Angular 2 Interview Questions

Anurag Gandhi, Aniket Agrawal, 4 Mar 2017

★★★★★ 4.92 (23 votes) Rate this:

A set of selected questions and answers of angular 2 that helps you to clarify the concepts of angular 2.

Introduction

Reading interview questions is one of the great way to learn and brush up the left-over concepts even if you are not preparing for the interview. In this article, we tried to touch most of the important concepts of Angular 2. We have also provided the external links/references for further reading.

Disclaimer

Reading this article does not guarantee, in any way, that you will be able to clear the interview in angular 2. Our sole purpose is to get you a reference for last minute revision along with further reading.

If you feel that some more topic needs to be covered, please let us know. We will add those in the article.

Questions

Explain the life cycle hooks of Angular 2 application

Angular 2 component/directive has lifecycle events, managed by @angular/core. It creates the component, renders it, creates and renders its children, processes changes when its data-bound properties change, and then destroys it before removing its template from the DOM. Angular provides a set of lifecycle hooks(special events) which can be tapped into this lifecycle and perform operations when required. The constructor executes prior to all lifecycle events. Each interface has a single hook method prefixed with ng. For example, *ngOnint* interface has *Oninit* method that must be implemented in the component.

Some of the events are applicable for both component/directives while few are specific to components.

- **ngOnChanges**: Responds when angular sets its data-bound property which receives the current and previous object values.
- **ngOnInit**: Initializes the component/directive after first ngOnChange triggers. This is most frequently used method to retrieve the data for the template from a back-end service.
- **ngDoCheck**: Detect and act upon changes occuring outside Angular context. It is called when every change detection run.

- **ngOnDestroy**: Cleanup just before Angular destroys the directive/component. Unsubscribe observables and detach event handlers to avoid memory leaks.

Component-specific hooks:

- **ngAfterContentInit**: Component content has been initialized
- **ngAfterContentChecked**: After Angular checks the bindings of the external content that it projected into its view.
- **ngAfterViewInit**: After Angular creates the component's view.
- **ngAfterViewChecked**: After Angular checks the bindings of the component's view.

What are the advantages of using Angular 2 over Angular 1?

1. Angular 2 is a platform not only a language:
2. Better Speed and Performance: *No $Scope in Angular 2, AOT*
3. Simpler Dependency Injection
4. Modular, cross platform
5. Benefits of ES6 and Typescript.
6. Flexible Routing with Lazy Loading Features
7. Easier to Learn

How routing works in Angular 2.

Routing is a mechanism which enables user to navigate between views/components. Angular 2 simplifies the routing and provide flexibility to configure and define at module level (Lazy loading).

The angular application has single instance of the Router service and whenever URL changes, corresponding Route is matched from the routing configuration array. On successful match, it applies redirects and the router builds a tree of ActivatedRoute objects and contains the current state of the router. Before redirection, the router will check whether new state is permitted by running guards (CanActivate). Route Guards is simply an interface method that router runs to check the route authorization. After guard runs, it will resolve the route data and activate the router state by instantiation the required components into `<router-outlet> </router-outlet>`.

Further Reading:

https://www.codeproject.com/Articles/1164813/Angular-Routing
https://vsavkin.com/angular-2-router-d9e30599f9ea#.kt4z1v957

What are Event Emitters and how it works in Angular 2?

Angular 2 doesn't have bi-directional digest cycle, unlike angular 1. In angular 2, any change occurred in the component always gets propagated from the current component to all its children in hierarchy. If the change from one component needs to be reflected to any of its parent component in hierarchy, we can emit the event by using Event Emitter api.

In short, EventEmitter is class defined in @angular/core module which can be used by components and directives to emit custom events.

```
    @output() somethingChanged = new EventEmitter();
```

We use somethingChanged.emit(value) method to emit the event. This is usually done in setter when the value is being changed in the class.

This event emit can be subscribed by any component of the module by using subscribe method.

```
myObj.somethingChanged.subscribe(val) => this.myLocalMethod(val));
```

Further Reading:

http://stackoverflow.com/questions/36076700/what-is-the-proper-use-of-an-eventemitter

https://angular.io/docs/ts/latest/api/core/index/EventEmitter-class.html

What is the use of codelyzer in angular 2 application.

All enterprise applications follows a set of coding conventions and guidelines to maintain code in better way. Codelyzer is an open source tool to run and check whether the pre-defined coding guidelines has been followed or not. Codelyzer does only static code analysis for angular and typescript project.

