilation?

Ans. Each Angular app gets compiled internally. The Angular compiler takes in the TypeScript code, compiles it and then produces some JavaScript code. As well as, this happens only when, once per occasion per user. It is known as AOT (Ahead-Of-Time) compilation.

### Ques. 17. What is NgRx?

Ans. NgRx is a group of Angular libraries for reactive extensions. Ngrx/Store implements the Redux pattern using the well-known RxJS observables of Angular 2. It provides several advantages by simplifying your application state to plain objects, enforcing unidirectional data flow, and more. The Ngrx/Effects library allows the application to communicate with the outside world by triggering side effects.

### Ques. 18. What is NGXS?

Ans. NGXS is a state management pattern + library for Angular. It acts as a single source of truth for your application’s state, providing simple rules for predictable state mutations. As well as, NGXS is modeled after the CQRS pattern popularly implemented in libraries like Redux and NgRx but reduces boilerplate by using modern TypeScript features such as classes and decorators.

### Ques. 19. What are observables?

Ans. Observables are just that – things you wish to observe and take action on. Angular 2 uses the Observer pattern which simply means – Observable objects are registered, and other objects observe (in Angular 2 using the subscribe method) them and take action when the observable object is acted on in some way.

They are similar to promises, but with some differences. Promises execute once and then are done. Observables continue to be observed after the event occurs. Observables can also be canceled (you can stop observing an object during runtime). Promises cannot be canceled – which makes sense since you’re only executing the promise one time.

import { Observable } from 'rxjs';

const observable = new Observable(observer => {

setTimeout(() => {

observer.next('Hello from a Observable!');

}, 2000);

});`

### Ques. 20. What does a Subscribe method do in Angular?

Ans. **Subscribe method!!** It is a method which subscribes to an observable. Whenever the subscribe method is called, an independent execution of the observable happens. In Angular .subscribe() is a method on the Observable type. The Observable type is a utility that asynchronously or synchronously streams data to a variety of components or services that have subscribed to the observable.  
So every time the observable emits some data, the subscribe method is triggered.

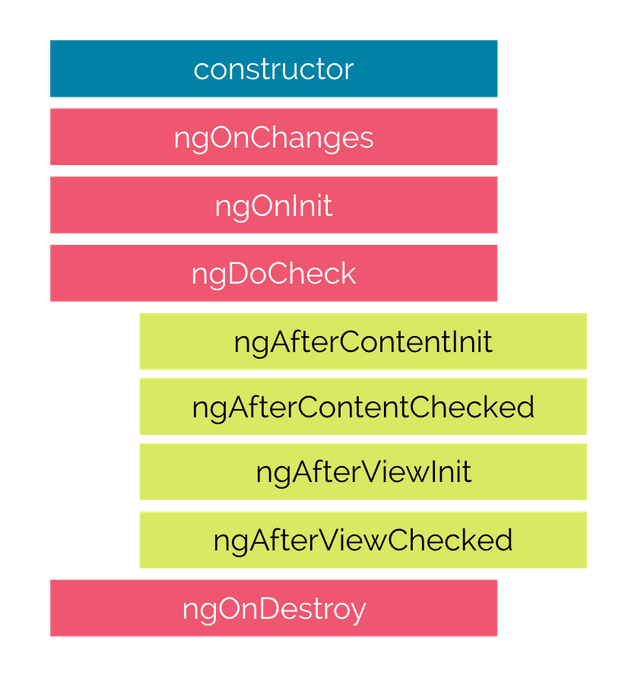
### Ques. 21. What is an AsyncPipe in Angular?

Ans. When an observable or promise returns something, we use a temporary property to hold the content. Later, we bind the same content to the template. Even, with the usage of AsyncPipe, the promise or observable can be directly used in a template and a temporary property is not required.

