# Angular JS Interview Question

1. **What is AngularJS?**
   AngularJS is an open-source JavaScript framework developed by Google to build dynamic, single-page web applications (SPAs).
2. **What is data binding in AngularJS?**
   Data binding is the automatic synchronization of data between the model and the view components. It allows changes in one part of the application to be automatically reflected in other parts.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <input type="text" ng-model="name">
  <p>Hello, {{ name }}</p>
</div>
```

3. **What is ng-app directive used for?**
   The `ng-app` directive defines the root element of an AngularJS application and initializes the AngularJS framework.

Example:

```
htmlCopy code
<!DOCTYPE html>
<html ng-app="myApp">
  <!-- ... -->
</html>
```

4. **Explain ng-model directive in AngularJS.**
   The `ng-model` directive binds the value of an HTML element (like input, select, textarea) to a property in the AngularJS scope.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <input type="text" ng-model="name">
  <p>Hello, {{ name }}</p>
</div>
```

5. **What are AngularJS expressions?**
   AngularJS expressions are used to bind application data to HTML and are placed inside double curly braces {{ expression }}.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <p>{{ 2 + 2 }}</p>
</div>
```

6. **How to create a module in AngularJS?**
   A module in AngularJS is created using the `angular.module` function.

Example:

```
jsCopy code
var app = angular.module('myApp', []);
```

7. **What is dependency injection in AngularJS?**
   Dependency injection is a design pattern used in AngularJS to inject required dependencies into an object rather than creating them within the object.

Example:

```
jsCopy code
app.controller('MyController', ['$scope', function($scope) {
  // Controller logic here
}]);
```

8. **Explain ng-repeat directive.**
   The `ng-repeat` directive is used to repeat a set of HTML elements for each item in an array or object.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <ul>
    <li ng-repeat="item in items">{{ item }}</li>
  </ul>
</div>
```

9. **How to make an HTTP request in AngularJS?**
   AngularJS provides the `$http` service to make HTTP requests.

Example:

```
jsCopy code
app.controller('MyController', ['$http', function($http) {
  $http.get('/api/data')
    .then(function(response) {
      // Process the response data
    })
    .catch(function(error) {
      // Handle the error
    });
}]);
```

10. **What is the use of ng-click directive?**
    The `ng-click` directive is used to bind a function to an element's click event.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <button ng-click="showMessage()">Click Me</button>
</div>
```

**Intermediate Level:**

12. **Explain AngularJS services.**
    AngularJS services are singleton objects that provide functionalities and can be used across an application.

Example:

```
jsCopy code
app.service('MyService', function() {
  this.sayHello = function(name) {
    return "Hello, " + name + "!";
  };
});
```

13. **What is the purpose of $rootScope in AngularJS?**
    The `$rootScope` is the top-level scope in an AngularJS application and is available throughout the application's lifecycle. It acts as a global scope.

Example:

```
jsCopy code
app.controller('MyController', ['$scope', '$rootScope', function($scope, $rootScope) {
  $rootScope.appTitle = "My App";
}]);
```

14. **Explain the concept of directives in AngularJS.**
    Directives are custom HTML attributes or elements that extend the functionality of AngularJS. They can be used to create reusable components.

Example:

```
jsCopy code
app.directive('myDirective', function() {
  return {
    restrict: 'E',
    template: '<p>This is my custom directive</p>'
  };
});
```

```
htmlCopy code
<my-directive></my-directive>
```

15. **What is a digest cycle in AngularJS?**
    The digest cycle is the process in which AngularJS updates the bindings between the model and the view. It keeps track of changes in the model and reflects those changes in the view.

16. **What is ng-if directive used for?**
    The `ng-if` directive conditionally renders an element based on an expression.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <div ng-if="showElement">This will be shown if showElement is truthy.</div>
</div>
```

17. **How does routing work in AngularJS?**
    AngularJS provides the `ngRoute` module to implement client-side routing. It allows different views and controllers to be loaded based on the route URL.

