**Basic AngularJS Interview Questions and Answers for Front-end Web Developers**

AngularJS is widely known Javascript framework. If you are a front-end web developer preparing for AngularJS interview, following basic AngularJS interview questions and answers might help you little bit. Before going to AngularJS interview, you should know basic concepts and architecture of AngularJS, key features of AngularJS, how AngularJS is different from jQuery or other Javascript frameworks etc. Following basic AngularJS interview questions and answers will give you little insight of these concepts.

**1. Why is this project called "AngularJS"? Why is the namespace called "ng"?**

Because HTML has Angular brackets and "ng" sounds like "Angular".

**2. Is AngularJS a library, framework, plugin or a browser extension?**

AngularJS fits the definition of a framework the best, even though it's much more lightweight than a typical framework and that's why many confuse it with a library.

AngularJS is 100% JavaScript, 100% client side and compatible with both desktop and mobile browsers. So it's definitely not a plugin or some other native browser extension.

**3. Why to choose AngularJS Javascript Framework for front-end web development?**

AngularJS is quickly becoming the dominant JavaScript framework for professional web development. With the growth and strength of HTML5 and the increasing performance in modern browsers, many JavaScript frameworks have been created to help develop rich client applications. These frameworks/libraries have given developers a huge toolkit to build enterprise complexity into client-side applications. Server side frameworks are becoming a thing of the past and being replaced with applications written in Backbone, Ember, AngularJS, Knockout, etc.

So why am I talking about AngularJS over frameworks/libraries like Backbone, Ember, or Knockout?

For me, the major points of separation in AngularJS’s favor are the following:

1. Good documentation

2. Write less code to do more

3. Backed by Google

4. Good developer community

5. Simple Data-Binding

6. Small footprint

If you’re looking for a robust, well-maintained framework for any sized project, I strongly recommend that you take a look at AngularJS. It can be downloaded for free at AngularJS.org, which also contains a wealth of information, including the full API documentation, as well as numerous examples and tutorials that cover every facet of front-end web development. Following are some reasons why to choose Angular JS Javascript Framework for front-end web development?

**1. Angular JS Framework is developed by Google**

Angular is built and maintained by dedicated Google engineers. This one may seem obvious, but it’s important to remember that many (not all) frameworks are made by hobbyists in the open source community. While passion and drive have forged frameworks, like Cappucino and Knockout, Angular is built and maintained by dedicated (and highly talented) Google engineers. This means you not only have a large open community to learn from, but you also have skilled, highly-available engineers tasked to help you get your Angular questions answered.

This isn’t Google’s first attempt at a JavaScript framework; they first developed their comprehensive Web Toolkit, which compiles Java down to JavaScript, and was used by the Google Wave team extensively. With the rise of HTML5, CSS3, and JavaScript, as both a front-end and back-end language, Google realized that the web was not meant to be written purely in Java.

AngularJS came about to standardize web application structure and provide a future template for how client-side apps should be developed.

Angular JS is being used by a host of applications, ranging from hobby to commercial products. Adoption of AngularJS as a viable framework for client-side development is quickly becoming known to the entire web development community.

Because AngularJS is built by Google, you can be sure that you’re dealing with efficient and reliable code that will scale with your project. If you’re looking for a framework with a solid foundation, Angular is your choice!

**2. Angular JS is equipped with a lot of features**

If you’re familiar with projects, like QUnit, Mocha or Jasmine, then you’ll have no trouble learning Angular’s unit-testing API.

Angular, similar to Backbone or JavaScriptMVC, is a complete solution for rapid front-end development. No other plugins or frameworks are necessary to build a data-driven web application. Here’s an overview of Angular’s stand-out features:

**A) REST Easy.** RESTful actions are quickly becoming the standard for communicating from the server to the client. In one line of JavaScript, you can quickly talk to the server and get the data you need to interact with your web pages. AngularJS turns this into a simple JavaScript object, as Models, following the MVVM (Model View View-Model) pattern.

