Angular Interview Questions and Answers

First, we cover basic Angular interview questions. After that, we switch to more advanced questions. These interview questions on Angular should give you the theoretical knowledge to prepare for an interview, but of course you want some to get some coding practice in as well.

Basic Angular Interview Questions

1. What is Angular?

Angular is a TypeScript-based open-source web application framework, 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, and end-to-end tooling, that facilitate web application development.

2. Why was Angular introduced as a client-side framework?

Traditionally, VanillaJS and jQuery were used by developers to develop dynamic websites. As websites became more complex, with added features and functionality, developers found it increasingly hard to maintain the code. Furthermore, there was no provision of data handling facilities across the views by jQuery.

Angular was built to address these issues, making it easier for 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.

3. What are some features of Angular?

There are several features of Angular that make it an ideal front-end JavaScript framework. Some of the most important ones are:

- **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, and instant deploying, among other things.
- **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 desktop, native, and progressive web apps. It provides support for building native app development using Cordova, Ionic, or NativeScript. It can also be used to build 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:** 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 and instant error spotting 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 scenario tests faster while being stable.

4. State some advantages of Angular over other frameworks.

Some of Angular's advantages over other frameworks are:

- **Out of box Features:** Several built-in features like routing, state management, rxjs library, and HTTP services are available from the get-go.
- **Declarative UI:** Angular uses HTML to render the UI of an application as it is a declarative language, and this is much easier to use than in JavaScript.
- **Long-term Google Support:** Google plans to stick with Angular and further scale up its ecosystem as it offers long-term support for the framework.

5. What is the difference between Angular and AngularJS?

The 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, uses () and [] for blinding an event and accomplishing property binding, respectively.
- **Mobile Support:** AngularJS doesn't have mobile support while Angular does.
- **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 AngularJS. On the contrary, Angular has active support with updates rolling out every now and then.

6. Can we make an Angular application render on the server-side?

Yes, we can, with Angular Universal. 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.

7. Explain Dependency Injection.

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

Dependencies in Angular are services that simply have some functionality. Components and directives in an application may need these functionalities. Angular provides a smooth mechanism by which these dependencies are injected into components and directives.

8. What are services in Angular?

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.

9. What are the advantages and disadvantages of using Angular?

The following table explains the advantages and disadvantages of Angular:

Advantages	Disadvantages
Can add custom directives	Complex SPAs can be inconvenient and laggy to use due to their size

Exceptional community support	Dynamic applications do not always perform well
Follows the MVC pattern architecture	Learning Angular requires time and effort
Offers support for static template and Angular template	
Support for two-way data binding	
Supports dependency injection, RESTful services, and validations	

10. What are some features of Angular 7?

Unlike the previous versions of Angular, the 7th major release comes with the splitting of @angular/core. This is done in order to reduce the size. Not 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.

Angular 7 features drag-and-drop and virtual scrolling. Lastly, it also has a new and enhanced version of the ng-compiler.

11. What is string interpolation in Angular?

Also referred to as mustache 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.

12. Explain Angular Authentication and Authorization.

The user login credentials are passed to an authenticated 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 different levels of access. While some may access everything, access for others might be restricted to only some resources. The level of access is authorization.

13. How to generate a class in Angular 7 using CLI?

```
ng generate classDummy [options]
```

This will generate a class named Dummy.

14. Explain what is the difference between Angular and Backbone.js?

The following are the differences between Angular and Backbone.js

- **Architecture:** Backbone.js makes use of the MVP architecture and doesn't offer any data binding process. Angular 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. Extensive documentation is also available. Although Backbone.js has some community support, it only documents on Underscore.js templates, not much else.

- **Data Binding:** Angular uses a two-way data binding process and thus is a bit complex. Backbone.js doesn't have any data binding process and thus has a simplistic API.

- **DOM:** The prime focus of Angular JS is on 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.

- **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 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.