### Ques. 22. Explain the differences between one-way binding and two-way binding?

Ans. The one-way binding used to bind the data from the model to view without updating the HTML template or view automatically. Thus in order to update the HTML template, we need to write a custom code which updates the view every time whenever a data-bound from model to view. Secondly, whereas, two-way binding used to bind the data from the model to view and vice versa(i.e view to model) by automatically updating the HTML template without writing any custom code.

### Ques. 23. What is the sequence of Angular Lifecycle Hooks?



### Ques. 24. What are Pure and Impure Pipes?

Ans. Pure pipes are stateless that flow input date without remembering anything or causing detectable side-effects. Pipes are pure by default, hence most pipes are pure. As well as, we can make a pipe impure by setting its pure flag to false. Secondly, Angular executes a pure pipe only when it detects a pure change to the input value. Also, a pure change is either a change to a primitive input value or a changed object reference.

Impure pipes are those which can manage the state of the data they transform. Also, a pipe that creates an HTTP request, stores the response and displays the output, is an impure or stateful pipe. Secondly, Stateful Pipes should be used cautiously. Angular provides AsyncPipe, which is stateful. In the following code, the pipe only calls the server when the request URL changes and it caches the server response.

@Pipe({

name: 'fetch',

pure: false

})

export class FetchJsonPipe implements PipeTransform {

private cachedData: any = null;

private cachedUrl = '';

constructor(private http: Http) {}

transform(url: string): any {

if(url !== this.cachedUrl) {

this.cachedData = null;

this.cachedUrl = url;

this.http.get(url)

.map(result => result.json())

.subscribe(result => this.cachedData = result);

}

return this.cachedData;

}

}

### Ques. 25. What is Angular TestBed (ATB)?

Ans. The **Angular Test Bed (ATB)** is a higher level Angular Only testing framework that allows us to easily test behaviors that depend on the Angular Framework. Secondly, it is a slightly easier way to create components, handle injection, test asynchronous behavior and interacts with the application. As well as, the TestBed creates a dynamically-constructed Angular test module that emulates an Angular **@NgModule.**

## **Ques. 26. What are the Core Dependencies of Angular 7?**

Ans. There are two core dependencies, RxJS and TypeScript.

1. **RxJS 6.3** – RxJS version 6.3 is used by Angular 7. As well as, it has no changes in the version from Angular 6.
2. **TypeScript 3.1** – TypeScript version 3.1 is used by Angular 7. Also, it is the upgrade from version 2.9 of Angular 6.

### Ques. 27.  Demonstrate navigating between different routes in an Angular application.

Ans**.** Here coding which helps to this question:

import {Router} from "@angular/router";

.

.

.

@Component({

selector: 'app-header',

template: `

<nav class="navbar navbar-light bg-faded">

<a class="navbar-brand" (click)="goHome()">Some Search App</a>

<ul class="nav navbar-nav">

<li class="nav-item">

<a class="nav-link" (click)="goHome()">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" (click)="goSearch()">Search</a>

</li>

</ul>

</nav>

`

})

class HeaderComponent {

constructor(private router: Router) {}

goHome() {

this.router.navigate(['']);

}

goSearch() {

this.router.navigate(['search']);

}

}

### Ques. 28. What is the difference between Structural and Attribute directives in Angular?

Ans.**Structural directives** are used to alter the DOM layout by removing and adding DOM elements. Also, it is far better in changing the structure of the view. Examples of Structural directives are NgFor and Nglf.  
  
**Attribute Directives** These are being used as characteristics of elements. For example, a directive such as built-in NgStyle in the template Syntax guide is an attribute directive.