Codelyzer runs on top of tslint and its coding conventions are usually defined in tslint.json file. Codelyzer can be run via angular cli or npm directly. Editors like Visual Studio Code and Atom also supports codelyzer just by doing a basic settings.

To set up the codelyzer in Visual Studio code, we can go to File -> Preferences -> User Settings and add the path for tslint rules.

Hide Copy Code

```
{
  "tslint.rulesDirectory": "./node_modules/codelyzer",
  "typescript.tsdk": "node_modules/typescript/lib"
}
```

To run from cli: `ng lint`.

To run from npm: `npm run lint`

Further Reading:

https://github.com/mgechev/codelyzer

https://www.youtube.com/watch?v=bci-Z6nURgE

What is lazy loading and How to enable lazy loading in angular 2?

Most of the enterprise application contains various modules for specific business cases. Bundling whole application code and loading will be huge performance impact at initial call. Lazy lading enables us to load only the module user is interacting and keep the rest to be loaded at runtime on demand.

Lazy loading speeds up the application initial load time by splitting the code into multiple bundles and loading them on demand.

Every Angular application must have one main module say AppModule. The code should be splitted into various child modules (NgModule) based on the application business case.

Plunkr Example: Link

1. We don't require to import or declare lazily loading module in root module.
2. Add the route to top level routing (app.routing.ts) and set loadChildren. loadChildren takes absolute path from root folder followed by #{ModuleName}. RouterModule.forRoot() takes routes array and configures the router.
3. Import module specific routing in the child module.
4. In the child module routing, specify path as empty string ' ', the empty path. RouterModule.forChild again takes routes array for the child module components to load and configure router for child.
5. Then, export const routing: `ModuleWithProviders = RouterModule.forChild(routes);`

What are the security threats should we be aware of in angular 2 application?

Just like any other client side or web application, angular 2 application should also follow some of the basic guidelines to mitigate the security risks. Some of them are:

a. Avoid using/injecting dynamic Html content to your component.
b. If using external Html, that is coming from database or somewhere outside the application, sanitize it.
c. Try not to put external urls in the application unless it is trusted. Avoid url re-direction unless it is trusted.
d. Consider using AOT compilation or offline compilation.
e. Try to prevent XSRF attack by restricting the api and use of the app for known or secure environment/browsers.

Further Reading:

https://angular.io/docs/ts/latest/guide/security.html#!#best-practices

How would you optimize the angular 2 application for better performance?

Well, optimization depends on the type and size of application and many other factors. But in general, I would consider the following points while optimizing the angular 2 app:

1. Consider AOT compilation.
2. Make sure the application is bundled, uglified, and tree shaking is done.
3. Make sure the application doesn't have un-necessary import statements.
4. Make sure that any 3rd party library, which is not used, is removed from the application.
5. Have all dependencies and dev-dependencies are clearly separated.
6. I would consider lazy loading instead of fully bundled app if the app size is more.

Further Reading:

https://medium.com/@areai51/the-4-stages-of-perf-tuning-for-your-angular2-app-922ce5c1b294#.pw4m2srmr

https://www.lucidchart.com/techblog/2016/05/04/angular-2-best-practices-change-detector-performance/

How would you define custom Typings to avoid editor warnings?

Well, in most of the cases, the 3rd party library comes with its own `.d.ts` file for its type definition. In some cases, we need to extend the existing type by providing some more properties to it or if we need to define additional types to avoid Typescript warning.

If we need to extend the type definition for external library, as a good practice, we should not touch the node_modules or existing typings folder. We can create a new folder, say "custom-typings" and keep all customized type definition in that.

To define typings for application (JavaScript/Typescript) objects, we should define interfaces and entity classes in models folder in the respective module of the application.

For those cases, we can define or extend the types by creating our own "`.d.ts`" file.

Further Reading:

https://www.typescriptlang.org/docs/handbook/declaration-merging.html

https://typescript.codeplex.com/wikipage?title=Writing%20Definition%20%28.d.ts%29%20Files

http://stackoverflow.com/questions/32948271/extend-interface-defined-in-d-ts-file

What is shadow DOM? How is it helping Angular 2 to perform better?

Shadow DOM is a part of the HTML spec which allows developers to encapsulate their HTML markup, CSS styles and JavaScript. Shadow DOM, along with a few other technologies, gives developers the ability to build their own 1st class tags, web components and APIs just like the <audio> tag. Collectively, these new tags and APIs are referred to as Web Components. Shadow DOM provides better separation of concern along with lesser conflict of styles and scripts with other HTML DOM elements.