Example:

```
htmlCopy code
<div ng-view></div>
```

```
jsCopy code
app.config(function($routeProvider) {
  $routeProvider
    .when('/home', {
      templateUrl: 'home.html',
      controller: 'HomeController'
    })
    .when('/about', {
      templateUrl: 'about.html',
      controller: 'AboutController'
    })
```

```
    .otherwise({
      redirectTo: '/home'
    });
});
```

18. **What are filters in AngularJS?**
    Filters are used to format data before displaying it in the view. AngularJS provides several built-in filters like `currency`, `date`, `uppercase`, etc.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <p>{{ price | currency }}</p>
</div>
```

19. **How to perform two-way data binding in AngularJS?**
    Two-way data binding is achieved using the `ng-model` directive, allowing changes in the view to update the model and vice versa.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <input type="text" ng-model="name">
  <p>Hello, {{ name }}</p>
</div>
```

20. **What is the purpose of ng-disabled directive?**
    The `ng-disabled` directive disables an element (like button or input) based on an expression.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <button ng-disabled="isDisabled">Click Me</button>
</div>
```

21. **How to handle forms in AngularJS?**
    AngularJS provides the `ngForm` directive to handle HTML forms and their validation.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <form name="myForm" ng-submit="submitForm()">
    <input type="text" name="username" ng-model="user.username" required>
    <button type="submit">Submit</button>
  </form>
</div>
```

```
jsCopy code
app.controller('MyController', ['$scope', function($scope) {
  $scope.submitForm = function() {
    if ($scope.myForm.$valid) {
      // Form is valid, submit the data
    }
  };
}]);
```

**Advanced Level:**

22. **What are directives with an isolated scope? Provide an example**.
    Directives with an isolated scope have their own scope that is independent of the parent scope. They use the scope property in the directive definition.

Example:

```
jsCopy code
app.directive('myDirective', function() {
  return {
    restrict: 'E',
    scope: {
      message: '@'
    },
    template: '<p>{{ message }}</p>'
  };
});
```

```
htmlCopy code
<my-directive message="Hello from the isolated scope"></my-directive>
```

23. **Explain transclusion in AngularJS directives.**
    Transclusion allows you to insert content into a directive's template from the directive's parent element.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <div my-directive>
    <p>This content will be transcluded into the directive template.</p>
  </div>
</div>
```

```
jsCopy code
app.directive('myDirective', function() {
  return {
    restrict: 'A',
    transclude: true,
    template: '<div><p>Directive content:</p><div ng-transclude></div></div>'
  };
});
```

24. **What is the difference between `$watch` and `$watchCollection`?**
    `$watch` is used to watch changes in the value of a scope variable, while `$watchCollection` is used to watch changes in an array or object.

25. **Explain the concept of custom filters in AngularJS.**
    Custom filters are used to perform custom data filtering in AngularJS. They can be used to format, sort, or manipulate data.

Example:

```
jsCopy code
app.filter('reverse', function() {
```

```
  return function(input) {
    return input.split('').reverse().join('');
  };
});
```

```
htmlCopy code
<div ng-app="myApp">
  <p>{{ 'Hello' | reverse }}</p>
</div>
```

26. **How to handle routing in AngularJS without the ngRoute module?**
    You can use the `$location` service and `$routeProvider` to handle routing without the `ngRoute` module.

Example:

```
jsCopy code
app.config(function($routeProvider, $locationProvider) {
  $routeProvider
    .when('/home', {
      templateUrl: 'home.html',
      controller: 'HomeController'
    })
    .when('/about', {
      templateUrl: 'about.html',
      controller: 'AboutController'
    })
    .otherwise({
      redirectTo: '/home'
    });

  $locationProvider.html5Mode(true);
});
```

27. **Explain the concept of promises in AngularJS.**
    Promises are used for handling asynchronous operations in AngularJS. They allow you to handle success and error cases of an asynchronous operation.

Example:

```
jsCopy code
app.controller('MyController', ['$http', function($http) {
  $http.get('/api/data')
    .then(function(response) {
      // Success callback
      console.log(response.data);
    })
    .catch(function(error) {
      // Error callback
      console.error(error);
    });
}]);
```

28. **What are AngularJS decorators? Provide an example.**
    Decorators are used to modify or extend the behavior of AngularJS services, directives, etc.

Example (Decorator for the $log service to prepend a timestamp to log messages):

```
jsCopy code
app.config(['$provide', function($provide) {
  $provide.decorator('$log', ['$delegate', function($delegate) {
    var originalLogFn = $delegate.log;

    $delegate.log = function() {
      var args = [].slice.call(arguments);
      args[0] = new Date().toISOString() + ' - ' + args[0];
      originalLogFn.apply(null, args);
    };

    return $delegate;
  }]);
}]);
```

29. **How does dependency injection work in AngularJS?**
    AngularJS uses dependency injection to provide necessary dependencies to components like controllers, services, and directives. Dependencies are specified as function parameters or through an array annotation.

30. **What is the purpose of the `$compile` service in AngularJS?**
    The `$compile` service is responsible for compiling the DOM elements that contain

AngularJS directives and expressions. It traverses the DOM, binds data to the view, and sets up watchers.

31. **Explain the concept of interpolation in AngularJS.**
    Interpolation is the process of binding data from the AngularJS scope to the view by using expressions placed within double curly braces {{ expression }}.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <p>Hello, {{ name }}</p>
</div>
```

32. **What are directives with a controller? Provide an example.**
    Directives with a controller are used to define a controller for a specific directive. The controller can expose methods and properties for the directive to use.

Example:

```
jsCopy code
app.directive('myDirective', function() {
  return {
    restrict: 'E',
    controller: function($scope) {
      $scope.message = "Hello from directive controller";
    },
    template: '<p>{{ message }}</p>'
  };
});
```

```
htmlCopy code
<my-directive></my-directive>
```

33. **How to implement form validation in AngularJS using ngMessages?**
    `ngMessages` is an AngularJS module that provides enhanced form validation messages. It allows you to display custom validation messages based on form validation states.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <form name="myForm">
    <input type="email" name="email" ng-model="user.email" required>
    <div ng-messages="myForm.email.$error">
      <div ng-message="required">Email is required.</div>
      <div ng-message="email">Invalid email address.</div>
    </div>
  </form>
</div>
```

34. **What are the differences between one-way binding ({{ }}) and two-way binding (ng-model)?**
    One-way binding (`{{ }}`) binds the data from the model to the view, while two-way binding (`ng-model`) binds the data both from the model to the view and from the view back to the model.

35. **Explain the concept of ngSanitize module in AngularJS.**
    The `ngSanitize` module in AngularJS is used to sanitize and parse HTML content to prevent cross-site scripting (XSS) attacks when using `ng-bind-html` or `ng-bind-html-unsafe`.

36. **How to handle HTTP errors globally in AngularJS?**
    You can use the `$httpProvider` and the `interceptors` property to intercept HTTP responses and handle errors globally.

Example:

```
jsCopy code
app.config(['$httpProvider', function($httpProvider) {
  $httpProvider.interceptors.push(function($q) {
    return {
      'responseError': function(rejection) {
        // Handle the error globally
        return $q.reject(rejection);
      }
    };
  });
}]);
```

37. **Explain the difference between $http and $resource services in AngularJS.**
    `$http` is the core service for making HTTP requests in AngularJS, while `$resource` is an AngularJS service that simplifies interacting with RESTful APIs by providing a higher-level abstraction.

38. **How to use AngularJS with TypeScript?**
    To use AngularJS with TypeScript, you need to define interfaces for models and use TypeScript annotations to specify types.

Example:

```
typescriptCopy code
interface IUser {
  id: number;
  name: string;
}

app.controller('MyController', ['$scope', function($scope) {
  $scope.user: IUser = {
    id: 1,
    name: 'John Doe'
  };
}]);
```

39. **What are AngularJS animations?**
    AngularJS animations are used to apply CSS-based animations to elements when they enter or leave the DOM, or when their state changes.

40. **How to include third-party libraries in AngularJS?**
    You can include third-party libraries in AngularJS using the `script` tag or by installing them through package managers like npm and using module bundlers like Webpack.

41. **How to implement pagination in AngularJS?**
    You can implement pagination in AngularJS by using the `ng-repeat` directive along with filters to display a limited number of items per page.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <ul>
```

```
    <li ng-repeat="item in items | limitTo: itemsPerPage * currentPage">
      {{ item }}
    </li>
  </ul>
  <button ng-click="prevPage()">Previous</button>
  <button ng-click="nextPage()">Next</button>
</div>
```

```
jsCopy code
app.controller('MyController', ['$scope', function($scope) {
  $scope.itemsPerPage = 5;
  $scope.currentPage = 0;
  $scope.items = [...]; // Your data array

  $scope.prevPage = function() {
    if ($scope.currentPage > 0) {
      $scope.currentPage--;
    }
  };

  $scope.nextPage = function() {
    var lastPage = Math.ceil($scope.items.length / $scope.itemsPerPage) - 1;
    if ($scope.currentPage < lastPage) {
      $scope.currentPage++;
    }
  };
}]);
```

42. **How to implement lazy loading in AngularJS?**
    Lazy loading is the technique of loading parts of an application only when they are needed. You can use the `ocLazyLoad` library or the native AngularJS route configuration to implement lazy loading.

43. **What is the role of the `$route` service in AngularJS?**
    The `$route` service in AngularJS is used to configure and manage application routes. It is responsible for handling routing and loading views and controllers.

44. **How to implement authentication and authorization in AngularJS?**
    Authentication and authorization can be implemented in AngularJS by using services to handle login, session management, and role-based access control.

45. **How to communicate between directives in AngularJS?**
    You can use the `require` property in the directive definition object to communicate

between directives or use a shared service.

46. **Explain the concept of state management in AngularJS.**
    State management refers to the process of managing the data and state of an application. In AngularJS, you can use services, `$rootScope`, and shared models for state management.

47. **How to optimize the performance of an AngularJS application?**
    To optimize the performance of an AngularJS application, you can minify and concatenate JavaScript files, enable HTTP caching, and use production-ready builds.

48. **What are factories and services in AngularJS? How do they differ?**
    Both factories and services are used to create reusable components in AngularJS.

A factory is a function that returns an object or a function and is called only once, while a service is a constructor function and is instantiated with the `new` keyword.

Example (Factory):

```
jsCopy code
app.factory('MyFactory', function() {
  return {
    // Factory logic here
  };
});
```

Example (Service):

```
jsCopy code
app.service('MyService', function() {
  // Service logic here
});
```

49. **How to implement server-side pagination in AngularJS?**
    To implement server-side pagination, you need to interact with the server API and fetch the data for each page based on the page number and page size.

50. **How to use $interval in AngularJS?**
    The `$interval` service is used to run a function repeatedly at a specified interval.

Example:

```
jsCopy code
app.controller('MyController', ['$scope', '$interval', function($scope, $interval) {
  var intervalPromise = $interval(function() {
    // Function to be executed at the interval
  }, 1000);

  $scope.stopInterval = function() {
    $interval.cancel(intervalPromise);
  };
}]);
```

51. **Explain the concept of testing in AngularJS.**
    Testing in AngularJS is done using tools like Karma and Jasmine. Unit tests and end-to-end tests are commonly used to ensure the quality and functionality of the application.

52. **How to handle route-based resolve in AngularJS?**
    You can use the `resolve` property in the route configuration to resolve data before a route change. This can be helpful in fetching data from APIs before displaying the route.

