# AngularJS Exercises

**Topics :** AngularJS
**Written on** January 09, 2024



### Basics:

1. Create a simple AngularJS application with a controller and a view.
2. Use `ng-model` to bind an input field to a variable in the controller.
3. Implement a basic two-way data binding example.
4. Create an array in the controller and use `ng-repeat` to display its elements in the view.
5. Explore and implement different built-in AngularJS filters (e.g., `currency`, `date`, `uppercase`).
6. Use `ng-if` and `ng-show` directives to conditionally display elements.
7. Implement a simple form with validation using AngularJS directives.
8. Explore and use the `$http` service to fetch data from an external API.
9. Create a custom filter that formats a string in a specific way.
10. Use `$timeout` to delay the execution of a function in the controller.

### Controllers and Scope:

11. Implement a nested controller structure where child controllers inherit from the parent controller's scope.
12. Use `$rootScope` to store a variable that is accessible across multiple controllers.
13. Create a controller that uses the `$watch` function to monitor changes in a variable.
14. Implement a controller with methods that perform different actions (e.g., add, delete).
15. Explore the use of controller aliases using the `controllerAs` syntax.
16. Use `$emit` and `$broadcast` to send events between controllers.
17. Implement a controller with a custom method for sorting an array of objects.
18. Create a controller with a function that filters data based on a specific condition.
19. Implement a controller with a method that makes a custom AJAX request using `$http`.
20. Use `$routeParams` to retrieve and display parameters from the URL in a controller.

### Directives:

21. Create a custom directive that changes the appearance of an element.
22. Implement a directive with an isolated scope and bind variables to it.
23. Use the `link` function in a directive to manipulate the DOM.
24. Explore and use the `ng-click` directive to handle click events.
25. Create a directive that dynamically generates HTML content.
26. Implement a custom directive that performs form validation.
27. Use the `ng-transclude` directive to include content within a custom directive.
28. Create a directive that communicates with a controller using a service.
29. Implement a directive that animates an element using ngAnimate.
30. Explore the use of directive priorities in AngularJS.

## Services and Dependency Injection:

31. Create a custom service that performs a specific task (e.g., data manipulation).
32. Use the `$http` service to fetch data from an external API in a service.
33. Implement a service that communicates with a RESTful API.
34. Explore and use the `$q` service for handling promises in a service.
35. Create a service with a method that stores and retrieves data from local storage.
36. Implement a service that shares data between controllers using dependency injection.
37. Use the `$location` service to navigate between different views in a service.
38. Create a service that manages user authentication using tokens.
39. Explore and use the `$rootScope` service in a custom service.
40. Implement a service that handles exception logging using `$exceptionHandler`.

## Routing and Views:

41. Set up AngularJS routing using the `ngRoute` module.
42. Define multiple routes with different controllers and templates.
43. Use route parameters to pass data between controllers and views.
44. Implement nested views using the `ng-view` directive.
45. Explore and use route resolve to fetch data before loading a view.
46. Create a navigation menu with links that navigate to different views.
47. Implement a "404 Not Found" page for unknown routes.
48. Use `$location` to programmatically navigate between views.
49. Explore and implement route animations using ngAnimate.
50. Implement a route that redirects to another route after a certain action.