15. What are templates in Angular?

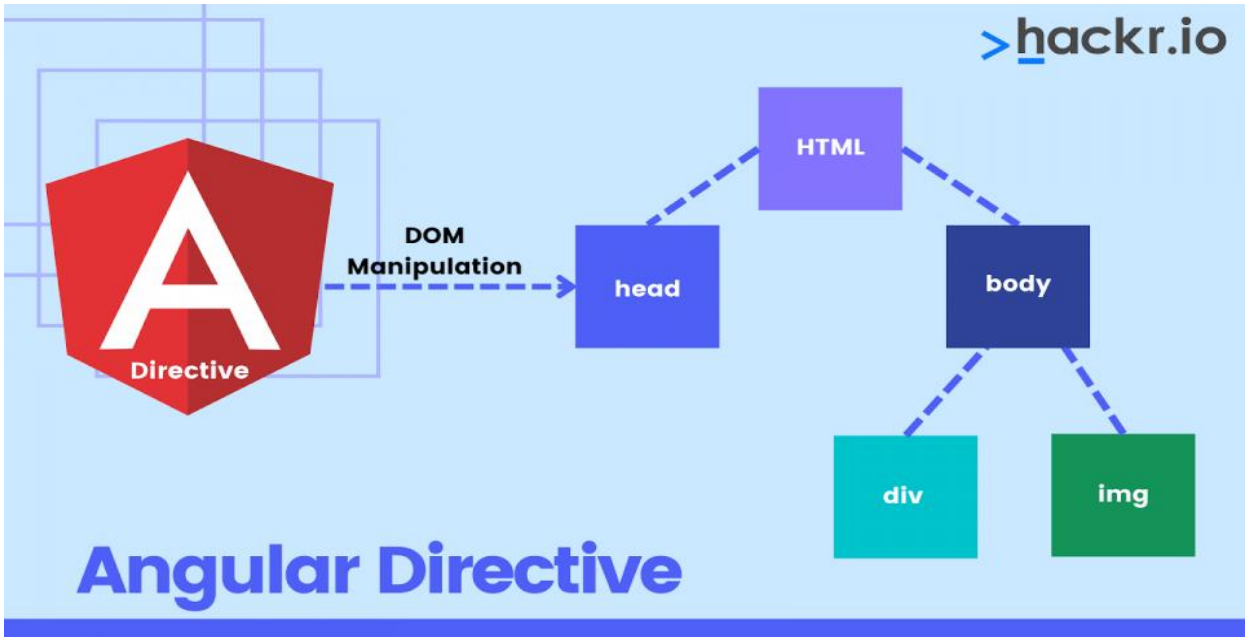
Written in HTML, templates in Angular contain Angular-specific attributes and elements. Combined with information coming from the controller and model, templates are then further rendered to cater to the user with the dynamic view.

16. Explain the difference between an Annotation and a Decorator in Angular.

In Angular, annotations are used for creating an annotation array. They are the 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.

17. What are directives in Angular?



Directives are one of the core features of Angular. They allow a developer to write new, application-specific HTML syntax. In actuality, 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

18. What are the differences between Angular and jQuery?

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 provide 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

19. Could you explain the difference between Angular expressions and JavaScript expressions?

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.

20. What is Angular Material?

It is a UI component library. Angular Material 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.

21. Why prioritize TypeScript over JavaScript in Angular?

TypeScript is a superset of Javascript as it is Javascript and extra features like typecasting for variables, annotations, and variable scope, among other things. It is purely object-oriented programming and offers a compiler that can convert it to Javascript-equivalent code.

