**Novice Level**

### **Familiarity of Basic Terminology**

### 1)What are the differences between AngularJS (angular 1.x) and Angular (Angular 2.x and beyond)?

**Architecture**

Version 1.0 vs. Version 2.0

Angular 2.0 shows a substantial change in the structure as compared to version 1.0. The architecture of Angular v1 is based on MVC whereas the architecture of Angular v2 is based on service/controller. There is very less possibility to upgrade the Angular v1 to v2, mainly developers have to rewrite the entire application code.

Version 2.0 vs. Version 4.0

The upgrade of the version from 2.0 to 4.0 has reduced it’s bundled file size by 60%. The code generated is reduced and has accelerated the application development. Here the developed code can be used for prod mode and debug.

**JavaScript and TypeScript**

v 1.0 vs. v 2.0

Angular v1.0 use JavaScript to build the application while Angular v2.0 uses the Typescript to write the application. TypeScript is a superset of JavaScript which helps to build more robust and structured code. Dart can be used by developers along with TypeScript in version 2.0.

v 2.0 vs. v 4.0

Angular v4.0 is compatible with newer versions TypeScript 2.1 and TypeScript 2.2. This helps with better type checking and also enhanced IDE features for Visual Studio Code.

**Mobile Support**

Angular 2.0 has made it possible to accomplish the native applications for a mobile platform like React Native. Angular 2.0 gives us the two layers: application layer and the rendering layer. As need, any view can be rendered in runtime for the required component.

**Component-based UI**

1.0 vs. 2.0

The controller concept which was present in Angular v1.0 is eliminated in Angular v2.0. Angular v2.0 has changed to component based UI. This helps a developer to divide the applications in terms of components with desired features and enable to call required UI. These have helped to improve the flexibility and reusability as compared to Angular v1.0

**SEO Friendly**

1.0 vs. 2.0

With Angular v1.0 developing the search engine friendly Single Page Applications was the major difficulty. But this bottleneck was eliminated in Angular v2.0. [**AngularJS development services**](https://www.angularminds.com/angularjs-development-company.html) build SEO friendly Single Page Applications by rendering the HTML at the server side.

2) What is a component? Why would you use it? What is the minimum definition of a component?

A Component is a special kind of [directive](https://docs.angularjs.org/guide/directive) that uses a simpler configuration which is suitable for a component-based application structure. This makes it easier to write an app in a way that's similar to using Web Components or using the new Angular's style of application architecture.

Advantages of Components:

* simpler configuration than plain directives
* promote sane defaults and best practices
* optimized for component-based architecture
* writing component directives will make it easier to upgrade to Angular

3) What is a module, and what does it contain?

You can think of a module as a container for the different parts of your app – controllers, services, filters, directives, etc. Most applications have a main method that instantiates and wires together the different parts of the application. AngularJS apps don't have a main method. Instead modules declaratively specify how an application should be bootstrapped. There are several advantages to this approach:

* The declarative process is easier to understand.
* You can package code as reusable modules.
* The modules can be loaded in any order (or even in parallel) because modules delay execution.
* Unit tests only have to load relevant modules, which keeps them fast.
* End-to-end tests can use modules to override configuration.

4) What is a service, and when will you use it?

AngularJS services are substitutable objects that are wired together using [dependency injection (DI)](https://docs.angularjs.org/guide/di). You can use services to organize and share code across your app.

AngularJS services are:

* Lazily instantiated – AngularJS only instantiates a service when an application component depends on it.
* Singletons – Each component dependent on a service gets a reference to the single instance generated by the service factory.

5) What is a promise? Explain it laymen's terms.

A promise is a special type of Object that we can either use, or construct ourselves to handle asynchronous tasks. We deem them promises because we are “promised” a result at a future point in time. For example an HTTP call could complete in 200ms or 400ms, a promise will execute when resolved.

A promise has three states, pending, resolved or rejected. Using $q in Angular, we can construct our own promises, however let’s look at the ES2015 Promise Object first to get familiar on how to create one.

6) What are the lifecycle hooks for components and directives?

Lifecycle hooks are simply functions that get called at specific points of a component’s life in our Angular apps. They landed in AngularJS 1.5 and are to be used alongside the [.component() method](https://toddmotto.com/exploring-the-angular-1-5-component-method/), and have slowly evolved over the last few versions to include some more powerful (and Angular v2+ inspired) hooks.

**constructor**

This is invoked when Angular creates a component or directive by calling new on the class.

**ngOnChanges**

Invoked **every** time there is a change in one of the *input* properties of the component.

**ngOnInit**

Invoked when given component has been initialized. This hook is only called **once** after the first ngOnChanges

**ngDoCheck**

Invoked when the change detector of the given component is invoked. It allows us to implement our own change detection algorithm for the given component.

**ngOnDestroy**

This method will be invoked just before Angular destroys the component. Use this hook to unsubscribe observables and detach event handlers to avoid memory leaks.

7) What are pipes? Give me an example.

Pipes allow us to change data inside of a template; i.e. filtering, ordering, formatting dates, numbers, currencies, etc. A quick example is you can transfer a string to lowercase by applying a simple filter in the template code.

Pipes are chaining, so you can use multiple combination of pipes. Example:

<p>{{ cars | **slice**:0:2 | **json** }}</p>

8) What are the differences between reactive forms and template driven forms?

**Template driven** form is something when we write logic, validation, controls everything in the template part I mean in html code. The template is responsible for setting up the form, the validations etc

Whereas In case of **Reactive forms** all the logic validation part in the controller side I mean in typescript file/class. For the reactive approach you are doing all that in code in the component. A far less noisy approach

Template Driven Forms Features

* Easier to use
* Suitable for simple scenarios
* Uses Two-way data binding (using [(NgModel)] syntax)
* Minimal component code
* Automatic track of the form and its data (handled by Angular)
* Unit testing is challenge

Reactive Forms Features

* Easier Unit testing
* More flexible, So Handles any complex scenarios.
* Reactive transformations can be made possible such as Adding elements dynamically
* No data binding is done (Immutable data model preferred by most developers)
* More component code and less HTML markup

9) What is a dumb, or presentation, component? What are the benefits of using dumb components?

Presentational (aka Dumb) components are usually only responsible for displaying data based on their Inputs and communicating with their parent through Outputs. Therefore, they are much easier to test.