ANGULAR 2 INTERVIEW QUESTIONS

Dear readers, these **Angular 2 Interview Questions** have been designed specially to get you acquainted with the nature of questions you may encounter during your interview for the subject of **Angular 2**. As per my experience good interviewers hardly plan to ask any particular question during your interview, normally questions start with some basic concept of the subject and later they continue based on further discussion and what you answer:

What is Angular 2?

AngularJS is a framework to build large scale and high performance web application while keeping them as easy-to-maintain. Following are the features of AngularJS framework.

- **Components** – The earlier version of Angular had a focus of Controllers but now has changed the focus to having components over controllers. Components help to build the applications into many modules. This helps in better maintaining the application over a period of time.
- **TypeScript** – The newer version of Angular is based on TypeScript. This is a superset of JavaScript and is maintained by Microsoft.
- **Services** – Services are a set of code that can be shared by different components of an application. So for example if you had a data component that picked data from a database, you could have it as a shared service that could be used across multiple applications.

What are the key components of Angular 2?

Angular 2 has the following components –

- **Modules** – This is used to break up the application into logical pieces of code. Each piece of code or module is designed to perform a single task.
- **Component** – This can be used to bring the modules together.
- **Templates** – This is used to define the views of an Angular JS application.
- **Metadata** – This can be used to add more data to an Angular JS class.
- **Service** – This is used to create components which can be shared across the entire application.

Explain Modules in Angular 2.

Modules are used in Angular JS to put logical boundaries in your application. Hence, instead of coding everything into one application, you can instead build everything into separate modules to separate the functionality of your application. A module is made up of the following parts –

- **Bootstrap array** – This is used to tell Angular JS which components need to be loaded so that its functionality can be accessed in the application. Once you include the component in the bootstrap array, you need to declare them so that they can be used across other components in the Angular JS application.
- **Export array** – This is used to export components, directives, and pipes which can then be used in other modules.

- **Import array** – Just like the export array, the import array can be used to import the functionality from other Angular JS modules.

Explain Components in Angular 2.

Each application consists of Components. Each component is a logical boundary of functionality for the application. You need to have layered services, which are used to share the functionality across components.Following is the anatomy of a Component. A component consists of –

- **Class** – This is like a C or Java class which consists of properties and methods.
- **Metadata** – This is used to decorate the class and extend the functionality of the class.
- **Template** – This is used to define the HTML view which is displayed in the application.

What are Angular 2 directives? Explain with examples.

A directive is a custom HTML element that is used to extend the power of HTML. Angular 2 has the following directives that get called as part of the BrowserModule module.

- **ngIf** –

  The **ngif** element is used to add elements to the HTML code if it evaluates to true, else it will not add the elements to the HTML code.

  ### Syntax

  ```
  *ngIf = 'expression'
  ```

  If the expression evaluates to true then the corresponding gets added, else the elements are not added.

- **ngFor** –

  The **ngFor** element is used to elements based on the condition of the For loop.

  ### Syntax

  ```
  *ngFor = 'let variable of variablelist'
  ```

  The variable is a temporary variable to display the values in the **variablelist**.

How will you handle errors in Angular 2 applications?

Angular 2 applications have the option of error handling. This is done by including the ReactJS catch library and then using the catch function.

- The catch function contains a link to the Error Handler function.
- In the error handler function, we send the error to the console. We also throw the error back to the main program so that the execution can continue.
- Now, whenever you get an error it will be redirected to the error console of the browser.

What is routing?

Routing helps in directing users to different pages based on the option they choose on the main page. Hence, based on the option they choose, the required Angular Component will be rendered to the user.

What is CLI?

Command Line Interface *CLI* can be used to create our Angular JS application. It also helps in creating a unit and end-to-end tests for the application.

What is Dependency Injection? Explain with example.

Dependency injection is the ability to add the functionality of components at runtime. Let's take a look at an example and the steps used to implement dependency injection.

Step 1 – Create a separate class which has the injectable decorator. The injectable decorator allows the functionality of this class to be injected and used in any Angular JS module.

```
@Injectable()
   export class classname {
}
```

Step 2 – Next in your appComponent module or the module in which you want to use the service, you need to define it as a provider in the @Component decorator.

```
@Component ({
   providers : [classname]
})
```

Explain tsconfig.json file.

This file is used to give the options about TypeScript used for the Angular JS project.

```
{
   "compilerOptions": {
      "target": "es5",
      "module": "commonjs",
      "moduleResolution": "node",
      "sourceMap": true,
      "emitDecoratorMetadata": true,
      "experimentalDecorators": true,
      "lib": [ "es2015", "dom" ],
      "noImplicitAny": true,
      "suppressImplicitAnyIndexErrors": true
   }
}
```

Following are some key points to note about the above code.