Advanced Angular Interview Questions

22. Define the ng-content Directive.

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

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

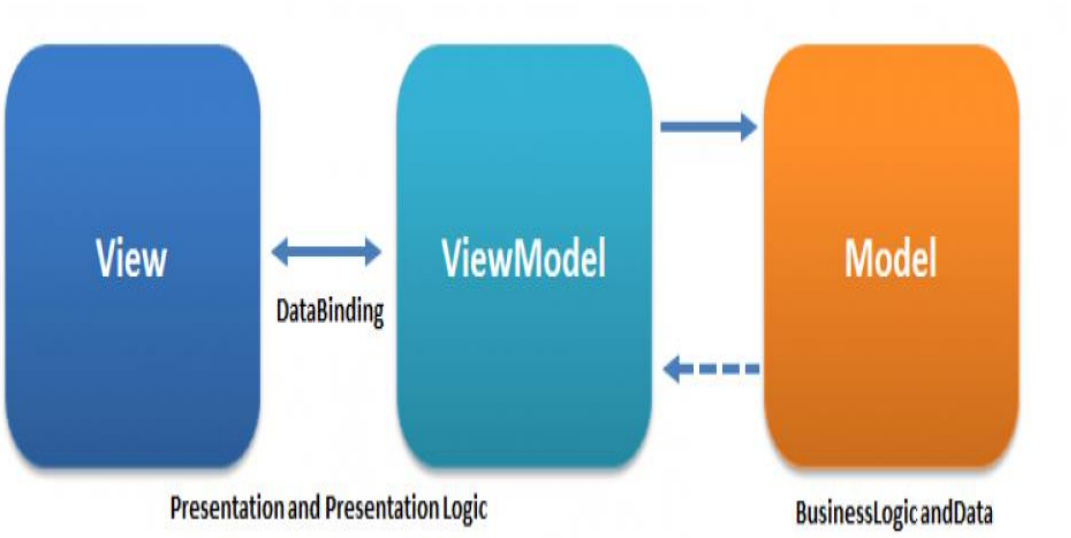
Now consider the following example of having custom text between angular tags:

```
<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.

23. Describe the MVVM architecture.



The MVVM architecture removes tight coupling between each component. It has 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, and 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.

24. Demonstrate navigating between different routes in an Angular application.

The 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>

`

})

classHeaderComponent {

 constructor(private router: Router) {}

 goHome() {

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

 }
```

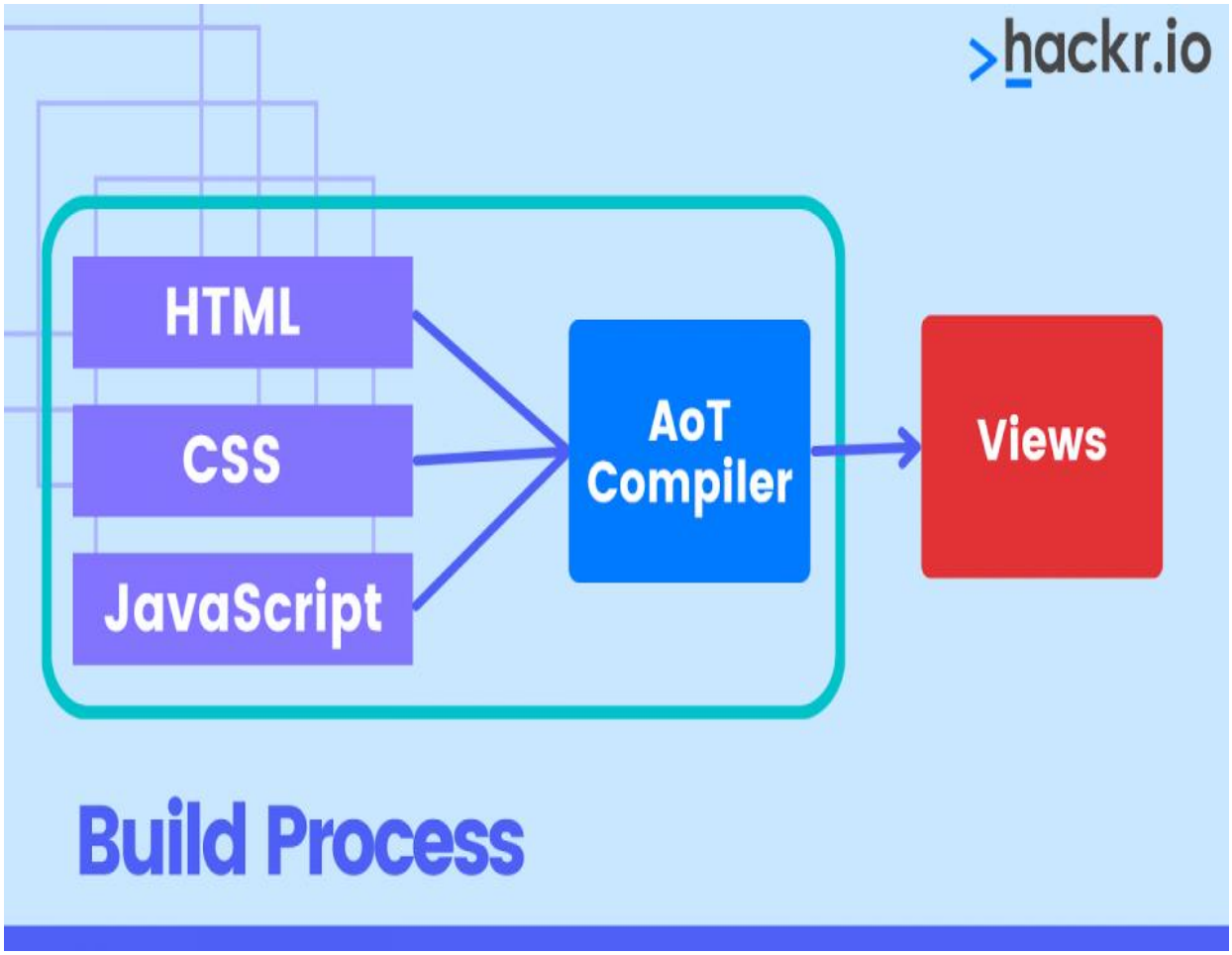
```
 goSearch() {


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


 }


}
```