### Ques. 29. What are directives in Angular?

Ans. In Angular, directives used to add behavior to an existing DOM element as well as an existing component instance. For Example:

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

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

export class HighlightDirective {

constructor(el: ElementRef) {

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

}

}

### Ques. 30. Give the methods to handle events in Angular?

Ans. There are various methods to handle events. Like:

#### **1. Binding to user input events:**

You are able to use the Angular event binding to answer to DOM event. User input triggers so many DOM events. As well as it is a very effective method to get input from the user. For example,

<button (click)="onClickMe()">Click me!</button>

#### **2. Get user input from the event object:**

DOM carries information that possibly is valuable for the components. Below given example make you clean:

src/app/keyup.components.ts (template v.1)

content\_copy

template: `

<input (keyup)="onKey($event)">

<p>{{values}} </p>

#### **3. Key event filtering:**

Every keystroke is heard by the (key-up) event handler. The enter keys matter the most, as it provides the sign of the user that he has done with the typing. As well as, the most efficient method of eliminating the noise is to look after every ***event.keyCode*** and the action is taken only when the enter key is pressed.

### Ques. 31. As a developer, what is new in angular7 or 8?

Ans. Angular Elements is enabled to support content projection with the help of web standard for custom elements.

##### **– Angular Material Gets Mini Updates**

Firstly, Angular Material got better in the display that gives it an elegant look in the new update. Moreover, it also added a new homepage for the material, material.io. As well as, in this, you get tooling, design guidance, development components and stay up-to-date with the latest news. Secondly, if you are using an Angular Material v7 then you observe a visual difference as library make changes to itself with the updated version of the Material design.

##### **– More Better Accessibility**

In the updated version, it includes a lot of new features to enhance accessibility for selects. Secondly, it adds a new feature of the native select inside mat-form-field. It is far better and outperformed than the mat- select. As well as, both the select and mat-select are available so you can choose what you want to do.

##### **– Virtual Scrolling**

The Component Dev Kit (CDK) is available in the market with the great virtual scrolling capabilities that the user can apply by importing the `ScrollingModule`!

<cdk-virtual-scroll-viewport itemSize="20">

<div \*cdkVirtualFor="let dog of dogsArray"> {{dog}}</div>

</cdk-virtual-scroll-viewport>

##### **– Drag & Drop**

The CDK in the new update also now recommends Drag & Drop, which possess these great hallmarks.

##### **– Automated render as a user moves items**

It is a new feature available only in Angular 7

##### **– Helper methods for reordering/transferring items in lists**

For reordering or transferring items in lists, as well as a developer can use a helper method: moveItemInArray and transferArrayItem.

##### **– Enhancing Application Performance**

You will get enhanced application performance in Angular 7. It is the advanced version of Angular.

##### **– A safeguard has come into play for the users of Angular 7**

It gives a portent to new application builders when they are crossing the budget with their bundle size. Then the warning occurs on 2 MB whereas, an error occurs over 5 MB. Also, it can change the limits simply in an angular.json file. Secondly, the thing you have to do is add in a bit about the warnings and error sizes with budget details.

"budgets": [

{

"type": "initial",

"maximumWarning": "2mb",

"maximumError": "5mb"

}

]

### Ques. 32. Explain Angular Authentication and Authorization.

Ans. The user login credentials are passed to an authenticated API, which is present on the server. As well as, Post-server-side validation of the credentials, a JWT (JSON Web Token) is returned. The JWT has information or attributes regarding the current user. The user is then identified with the given JWT. This is called authentication. Secondly, Post-logging-in successfully, different users have a different level of access. While some may access everything, access for others might be restricted to only some resources. The level of access is authorization.

### Ques. 33. What is Angular Material?

Ans. It is a UI component library. Angular Material helps in creating attractive, consistent, and fully functional web pages as well as web applications. Secondly, it does so while following modern web design principles, including browser portability and graceful degradation.

### Ques. 34. Comparison between the service () and the factory() functions?

Ans**.** Used for the business layer of the application, the service() function operates as a constructor function. As well as, the function is invoked at runtime using the new keyword. Although the factory() function works in pretty much the same way as the service() function does, the former is more flexible and powerful. Secondly, the factory() function are design patterns that help in creating objects.