**B) MVVM to the Rescue!** Models talk to ViewModel objects (through something called the $scope object), which listen for changes to the Models. These can then be delivered and rendered by the Views, which is the HTML that expresses your code. Views can be routed using the $routeProvider object, so you can deep-link and organize your Views and Controllers, turning them into navigable URLs. AngularJS also provides stateless controllers, which initialize and control the $scope object.

**C) Data Binding and Dependency Injection.** Everything in the MVVM pattern is communicated automatically across the UI whenever anything changes. This eliminates the need for wrappers, getters/setters or class declarations. AngularJS handles all of this, so you can express your data as simply as with JavaScript primitives, like arrays, or as complex as you wish, through custom types. Since everything happens automatically, you can ask for your dependencies as parameters in AngularJS service functions, rather than one giant main() call to execute your code.

**D) Extends HTML.** Most websites built today are a giant series of <div> tags with little semantic clarity. You need to create extensive and exhaustive CSS classes to express the intention of each object in the DOM. With Angular, you can operate your HTML like XML, giving you endless possibilities for tags and attributes. Angular accomplishes this, via its HTML compiler and the use of directives to trigger behaviors based on the newly-created syntax you write.

**E) Makes HTML your Template**. If you’re used to Mustache or Hogan.js, then you can quckly grasp the bracket syntax of Angular’s templating engine, because it’s just HTML. Angular traverses the DOM for these templates, which house the directives mentioned above. The templates are then passed to the AngularJS compiler as DOM elements, which can be extended, executed or reused. This is key, because, now, you have raw DOM components, rather than strings, allowing for direct manipulation and extension of the DOM tree.

**F) Enterprise-level Testing.** As stated above, AngularJS requires no additional frameworks or plugins, including testing. If you’re familiar with projects, like QUnit, Mocha or Jasmine, then you’ll have no trouble learning Angular’s unit-testing API and Scenario Runner, which guides you through executing your tests in as close to the actual state of your production application as possible.

These are the fundamental principles that guide AngularJS to creating an efficient, performance-driven, and maintainable front-end codebase. As long as you have a source for storing data, AngularJS can do all of the heavy lifting on the client, while providing a rich, fast experience for the end user.

**4. What are the key features of AngularJS?**

**Scope**

The job of the Scope is to detect changes to model objects and create an execution context for expressions. There is one root scope for the application (ng-app) with hierarchical children scopes. It marshals the model to the view and forwards events to the controller.

**Controller**

The Controller is responsible for construction of the model and connects it to the view (HTML). The scope sits between the controller and the view. Controllers should be straightforward and simply contain the business logic needed for a view. Generally you want thin controllers and rich services. Controllers can be nested and handle inheritance. The big difference in AngularJS from the other JavaScript frameworks is there is no DOM manipulation in controllers. It is something to unlearn when developing in AngularJS.

**Model**

In AngularJS, a Model is simply a JavaScript object. No need to extend anything or create any structure. This allows for nested models  - something that Backbone doesn’t do out-of-the-box.

**View**

The View is based on DOM objects, not on strings. The view is the HTML. HTML is declarative – well suited for UI design. The View should not contain any functional behavior. The flexibility here is to allow for multiple views per Controller.

**Services**

The Services in AngularJS are singletons that perform common tasks for web applications. If you need to share common functionality between Controllers, then use Services. Built-in AngularJS, Services start with a $. There are several ways to build a service: Service API, Factory API, or the $provide API.

**Data Binding**

Data Binding in AngularJS is a two-way binding between the View and the Model. Automatic synchronizing between views and data models makes this really easy (and straightforward) to use. Updating the model is reflected in View without any explicit JavaScript code to bind them together, or to add event listeners to reflect data changes.

**Directives**

Now this is cool. AngularJS allows you to use Directives to transform the DOM or to create new behavior. A directive allows you to extend the HTML vocabulary in a declarative fashion. The ‘ng’ prefix stands for built-in AngularJS directives. The App (ng-app), Model (ng-model), the Controller (ng-controller), etc. are built into the framework. AngularJS allows for building your own directives. Building directives is not extremely difficult, but not easy either. There are different things that can be done with them. Please check out AngularJS’s documentation on directives.