- The target for the compilation is es5 and that is because most browsers can only understand ES5 typescript.
- The sourceMap option is used to generate Map files, which are useful when debugging. Hence, during development it is good to keep this option as true.
- The "emitDecoratorMetadata": true and "experimentalDecorators": true is required for Angular JS decorators. If not in place, Angular JS application will not compile.

Explain package.json file.

This file contains information about Angular 2 project. Following are the typical settings in the file.

```
{
   "name": "angular-quickstart",
   "version": "1.0.0",
   "description": "QuickStart package.json from the documentation,
      supplemented with testing support",

   "scripts": {
      "build": "tsc -p src/",
      "build:watch": "tsc -p src/ -w",
      "build:e2e": "tsc -p e2e/",
      "serve": "lite-server -c=bs-config.json",
      "serve:e2e": "lite-server -c=bs-config.e2e.json",
      "prestart": "npm run build",
      "start": "concurrently \"npm run build:watch\" \"npm run serve\"",
      "pree2e": "npm run build:e2e",
      "e2e": "concurrently \"npm run serve:e2e\" \"npm run protractor\" --killothers --success
first",
      "preprotractor": "webdriver-manager update",
      "protractor": "protractor protractor.config.js",
      "pretest": "npm run build",
      "test": "concurrently \"npm run build:watch\" \"karma start karma.conf.js\"",
      "pretest:once": "npm run build",
      "test:once": "karma start karma.conf.js --single-run",
      "lint": "tslint ./src/**/*.ts -t verbose"
   },

   "keywords": [],
   "author": "",
   "license": "MIT",
   "dependencies": {
      "@angular/common": "<2.4.0",
      "@angular/compiler": "<2.4.0",
      "@angular/core": "<2.4.0",
      "@angular/forms": "<2.4.0",
      "@angular/http": "<2.4.0",
      "@angular/platform-browser": "<2.4.0",
      "@angular/platform-browser-dynamic": "<2.4.0",
      "@angular/router": "<3.4.0",
      "angular-in-memory-web-api": <0.2.4",
      "systemjs": "0.19.40",
      "core-js": "^2.4.1",
      "rxjs": "5.0.1",
      "zone.js": "^0.7.4"
   },

   "devDependencies": {
      "concurrently": "^3.2.0",
      "lite-server": "^2.2.2",
      "typescript": "<2.0.10",
      "canonical-path": "0.0.2",
      "tslint": "^3.15.1",
      "lodash": "^4.16.4",
      "jasmine-core": "<2.4.1",
      "karma": "^1.3.0",
      "karma-chrome-launcher": "^2.0.0",
      "karma-cli": "^1.0.1",
      "karma-jasmine": "^1.0.2",
      "karma-jasmine-html-reporter": "^0.2.2",
      "protractor": <4.0.14",
      "rimraf": "^2.5.4",
      "@types/node": "^6.0.46",
      "@types/jasmine": "2.5.36"
   },
   "repository": {}
}
```

Some key points to note about the above code –

- There are two types of dependencies, first is the dependencies and then there are dev dependencies. The dev ones are required during the development process and the others are needed to run the application.
- The "build:watch": "tsc -p src/ -w" command is used to compile the typescript in the background by looking for changes in the typescript files.

Explain systemjs.config.json file.

This file contains the system files required for Angular JS application. This loads all the necessary script files without the need to add a script tag to the html pages. The typical files will have the following code.

```
/**
 * System configuration for Angular samples
 * Adjust as necessary for your application needs.
*/
(function (global) {
   System.config({
      paths: {
         // paths serve as alias
         'npm:': 'node_modules/'
      },

      // map tells the System loader where to look for things
      map: {
         // our app is within the app folder
         app: 'app',

         // angular bundles
         '@angular/core': 'npm:@angular/core/bundles/core.umd.js',
         '@angular/common': 'npm:@angular/common/bundles/common.umd.js',
         '@angular/compiler': 'npm:@angular/compiler/bundles/compiler.umd.js',
         '@angular/platform-browser': 'npm:@angular/platformbrowser/bundles/platform-
browser.umd.js',
         '@angular/platform-browser-dynamic': 'npm:@angular/platform-
browserdynamic/bundles/platform-browser-dynamic.umd.js',
         '@angular/http': 'npm:@angular/http/bundles/http.umd.js',
         '@angular/router': 'npm:@angular/router/bundles/router.umd.js',
         '@angular/forms': 'npm:@angular/forms/bundles/forms.umd.js',

         // other libraries
         'rxjs':  'npm:rxjs',
         'angular-in-memory-web-api': 'npm:angular-in-memory-web-api/bundles/inmemory-web-
api.umd.js'
      },

      // packages tells the System loader how to load when no filename and/or no extension
      packages: {
         app: {
            defaultExtension: 'js'
         },
         rxjs: {
            defaultExtension: 'js'
         }
      }
   });
})(this);
```