25. What is the AOT (Ahead-Of-Time) Compilation? What are its advantages?



An Angular application consists of components and templates that 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 compilations 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.

26. Explain the concept of scope hierarchy in Angular.

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. The following is a typical representation of a scope hierarchy:

- Root $scope
    - $scope for Controller 1
    - $scope for Controller 2
    - …
    - ..
    - .
    - $scope for Controller n

27. How do Observables differ from Promises?

As soon as a promise 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 the passing of more than one event. A callback is made for each event in an observable.

28. What are the building blocks of Angular?

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. 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 coordinate 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 are attribute and structural directives.
5. **Metadata:** Metadata lets Angular know how to process a class.
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, known as the root module.
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 how it is meant to render the component.

29. What is Data Binding? In how many ways can it be executed?

Data binding is used to connect application data with the DOM (Data Object Model). 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 get automatically reflected in the view and vice-versa. The ngModel directive is used for achieving this type of data binding.

30. Draw a comparison between the service() and the factory() functions.

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 actuality, the factory() function is to design patterns that help in creating objects.

31. Explain the digest cycle in Angular.

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.

32. Explain the various types of filters in AngularJS.

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 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.

Some AngularJS filters are:

- **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

33. What is SPA(Single Page Application)? Contrast SPA technology with traditional web technology.

With SPA technology, only a single page - index.HTML - is maintained, although the URL keeps on changing. SPA technology is faster and easier to develop than traditional web technology.

In conventional web technology, as soon as a client requests a web page, 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.

34. What is the code for creating a decorator?

We create a decorator called Dummy:

```
functionDummy(target) {
```

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


}
```

35. What is ViewEncapsulation and how many ways are there to do it in Angular?

To put it 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 don’t spread to the component.
- **None:** Styles defined in a component are visible to all components of the application.

36. What are Lifecycle hooks in Angular? Explain some life cycle hooks.

Angular components enter their 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.

How Do You Prepare for an Angular Interview?

To prepare for an Angular interview, you should be aware of the latest standards and practices of various aspects of web development processes. This includes knowing TypeScript, JavaScript, HTML, and the application of these skills.

You also want to do research on the particular company you are applying for. This will give you a sense of what they expect, and you can prepare accordingly.

Don’t forget that actual coding and building is also very important. You should be ready to demonstrate your knowledge, and that is best practiced through hands-on work. You can look up Angular coding questions to help you practice.

1. What are basic Angular interview questions?

The basic Angular interview questions listed above cover just about all the basic concepts of the framework. These include such things as Angular’s features, and its advantages and disadvantages. Additionally, concepts such as architecture, services, and directives are also fundamental.

2. What is Ng in Angular?

Ng stands for next generation, and it relates to built-in directives in Angular. Directives are classes that add behavior to elements.

3. What are pipes in Angular?

Pipes are a feature that transforms values in Angular templates. They simply accept a value, perform a function on it, and then produce an outpu