**Filters**

The Filters in AngularJS perform data transformation. They can be used to do formatting (like I did in my Directives example with padding zeros), or they can be used to do filter results (think search).

**Validation**

AngularJS has some built-in validation around HTML5 input variables (text, number, URL, email, radio, checkbox) and some directives (required, pattern, minlength, maxlength, min, max). If you want to create your own validation, it is just as simple as creating a directive to perform your validation.

**Testable**

Testing is a big concern for enterprise applications. There are several different ways to write and run tests against JavaScript code, thus against AngularJS. The developers at AngularJS advocate using Jasmine tests ran using Testacular. I have found this method of testing very straightforward and, while writing tests may not be the most enjoyable, it is just as importable as any other piece of developing an application.

**5. What is a scope in AngularJS?**

scope is an object that refers to the application model. It is the glue between application controller and the view. Both the controllers and directives have reference to the scope, but not with each other. It is an execution context for expressions and arranged in hierarchical structure. Scopes can watch expressions and propagate events.

**6. Can you explain the concept of scope hierarchy? How many scopes can an application have?**

Each Angular application has exactly one root scope, but may have several child scopes. The application can have multiple scopes, because child controllers and some directives create new child scopes. When new scopes are created, they are added as children of their parent scope. This creates a hierarchical structure similar to the DOM where they're attached.

When Angular evaluates a bound variable like say {{firstName}}, it first looks at the scope associated with the given element for the firstName property. If no such property is found, it searches the parent scope and so on until the root scope is reached. In JavaScript this behaviour is known as prototypical inheritance, and child scopes prototypically inherit from their parents. The reverse is not true. i.e. the parent can't see it's children's bound properties.

**7. Is AngularJS a templating system?**

At the highest level, Angular does look like a just another templating system. But there is one important reason why the Angular templating system is different, that makes it very good fit for application development: bidirectional data binding. The template is compiled in the browser and the compilation step produces a live view. This means you, the developers, don't need to write code to constantly sync the view with the model and the model with the view as in other templating systems.

**8. Do I need to worry about security holes in AngularJS?**

Like any other technology, AngularJS is not impervious to attack. Angular does, however, provide built-in protection from basic security holes including cross-site scripting and HTML injection attacks. AngularJS does round-trip escaping on all strings for you and even offers XSRF protection for server-side communication.

AngularJS was designed to be compatible with other security measures like Content Security Policy (CSP), HTTPS (SSL/TLS) and server-side authentication and authorization that greatly reduce the possible attack vectors and we highly recommended their use.

**9. What's Angular's performance like?**

The startup time heavily depends on your network connection, state of the cache, browser used and available hardware, but typically we measure bootstrap time in tens or hundreds of milliseconds.

The runtime performance will vary depending on the number and complexity of bindings on the page as well as the speed of your backend (for apps that fetch data from the backend). Just for an illustration we typically build snappy apps with hundreds or thousands of active bindings.

**10. Does Angular use the jQuery library?**

Yes, Angular can use jQuery if it's present in your app when the application is being bootstrapped. If jQuery is not present in your script path, Angular falls back to its own implementation of the subset of jQuery that we call jQLite.

Due to a change to use on()/off() rather than bind()/unbind(), Angular 1.2 only operates with jQuery 1.7.1 or above.

**11. What are the key differences between AngularJS and jQuery?**

[Read complete answer](http://theprofessionalspoint.blogspot.in/2014/02/angularjs-vs-jquery-difference-between.html)

**12. How will you compare AngularJS with other Javascript frameworks like Ember and Backbone?**  
[Read complete answer](http://theprofessionalspoint.blogspot.in/2014/02/angularjs-vs-ember-vs-backbone-which.html)  
 **13. How will you display different images based on the status being red, amber, or green?**

Use the ng-switch and ng-switch-when directives as shown below.