Since shadow DOM are static in nature, it's a good candidate to be cached as it is not accessible to developer. The cached DOM would be rendered faster in the browser providing better performance. Moreover, shadow DOM can be managed comparatively well while detecting the change in angular 2 application and re-paint of view can be managed efficiently.

References/Further Reading:

https://developer.mozilla.org/en-US/docs/Web/Web_Components/Shadow_DOM

https://glazkov.com/2011/01/14/what-the-heck-is-shadow-dom/

https://code.tutsplus.com/tutorials/intro-to-shadow-dom--net-34966

What is AOT compilation?

AOT compilation stands for Ahead Of Time compilation, in which the angular compiler compiles the angular components and templates to native JavaScript and HTML during the build time. The compiled Html and JavaScript is deployed to the web server so that the compilation and render time can be saved by the browser.

Advantages

1. Faster download: Since the app is already compiled, many of the angular compiler related libraries are not required to be bundled, the app bundle size get reduced. So, the app can be downloaded faster.
2. Lesser No. of Http Requests: If the app is not bundled to support lazy loading (or whatever reasons), for each associated html and css, there is a separate request goes to the server. The pre-compiled application in-lines all templates and styles with components, so the number of Http requests to the server would be lesser.
3. Faster Rendering: If the app is not AOT compiled, the compilation process happens in the browser once the application is fully loaded. This has a wait time for all necessary component to be downloaded, and then the time taken by the compiler to compile the app. With AOT compilation, this is optimized.
4. Detect error at build time: Since compilation happens beforehand, many compile time error can be detected, providing a better degree of stability of application.

Disadvantages

1. Works only with HTML and CSS, other file types need a previous build step
2. No watch mode yet, must be done manually (bin/ngc-watch.js) and compiles all the files
3. Need to maintain AOT version of bootstrap file (might not be required while using tools like cli)
4. Needs cleanup step before compiling

References/Further Reading:

https://angular.io/docs/ts/latest/cookbook/aot-compiler.html

What are the core differences between Observables and Promises?

A nice answer taken from stack overflow:

A Promise handles a **single event** when an async operation completes or fails.

Note: There are Promise libraries out there that support cancellation, but ES6 Promise doesn't so far.

An Observable is like a **Stream** (in many languages) and allows to pass zero or more events where the callback is called for each event. Often Observable is preferred over Promise because it provides the features of Promise and more. With Observable it doesn't matter if you want to handle 0, 1, or multiple events. You can utilize the same API in each case. Observable also has the advantage over Promise to be **cancelable**. If the result of an HTTP request to a server or some other expensive async operation isn't needed anymore, the Subscription of an Observable allows to cancel the subscription, while a Promise will eventually call the success or failed callback even when you don't need the notification or the result it provides anymore. Observable provides **operators** like map, forEach, reduce, ... similar to an array. There are also powerful operators like retry(), or replay(), ... that are often quite handy.

Promises vs Observables

- Promises:
    i. returns a single value
    ii. not cancellable
- Observables:
    1. works with multiple values over time
    2. cancellable
    3. supports map, filter, reduce and similar operators
    4. proposed feature for ES 2016
    5. use Reactive Extensions (RxJS)
    6. an array whose items arrive asynchronously over time

References/Further Readings:

http://stackoverflow.com/questions/36064303/what-are-the-differences-between-observables-and-promises-in-javascript

Explain local reference variables, ViewChild, and ContentChild.

Local template variables in angular2 is used to refer HTML elements and use their properties to access siblings or children.

Let's consider you have an input field named username.

`<input type="text" required ... />`

This HTMLInputField can be made available to the template using # symbol with a variable name say username.

`<input type="text" #username required ... />`

Now, this HTMLInputElement can be accessed from anywhere in the current template for example, checking validation and showing appropriate message based on the validation rule. But, username HTML reference is not accessible in the component/directive.

To access this in the component, angular 2 provides @ViewChild decorator which accepts the local reference variable.

`@ViewChild('username') username: HTMLInputElement;`

`ViewChild` element can be read after the view is initialized (`ngAfterViewInit`).

`ContentChild` is used to query the reference of the DOM within ng-content. Content Child are set before the `ngAfterContentInit` lifecycle hook.