Example:

```
jsCopy code
app.config(['$routeProvider', function($routeProvider) {
  $routeProvider
    .when('/items', {
      templateUrl: 'items.html',
      controller: 'ItemsController',
      resolve: {
        itemsData: function($http) {
          return $http.get('/api/items').then(function(response) {
            return response.data;
          });
        }
      }
    });
}]);

app.controller('ItemsController', ['$scope', 'itemsData', function($scope, itemsData) {
  $scope.items = itemsData;
}]);
```

53. **How to use $cookies in AngularJS?**
    The `$cookies` service in AngularJS is used to read, write, and delete browser cookies.

Example:

```
jsCopy code
app.controller('MyController', ['$scope', '$cookies', function($scope, $cookies) {
  // Write a cookie
  $cookies.put('username', 'John Doe');

  // Read a cookie
  var username = $cookies.get('username');

  // Delete a cookie
  $cookies.remove('username');
}]);
```

54. **How to use ngTouch in AngularJS?**
    `ngTouch` is an AngularJS module that provides touch event support for mobile devices. It is used to improve the performance and responsiveness of touch-based interactions.

55. **How to implement internationalization (i18n) in AngularJS?**
    You can use the `angular-translate` library or AngularJS's built-in `$locale` service to implement internationalization in AngularJS applications.

56. **Explain the concept of the digest cycle and how it works in AngularJS.**
    The digest cycle is a loop in which AngularJS checks for changes in the model and updates the view accordingly. It runs until all the watchers have been checked and no changes are detected.

During the digest cycle, AngularJS compares the current model values with the previous values and updates the bindings in the view if necessary. The digest cycle is triggered by various events like user interactions, AJAX requests, timers, etc.

57. **How to handle exceptions and errors in AngularJS?**
    You can use the `$exceptionHandler` service to handle exceptions and errors in AngularJS. By default, it logs errors to the console, but you can customize it to handle errors differently.

58. **What are the differences between AngularJS and Angular (Angular 2+)?**
    AngularJS is the first version of the Angular framework, and Angular (also known as Angular 2+) is a completely rewritten version of AngularJS with major architectural changes. Some of the differences include:

- **Language**: AngularJS uses JavaScript, while Angular uses TypeScript.
- **Architecture**: AngularJS uses a two-way data binding approach, whereas Angular uses a unidirectional data flow and hierarchical dependency injection.
- **Performance**: Angular is generally faster and more performant due to its optimized rendering and change detection mechanisms.
- **Modules**: AngularJS uses `angular.module()` for defining modules, while Angular uses `@NgModule` decorators.
- **Component**-based Architecture: Angular introduces the concept of components, making it more component-based and modular.
- **Mobile Support**: Angular has better support for mobile applications compared to AngularJS.
- **Learning Curve**: Angular has a steeper learning curve compared to AngularJS.

1. **How to handle memory leaks in AngularJS applications?**
   To handle memory leaks, you need to make sure that you unsubscribe from all subscriptions, remove all event listeners, and clear references to objects when they are no longer needed. Use tools like `$onDestroy` and `removeEventListener` to clean up resources properly.
2. **How to use AngularJS with RESTful APIs?**
   You can use the `$http` service or `$resource` service to interact with RESTful APIs in AngularJS. The `$http` service is more versatile and can handle various HTTP methods, while `$resource` provides a higher-level abstraction for working with RESTful APIs.
3. **How to implement file uploads in AngularJS?**
   To implement file uploads, you can use the `input type="file"` element along with the `FormData` API to send files to the server using the `$http` service.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <input type="file" ng-model="file" accept=".jpg, .png">
  <button ng-click="uploadFile()">Upload</button>
</div>
```

```
jsCopy code
app.controller('MyController', ['$scope', '$http', function($scope, $http) {
  $scope.uploadFile = function() {
    var formData = new FormData();
    formData.append('file', $scope.file);

    $http.post('/api/upload', formData, {
      transformRequest: angular.identity,
      headers: { 'Content-Type': undefined }
    })
    .then(function(response) {
      // File uploaded successfully
    })
    .catch(function(error) {
      // Error uploading file
    });
  };
}]);
```

59. **How to implement server-side rendering (SSR) in AngularJS?**
    Server-side rendering can be achieved using technologies like Angular Universal, which allows you to pre-render AngularJS applications on the server and improve the initial page load time.

60. **What are the best practices for organizing code in AngularJS applications?**

- Use a modular structure with separate modules for each feature.
- Organize code by feature rather than by type (controllers, services, etc.).
- Use a consistent naming convention for files and components.
- Keep business logic in services and use controllers for view-related logic.
- Minimize the use of the `$rootScope` and prefer using `$scope`.
- Use the "Controller As" syntax for better code readability.

1. **How to handle real-time updates in AngularJS?**
   To handle real-time updates, you can use technologies like WebSockets or Server-Sent Events (SSE) to receive data updates from the server and use AngularJS's data binding to update the view accordingly.

61. **How to use third-party libraries that manipulate the DOM with AngularJS?**
When using third-party libraries that manipulate the DOM directly, you should be cautious to avoid conflicts with AngularJS's digest cycle. Make sure to use `$apply` or `$timeout` to trigger the digest cycle after making changes with the library.

Example:

```
jsCopy code
app.controller('MyController', ['$scope', function($scope) {
  $scope.$apply(function() {
    // Call the third-party library's DOM manipulation code here
  });
}]);
```

62. **How to implement lazy loading for images in AngularJS?**
   You can use the `ng-src` directive with a placeholder image to implement lazy loading for images in AngularJS. When the image is in the viewport, you can dynamically change the `src` attribute to load the actual image.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <img ng-src="{{ isVisible ? 'actual_image.jpg' : 'placeholder.jpg' }}" alt="Image">
</div>
```

```
jsCopy code
app.controller('MyController', ['$scope', function($scope) {
  // Check if the image is in the viewport
  $scope.isVisible = isImageVisibleInViewport();

  function isImageVisibleInViewport() {
    // Your logic to check if the image is visible in the viewport
  }
```

```
}]);
```

63. **How to handle multiple environments (development, production) in AngularJS?**
    You can use AngularJS constants or environment variables to handle multiple environments. Define different configuration files for each environment and load the appropriate configuration based on the environment.

Example:

```
jsCopy code
// app.config.js
var env = 'development'; // or 'production'
angular.module('myApp')
  .constant('ENV', env)
  .constant('API_BASE_URL', env === 'production' ? 'https://api.prod.com' : 'https://api.d
ev.com');
```

64. **How to implement infinite scrolling in AngularJS?**
    Infinite scrolling can be implemented by using the `ngScroll` directive or by manually handling the scroll event and loading more data when the user reaches the bottom of the page.

65. **How to use ngClass in AngularJS?**
    The `ngClass` directive allows you to dynamically apply CSS classes to an element based on conditions or expressions.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <div ng-class="{'highlight': isHighlighted, 'bold': isBold}">Content</div>
</div>
```

```
jsCopy code
app.controller('MyController', ['$scope', function($scope) {
  $scope.isHighlighted = true;
  $scope.isBold = false;
```

```
}]);
```

66. **How to implement content filtering in AngularJS?**
    You can use the `ng-filter` directive along with the `filter` filter to implement content filtering in AngularJS.

Example:

```
htmlCopy code
<div ng-app="myApp">
  <input type="text" ng-model="searchText">
  <ul>
    <li ng-repeat="item in items | filter: searchText">{{ item }}</li>
  </ul>
</div>
```

67. **How to handle security in AngularJS applications?**
    To handle security, you should implement measures like input validation, data sanitization, server-side validation, and secure authentication and authorization mechanisms. Also, use HTTPS for secure communication between the client and the server.