<div ng-switch on="account.status">

 <div ng-switch-when="AMBER">

  <img class="statusIcon"

   src='apps/dashboard/amber-dot.jpg' />

 </div>

 <div ng-switch-when="GREEN">

  <img class="statusIcon"

   src='apps/dashboard/green-dot.jpg' />

 </div>

 <div ng-switch-when="RED">

  <img class="statusIcon"

   src='apps/dashboard/red-dot.jpg' />

 </div>

</div>

**14. How will you initialize a select box with options on page load?**

Use the ng-init directive.

<div ng-controller="apps/dashboard/account" ng-switch

 on="!!accounts" ng-init="loadData()">

**15. How will you show/hide buttons and enable/disable buttons conditionally?**

Using the ng-show and ng-disabled directives.

<div class="dataControlPanel"

    ng-show="accounts.releasePortfolios">

    <div class="dataControlButtons">

     <button class="btn btn-primary btn-small"

      ng-click="saveComments()" ng-disabled="disableSaveButton">Save</button>

     <button class="btn btn-primary btn-small"

      ng-click="releaseRun()" ng-disabled="disableReleaseButton">Release</button>

    </div>

</div>

**16. How will you loop through a collection and list each item?**

Using the ng-repeat directive.

<table

  class="table table-bordered table-striped table-hover table-fixed-head portal-data-table">

  <thead>

   <tr>

    <th>account</th>

    <th>Difference</th>

    <th>Status</th>

   </tr>

  </thead>

  <tbody>

   <tr

    ng-repeat="account in acounts">

    <td width="40%">{{account.accountCode}}</td>

    <td width="30%" style="text-align: right">{{account.difference

     | currency: ""}}</td>

    <td width="30%">

     <div ng-switch on="account.status">

      <div ng-switch-when="AMBER">

       <img class="statusIcon"

        src='apps/dashboard/amber-dot.jpg' />

      </div>

      <div ng-switch-when="GREEN">

       <img class="statusIcon"

        src='apps/dashboard/green-dot.jpg' />

      </div>

      <div ng-switch-when="RED">

       <img class="statusIcon"

        src='apps/dashboard/red-dot.jpg' />

      </div>

     </div>

    </td>

   </tr>

  </tbody>

</table>

**17. How will you add options to a select box?**

Using the ng-options and ng-model directives.

<fieldset>

 <dl class="control-group">

  <dt>

   <label for="cientId">

    <h4>Client Id:</h4>

   </label>

  </dt>

  <dd>

   <select id="cientId" class="input-xlarge" ng-model="clientId"

    ng-options="reportClient.clientId as reportClient.clientId  for reportClient in reportClients "

    ng-click="getReportParams()" ng-change="getValuationDates()" />

  </dd>

 </dl>

 <dl class="control-group">

  <dt>

   <label for="valuationDate">

    <h4>

     Valuation Date <small>(dd/mm/yyyy)</small>

    </h4>

   </label>

  </dt>

  <dd>

   <select id="valuationDate" class="input-xlarge"

    ng-model="valuationDate"

    ng-options="reportdate for reportdate in reportDates" />

  </dd>

 </dl>

</fieldset>

**18. How will you display inprogress revolving image to indicate that RESTful data is bing loaded?**

<div ng-show="loading">

 <img class="loading" src="portal/images/loading\_32.gif" />

</div>

 $scope.loadReportData = function($http) {

 $scope.loading = true;  // start spinng the image

 $http(

   {

    method : 'GET',

    url : propertiesService.get('restPath')

      + '/myapp/portfolio/'

      + $scope.clientId

      + '/'

      + dateService

        .userToRest($scope.reportDate),

    cacheBreaker : true

   }).success(

   function(data, config) {

    $scope.reportData = data;

    portal.log('reportData: ',

      $scope.reportData);

    $scope.loading = false;   // stop spinning the image

   }).error(

   function(data, status, headers, config) {

    if(data.errorMsg != null) {

     $scope.httpError = data.errorMsg;

    }

    else {

    $scope.httpError = "Error retrieving data from " + errorService

      .getApacheErrorTitleMessage(status,

        data, config);

       }

    $scope.loading = false;  // stop spinning the image

   });

};