For example:

Hide Copy Code

```
// <code>app.component.ts</code>
<my-component>
    <p #contentRef>{{test}}</p>
</ my-component >

// MyComponent.component.ts
@Component({
    selector: ‘my-component',
    template: `
    <ng-content></ng-content>
    <div> ContentChild Example </div>
})
export class LifecycleComponent implements ngAfterContentInit{
                @ContentChild(‘contentRef’)   childContent: HTMLElement;

ngAfterContentInit() {
              this.log('ngAfterContentInit');
console.log(this.childContent);
    }
}
```

Further Reading:

http://www.codelecture.com/wp/28-angular-2-_-template-access-with-local-references-viewchild-and-contentchild-w-subs/

Points of Interest

Do you think that this list is not sufficient for brushing up angular 2? We also think the same. We are working on our available time to add more richness. Please watch out this article for updates. You may also please suggest the questions that should be added here.

History

2017-02-11: First Version released.

2017-03-04: Added more questions.

License

Share

TWITTER FACEBOOK

About the Authors

Anurag Gandhi

Architect
India

Anurag Gandhi is a Developer, Consultant, Architect, Blogger, and a Speaker. He is passionate about programming.
He is extensively involved in Asp.Net Core, MVC/Web API, Microsoft Azure/Cloud, web application hosting/architecture, Angular, AngularJs, design and development. His languages of choice are C#, Node/Express, JavaScript, Asp .NET MVC, Asp, C, C++. He is familiar with many other programming languages as well. He mostly works with MS Sql Server as the preferred database and has worked with Redis, MySQL, Oracle, MS Access, etc. also.
He is active in programming communities and loves to share the knowledge with others whenever he gets the time for it.
He is also a passionate chess player.
He can be contacted at: soft.gandhi@gmail.com

Aniket Agrawal

Engineer
India

No Biography provided

You may also be interested in...

A Solution Blueprint for DevOps

Building a New Parse Server & MongoDB Atlas-Based Application

JQUERY, JSON , Angular and Less Interview questions

Beyond Arduino Create: Developing UP Squared Projects in Intel® System Studio

AngularJS Interview Questions and Answers

Wind River Helix Device Cloud Application Deployment: POC Retail Vending Machine

Comments and Discussions

You must Sign In to use this message board.

Search Comments

First Prev Next

appreciation
Aravind 25-Apr-18 1:43

Szia,

Thank you! Thank you! Thank you! Your blog was a total game changer!
In my current project, I need to develop a UI which contains 4 different dropdown or select controls data of which will be

Hide Copy Code

```
<a href="https://asha24.com/blog/angularjs-interview-questions-and-answers/"> populated from </a>
```

web api methods. I call 4 different http.get method for 4 dropdown control. But issue is that wehn I run the page, all dropdown data not populated properly. Some times, dropdown1 data bind in dropdown3. I don't understand the reason of this issue.
Thanks a lot. This was a perfect step-by-step guide. Don't think it could have been done better.
Obrigado,
Aravind

Sign In · View Thread

ngOnInit is a method and OnInit is an interface.
Member 13632977 19-Jan-18 6:50

For example, ngOnint interface has Oninit method that must be implemented in the component. This is not correct.

ngOnInit is a method you would implement in the component and OnInit is an interface you would implement.

Sign In · View Thread 5.00/5 (1 vote)

Message Closed
11-Sep-17 1:20

Message Closed

Good Work
san2debug 8-Sep-17 4:21

Good Work 😊 Thank you for sharing us

Sign In · View Thread

Preparing for an interview
pkmode 14-Mar-17 3:59

Excellent article. I would advise asking what's expected of you in the interview beforehand. In most cases you'll get a direct answer so you'll know how to prepare. Of course you can always practice Angular interview questions on your own, but it's better to ask first so that you know what to focus on.

Sign In · View Thread

Re: Preparing for an interview
Anurag Gandhi 5-Apr-17 21:26

Hi,
When I take the interview, i expect the answers based on candidate's practical exposure about the concepts. I personally focus more on logical competency than the framework feature. How much practically you have worked on the technologies helps you to learn the technology in depth.
I start with basics and switch gears based on candidates level of answers. That's my way of interviewing and may not match with other interviewers.

Focus on creating real world applications in angular 2. Many concepts will fit in naturally.

The questions and answers here are to help you re-word, revise and brush-up your concepts about Angular 2. It also helps you to validate if you have missed out any important topic and you would like to learn them in depth.

Life is a computer program and everyone is the programmer of his own life.

Sign In · View Thread

Nice Share
M,AqibShehzad 7-Mar-17 1:31

Thanks for sharing the nice stuff.

Sign In · View Thread

Re: Nice Share
Anurag Gandhi 7-Mar-17 19:21

Thank you.
We are coming up with more questions and answers soon. Please keep watching for updates.

Life is a computer program and everyone is the programmer of his own life.

Sign In · View Thread

Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin