Angular

**1. What is Angular?**

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. AngularJS's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology.  
  
AngularJS is what HTML would have been, had it been designed for applications. HTML is a great declarative language for static documents. It does not contain much in the way of creating applications, and as a result building web applications is an exercise in what do I have to do to trick the browser into doing what I want?

AngularJS takes another approach. It attempts to minimize the impedance mismatch between document centric HTML and what an application needs by creating new HTML constructs. AngularJS teaches the browser new syntax through a construct we call directives. Examples include:  
  
1. Data binding, as in {{}}.  
2. DOM control structures for repeating, showing and hiding DOM fragments.  
3. Support for forms and form validation.  
4. Attaching new behavior to DOM elements, such as DOM event handling.  
5. Grouping of HTML into reusable components.

**2. What is a Single Page Application?**

In general, a SPA can be characterized by having an outer 'shell' that serves as the header and navigation for the site while the content of the page changes as different parts of the site are visited.

**3. What does MVC stand for? What does it mean?**

The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller.

Model: the data  
View: what the user sees  
Controller: the logic that brings it all together

**4. What are directives in Angular? What are some examples of built in Angular directives?**

Directives are where you manipulate the DOM and catch DOM events. This is why the directive's compile and link functions both receive the "element" as an argument. You can  
  
-define a bunch of HTML (i.e., a template) to replace the directive  
-bind events to this element (or its children)  
-add/remove a class  
-change the text() value  
-watch for changes to attributes defined in the same element (actually it is the attributes' values -that are watched -- these are scope properties, hence the directive watches the "model" for changes)

Examples of built in directives are: ng-app, ng-model, ng-class, etc.

**5. What does the directive ng-model do?**

The ngModel directive binds an input,select, textarea (or custom form control) to a property on the scope using NgModelController, which is created and exposed by this directive.

ngModel is responsible for:  
  
-Binding the view into the model, which other directives such as input, textarea or select require.  
Providing validation behavior (i.e. required, number, email, url).  
-Keeping the state of the control (valid/invalid, dirty/pristine, touched/untouched, validation errors).  
-Setting related css classes on the element (ng-valid, ng-invalid, ng-dirty, ng-pristine, ng-touched, ng-untouched, ng-empty, ng-not-empty) including animations.  
-Registering the control with its parent form.

ngModel will try to bind to the property given by evaluating the expression on the current scope. If the property doesn't already exist on this scope, it will be created implicitly and added to the scope

* **Using the following HTML, describe what the filter is doing.**

<p ng-repeat=”dog in dogs | orderBy”> {{ dog }} </p>

(default = sorts alphabetically)

* **Based on the following HTML, describe what is wrong.**

<li ng-repeat=”person in people”> {{ people }} </li>

(needs to be person in the angular expression)

**6. Create an text input and use ng-model to bind the property ‘name’ to the scope object.**

<input type = “text” ng-model=”name”>

**7. What does the directive ng-app do?**

Declares an element and all its children as an angular app.

**8. What is a custom directive?**

Directives we can make ourselves to attach desired behavior(or functionality) to components or HTML.

**9. What is a component?**

Used for creating custom HTML elements.

**10. What is the module?**

A container for the different parts of an app including controllers, services, filters, directives. Modules are the building blocks that apps are made of. A module's primary job is to serve as place for the pieces of an app to be registered.

**11. Let’s call our module, “someApp”. How do we create the module with no dependencies in Javascript? How do we link it on the html element?**

angular.module(‘someApp’, []); , <html ng-app=’someApp’></html>

**12. What is a controller?**

Controllers have one job and one job only: to control data moving between the view and the model. They should never do any heavy data processing or logic. They should contain minimal amounts of code. They simply take information from one piece of the application and sends it to another piece. They should never contain logic that will manipulate the DOM.

**13. What does the scope refer to in Angular?**

Scope is Angular's 'glue object' that marries the variables and properties on a controller to the view.

**14. Let’s write a controller called “SignOffController”.**

(function(){

function SignOffController(){

}

angular

.module(“app”)

.controller(“SignOffController”, SignOffController)

})();

**15. Let’s add a property to the scope within the controller called “message” and give it the value, “Goodbye!” inside it.**

(function(){

function SignOffController(){

var vm = this;

vm.message = “Goodbye!”;

}

angular

.module(“app”)

.controller(“SignOffController”, SignOffController)

})();

**16. What does dependency injection refer to? Give an example of one.**

Dependency Injection is a concept in software design that allows for the components of a software project to be loosely coupled. This makes them easier to test and change without affecting the other modules that depend on them.  
  
In Angular, software components (modules, services, and directives) are injected by passing them into the constructor function of whatever it is you're instantiating.

Examples are $http, $location, injecting services into controllers, etc.

**17. What Angular component do we use to share data between controllers?**

Services. Services are reusable components that are separated from views and can be used across multiple controllers and views. You do this by using the .service() or .factory() methods.

**18. What are words we use to define services in Angular?**

Service, factory,

**19. Write an example of a service called ‘MyService’ with a function ‘message’.**

function MyService() {

this.message = function() {}

};  
angular  
 .module("app")

.service("MyService", MyService);

**20. Write a controller called ‘MyController’ using ‘MyService’ we just created.**

function MyController(MyService){

MyService.message();

}

angular  
 .module("app")  
 .controller("MyController", MyController);

**21. What is a factory?**

It’s another way to create a service. With a factory, you create an object, add properties, and then return that object literal. Once injected into the controller, the properties of that object are accessible to the controller.

**22. Write a factory for ‘MyFactory’ with the function ‘MyFunction’.**

function MyFactory() {

return {

MyFunction : MyFunction  
}

function MyFunction() {

}

} // awesome code

angular  
.module("app")

.factory("MyFactory", MyFactory);

Note: We use it in the controller the same way, just create it differently.

**23. What are some built in services we use in Angular?**

$http, $timeout, $location.

**24. What is routing in Angular?**

In a multi-page site we can link to different pages using the same anchor tags and attributes. For a single page application, we need a way to tell the browser to load different content. We can do this with routing.  
  
Originally routing was built-in to Angular. Later, it was decided that it should be maintained as its own module and repository. So we must link to it and include it as a module dependency in order to use it.

**25. How do we configure a route?**

For our routes, we will add a object to our module. Inside this config object we will inject the  
to define our routes and define which controllers are used with what views.  
The function takes a route name as a string and an object with the route's properties such as  
and templateUrl.  
The function defines what the router should do for unknown routes.

angular  
 .module("app", ["ngRoute"])

.config(function($routeProvider) {

$routeProvider

.when("/home", {

templateUrl: "home.html"

}).when("/greatness", {  
 controller: "GreatnessController as routing",

templateUrl: "greatness.html"  
}).when("/amazing", {  
 controller: "AmazingController as routing",

templateUrl: "amazing.html"  
}).otherwise({ redirectTo: "/home" });

});

**26. Let’s configure a route with the path /about to the templateUrl about.html and use the controller aboutCtrl.**

angular  
 .module("app", ["ngRoute"])

.config(function($routeProvider) {

$routeProvider

.when("/about", {

controller: “AboutController as routing”,

templateUrl: "about.html"

\*\*\*\*CAN ALSO DO - template: “<component></component>” AND DON’T NEED A CONTROLLER\*\*\*\*

});

**27. What is a view and how do we use it with routing?**

A view is the visible part of the website (the DOM). When using a router like config , views are usually partial (incomplete) snippets of HTML that are injected into the viewport as needed.

We can link to these snippets of html in our router configuration using the property. We can also define templates inline using the property but this is usually not recommended for anything but the simplest templates.

**28. What’s the difference between template and templateUrl?**

Template is a string of HTML rendered within a directive, templateUrl is a path to an .html file that will render into the directive.

**29. How do we write out custom directives?**

function helloWorld() {

return {  
 restrict: "A",  
 link: function($scope, $element, $attrs) {  
 $element.on("mouseover", function() {

alert("Hello World!");  
});

});

angular

.module(“app”);

.directive("helloWorld", helloWorld);

**30. How do we link it in our HTML?**

<div hello-world></div>

**31. Write a component called newHeader to create a custom header that says “Hello World!”**

var newHeader = {

controller: "TestController",

template: "<h1>Hello World </h1>"  
}  
function TestController() {

var vm = this;

vm.message = "Hello World!";  
}  
angular  
.module("app")  
.component(“newHeader", newHeader);

**32. Write a GET request in Angular using $http service using the url www.something.com/something.json.**

$http({

url: ‘[www.something.com/something.json](http://www.something.com/something.json)’,

method: ‘GET’

})

**33. Console.log the response:**

$http({

url: ‘[www.something.com/something.json](http://www.something.com/something.json)’,

method: ‘GET’

})**.**then(function(response){

console.log(response);

})