1. What is Angular?

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

2. What are the key components of Angular?

Angular 2 and earlier versions have 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.
- **Templates** − This is used to define the views of an Angular application.
- **Directive -** it can be used to extend HTML with new attributes
- **Pipe** - it helps transform values in Angular template
- **Service** − This is used to create components which can be shared across the entire application.

3. Explain Modules in Angular

Modules are used in Angular 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 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 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 modules.

4. Explain Components in Angular

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.

5. What are Angular 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**.

6. How will you handle HTTP errors in Angular applications?

You can work with HTTP errors if you attach a “catch” to your request. For example:

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { HttpClient } from '@angular/common/http';
```

```
import 'rxjs/add/observable/throw';

@Injectable()
export class Client {

  constructor(
    public http: HttpClient
  ) {}

  public fetch() {
    return this.http.post('https://thisurliswrong123123.com', {})
      .catch((err) => {

          // Do messaging and error handling here

          return Observable.throw(err)
      });
  }
}
```

You should use HTTP Interceptor to route all request properly. And you can automatically add or remove properties to each request.

7. 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.

8. What is CLI?

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

9. What is Dependency Injection?

Dependency injection is an app design pattern. Angular provides a developer with its own DI framework and it is used to increase the efficiency of Angular apps. Dependencies are services or objects that a class needs to perform its function. DI is a coding pattern in which a class asks for dependencies from external sources rather than creating them itself (Angular.io).

10. Explain tsconfig.json file.

This file is used to give the options about TypeScript used for the Angular 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 decorators. If not in place, Angular application will not compile.

11. 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.

12. 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.