### Ques. 35. What are the various types of filters in an Angular?

Ans. Filters in Angular, it is possible to add these filters to the controllers, directives, services, or templates. Angular also provides support for creating custom filters. Secondly, Organizing data in such a way so that it is displayed only when certain criteria are fulfilled is made possible using filters. Various types of Angular filters are enumerated as follows:

* currency – Formats a number to the currency format
* date – Formats a data to some specific format
* filter – Selects a subset of items from an array
* JSON – Formats an object to a JSON string
* limitTo – Limits an array or string into a specified number of characters or elements
* lowercase – Formats a string to lowercase
* number – Formats a number to a string
* orderBy – Orders an array by an expression

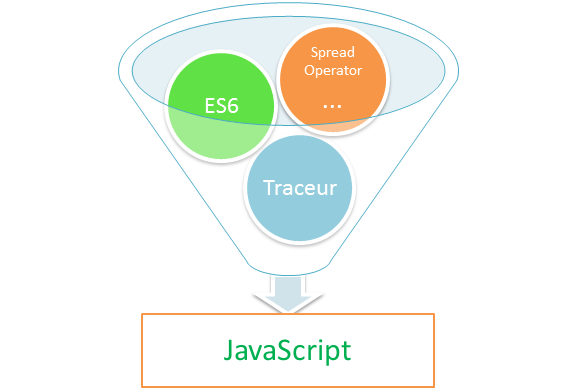
### Ques. 36. What is routing in Angular?

Ans. Angular provides a very powerful and simple routing mechanism. Since angular is a SPA no server requests are needed to navigate between routes which make the page loading instantaneous. Secondly, An Angular router provides functions like navigateByUrl(‘route’) for navigation which can also be used to pass some optional data. Angular also provides options for authorization while accessing various routes like AuthGuard(CanAct) and make the application safer.

### Ques. 37. Why prioritize TypeScript over JavaScript in Angular?

Ans. TypeScript is developed by Microsoft and it is a superset of JavaScript. As well as, the issue with JavaScript is that it isn’t a true OOP language. Secondly, the JavaScript code doesn’t follow the Prototype Pattern, the bigger the size of the code the messier it gets. Although, it leads to difficulties in maintainability as well as reusability. To offset this, TypeScript follows a strict OOP approach.

### Ques. 38. What do you understand by Traceur compiler in Angular?



Ans. Traceur is a compiler which takes ECMAScript and compiles it down to regular Javascript that runs in the browser. Traceur can be used in several ways like- typing or pasting the ES6 code into the read-eval-print-loop page, as well as, by including traceur in the web page and compiling ES6 code content on the fly, or many other ways. Even it is written in ES6, compiled to ES5.  
  
Secondly, the main goal of a traceur compiler is to inform the designs of Javascript features and allows us to write the code in a better manner. As well as, nowadays, traceur compilers are broadly used in Angular platform.

### Ques. 39. What CLI stands for? Also, explain it in few words.

Ans. CLI is Command Line Interface, which can be used to create the Angular application. As well as, using CLI, it can also create a unit secondly, end-to-end tests for the Angular application.

### Ques. 40. Can an Angular app be PWA? Or what is Angular PWA?

Ans. Yes, Angular can be a PWA i.e a progressive web app secondly, it is very much configurable. As well as, Progressive Web Apps, that allow any web application to feel and behave very much like a native app on a mobile device: Offline caching with service workers so your app can work without an internet connection.

### Ques. 41. What is Angular firebase?

Ans. Firebase authentication, as well as all backend related tasks, can be easily implemented in an Angular project using Firebase functions. Secondly, Firebase is a good choice for web or mobile apps developed with Angular because it provides highly useful backend services like real-time database, storage, authentication, etc.

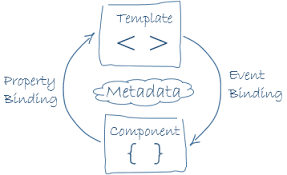
### Ques. 42. Do you think, Angular help in SEO?