Some key points to note about the above code –

- 'npm:': 'node_modules/' tells the location in our project where all the npm modules are located.
- The mapping of app: 'app' tells the folder where all our applications files are loaded.

Explain app.module.ts file.

The following code will be present in the **app.module.ts** file.

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent }  from './app.component';

@NgModule({
   imports:      [ BrowserModule ],
   declarations: [ AppComponent ],
   bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Let's go through each line of the code in detail.

- The import statement is used to import functionality from the existing modules. Thus, the first 3 statements are used to import the NgModule, BrowserModule and AppComponent modules into this module.
- The NgModule decorator is used to later on define the imports, declarations, and bootstrapping options.
- The BrowserModule is required by default for any web based angular application.
- The bootstrap option tells Angular which Component to bootstrap in the application.

How will you convert an input to all lowercase?

lowercase filter is used to convert the input to all lowercase.

In below example, we've added lowercase filter to an expression using pipe character. Here we've added lowercase filter to print student name in all lowercase letters.

```
<div>
   The name of this Tutorial is {{TutorialName}}

   The first Topic is {{appList[0] | lowercase}}

   The second Topic is {{appList[1] | lowercase}}

   The third Topic is {{appList[2]| lowercase}}

</div>
```

How will you convert an input to all uppercase?

uppercase filter is used to convert the input to all uppercase.

In below example, we've added uppercase filter to an expression using pipe character. Here we've added uppercase filter to print student name in all uppercase letters.

```
<div>
   The name of this Tutorial is {{TutorialName}}

   The first Topic is {{appList[0] | uppercase}}

   The second Topic is {{appList[1] | uppercase}}

   The third Topic is {{appList[2]| uppercase}}

</div>
```

How will you get a substring from a string?

slice filter is used to slice a piece of data from the input string.

In below example, we've added slice filter to an expression using pipe character. Here property value will be sliced based on the start and end positions.

```
<div>
   The name of this Tutorial is {{TutorialName}}

   The first Topic is {{appList[0] | slice:1:2}}

   The second Topic is {{appList[1] | slice:1:3}}

   The third Topic is {{appList[2]| slice:2:3}}

</div>
```

How will you convert a string into a date?

date filter is used to convert the input string to date format.

In below example, we've added date filter to an expression using pipe character. Here property value will be converted to date format.

```
<div>
   The date of this Tutorial is {{newdate | date:"MM/dd/yy"}}

</div>
```

How will you convert a string into a currency?

currency filter is used to convert the input string to currency format.

In below example, we've added currency filter to an expression using pipe character. Here property value will be converted to currency format.

```
<div>
   The currency of this Tutorial is {{newValue | currency}}

</div>
```

How will you convert a string into a percentage?

percent filter is used to convert the input string to percentage format.

In below example, we've added percent filter to an expression using pipe character. Here property value will be converted to percentage format.

```
<div>
   The percentage of this Tutorial is {{newValue | percent}}

</div>
```

When ngOnChanges event get called in Angular 2 Application Lifecycle?

When the value of a data bound property changes, then this method is called.

When ngOnInit event get called in Angular 2 Application Lifecycle?

This is called whenever the initialization of the directive/component after Angular first displays the data-bound properties happens.

When ngDoCheck event get called in Angular 2 Application Lifecycle?

This is for the detection and to act on changes that Angular can't or won't detect on its own.

When ngAfterContentInit event get called in Angular 2 Application Lifecycle?

This is called in response after Angular projects external content into the component's view.

When ngAfterContentChecked event get called in Angular 2 Application Lifecycle?

This is called in response after Angular checks the content projected into the component.

When ngAfterViewInit event get called in Angular 2 Application Lifecycle?

This is called in response after Angular initializes the component's views and child views.

When ngAfterViewChecked event get called in Angular 2 Application Lifecycle?

This is called in response after Angular checks the component's views and child views.

When ngOnDestroy event get called in Angular 2 Application Lifecycle?

This is the cleanup phase just before Angular destroys the directive/component.