Ans. Yes absolutely, With features like Server Side Rendering(SSR), service workers, PWA’s, AOT, MetaService, as well as TitleService Angular really helps in boosting up the SEO of an app as well as the website.

### Ques. 43. What is state management in Angular?

Ans. State management refers to the management of the state of one or more user interface controls such as text fields, OK buttons, radio buttons, etc. in a graphical user interface. Secondly, in this user interface programming technique, the state of one UI control depends on the state of other UI controls.  
  
For example, a state management UI control such as a button will be in the enabled state when input fields have valid input values as well as the button will be in the disabled state when the input fields are empty or have invalid values.  
  
Libraries like NgRx and NGXS can be used for state management in Angular.

### Ques. 44. What is metadata?



Ans. **Metadata** is a way of processing the class and a component called MyComponent will act as a class until we tell **Angular** that it’s a component. As well as, user can use **metadata** to the class to tell **Angular** that MyComponent is a component. As well as, **Metadata** can be attached to TypeScript using a decorator.

### Ques. 45. Differentiate between constructor and ngOnInit?

Ans. A constructor is a special method which will be called whenever we create new objects. And generally, it used initializing the class members. Also, it is a feature of the class(typescript) itself, an object-oriented design concept not Angular. Whereas, ngOnInit is a life cycle hook managed by Angular that being added to a prototype of the class created. As well as it is called by Angular when a component is initialized.

### Ques. 46. How to make API calls in Angular?

Ans. **API** calls in Angular can be executed by using the HttpClient which is an inbuilt package provided by Angular. All class can be made by the HttpClient of Angular like getting Requests, Post Requests, Put Requests, Delete Requests. **API** (Application Programming Interface) in Angular, it a set of global JavaScript functions use to carrying out the common tasks such as comparing objects, iterating objects, converting data.

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

import {Http, Response} from '@angular/http';

import {Observable} from 'rxjs/Observable';

import 'rxjs/add/operator/map';

@Injectable()

export class OombaDataService {

constructor(private http: Http) {}

private usersUrl = 'http://swapi.co/api/people/';

getData() {

return this.http.get(this.usersUrl)

.map(this.extractData)

}

private extractData(res: Response) {

let body = res.json();

return body.data || { };

}

private handleError (error: any) {

// In a real world app, we might use a remote logging infrastructure

// We'd also dig deeper into the error to get a better message

let errMsg = (error.message) ? error.message :

error.status ? `${error.status} - ${error.statusText}` : 'Server error';

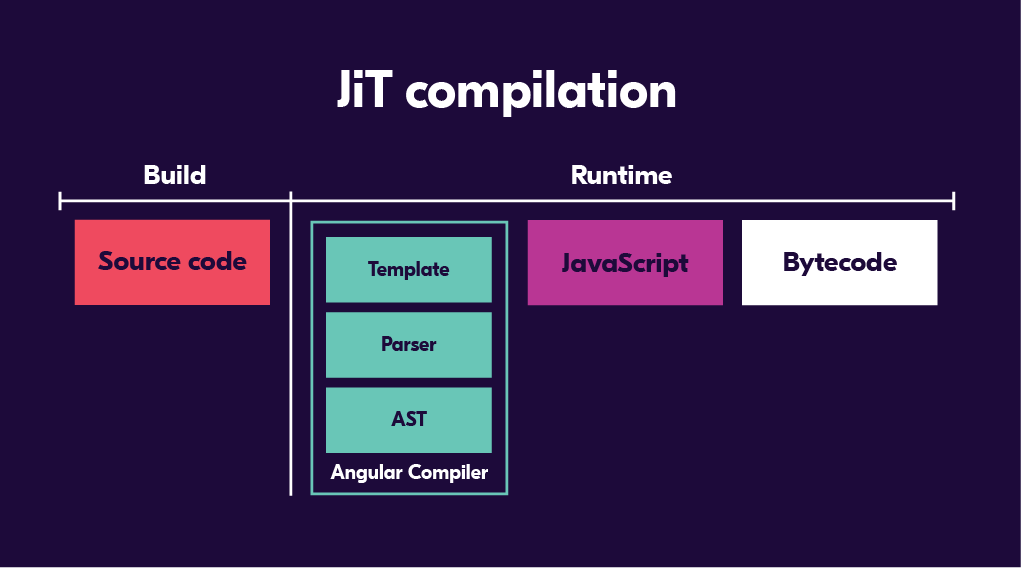
console.error(errMsg); // log to console instead

return Observable.throw(errMsg);

}

}

### Ques. 47. What is JIT?



Ans. **JIT** – Just-in-Time Compilation: **JIT** compilation as the name implies, compiles the application Just-in-Time in the browser at runtime. As well as, the vendor bundle contains the compiler along with the **angular** framework. Also, the compiler code is roughly half of the **Angular** framework.

### Ques. 48. What is a service worker in Angular?

Ans. Angular applications, as single-page applications, are in a prime position to benefit from the advantages of service workers. So, Starting with version 5 Angular ships with a service worker implementation.

As well as, Angular developers can take advantage of this service worker and benefit from the increased reliability and performance it provides, without needing to code against low-level APIs.  
  
Secondly, an Angular service worker is designed to optimize the end user experience of using an application over a slow or unreliable network connection, while also minimizing the risks of serving outdated content.

### Ques. 49. What do you understand by Angular elements?

Ans. Angular elements are Angular components packaged as **custom elements**(it is a web standard for defining new HTML elements in a framework). Angular Elements hosts an Angular component, providing a bridge between the data as well as logic defined in the component and standard DOM APIs, thus, providing a way to use Angular components.

### Ques. 50. What is Angular DSL?

Ans. A domain-specific language (DSL) is a computer language specialized to a particular application domain. As well as, Angular has its own Domain Specific Language (DSL) which allows us to write Angular specific HTLM-like syntax on top of normal HTML. As well as, it has its own compiler that compiles this syntax to HTML that the browser can understand. This DSL is defined in NgModules such as animations, forms, and routing and navigation.

There are 3 main syntaxes in Angular DSL

1. (): Used for Output and DOM events.
2. []: Used for Input and specific DOM element attributes.
3. \*: Structural directives (\*ngFor or \*ngIf) will affect/change the DOM structure.

#### **Question: What is the difference between Angular and AngularJS.**

**Answer:**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **AngularJS** | **Angular** |
| **Architecture** | MVC or  Model-View-Controller architecture facilitates the AngularJS framework, where the Model contains the business logic and Controllers processes information, lastly, View shows the information present in the Model. | Angular replaces controllers with Components that are directives with a predefined template. |
| **Language** | AngularJS uses JavaScript language, which is a dynamically typed language. | Angular uses TypeScript language, a statically typed language, and a superset of JavaScript. Angular provides better performance while developing larger applications. |
| **Mobile Support** | Does not support mobile support. | Supported by all popular mobile browsers. |
| **Structure** | The process of maintaining code becomes tedious in the case of larger applications. | It is easier to maintain code for larger applications as it provides a better structure. |
| **Expression Syntax** | A developer needs to remember the correct ng-directive for binding an event or a property. | Property binding is done using "[ ]" attribute and event binding is done using "( )" attribute. |
| **Routing** | AngularJS uses $routeprovider.when() | Angular uses @RouteConfig{(…)} |
| **Speed** | The development effort and time are reduced significantly because of the two-way data binding | Angular is faster due to upgraded features. |
| **Dependency** **Injection** | AngularJS doesn’t support DI. | Angular supports a hierarchical Dependency Injection with unidirectional tree-based change detection. |
| **Support** | No official support or updates are available for AngularJS. | Angular has active support with updates rolling out every now and then. |