Front End Development using Angular JS

Introduction to Angular JS

- AngularJS version 1.0 was released in 2012. The idea turned out very well, and the project is now officially supported by Google.

- AngularJS is a **JavaScript framework**. It can be added to an HTML page using a CDN link with a **<script> tag.**

- AngularJS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.

Introduction to Angular JS

- AngularJS directives are HTML attributes with an **ng** prefix.
- AngularJS extends HTML with **ng-”directives”**.
- The **ng-app** directive defines an AngularJS application.
- The **ng-model** directive binds the value of HTML controls (input, select, text-area) to application data.

Introduction to Angular JS

- The **ng-bind directive** binds application data to the HTML view.
- The **ng-init directive** initializes AngularJS application variables.
- Coding Example

Role of Angular JS in Web Development

- AngularJS, developed by Google, plays a significant role in modern web development, especially in building **dynamic, single-page applications (SPAs)**.

- AngularJS automatically synchronizes data between the **model** (JavaScript objects) and the **view** (HTML). **[Two –Way Data Binding]**

- It follows the **Model-View-Controller** pattern, helping developers organize code better and **separate concerns**, making applications more scalable and maintainable. [MVC Architecture]

- AngularJS introduces directives like ng-model, ng-repeat, and ng-if to extend HTML with custom behavior, enabling creation of reusable UI components.[Directives and Components]

Extra : Single Page Application (SPA)

- A **Single Page Application (SPA)** is a type of web application that loads and updates content dynamically without refreshing the entire page.

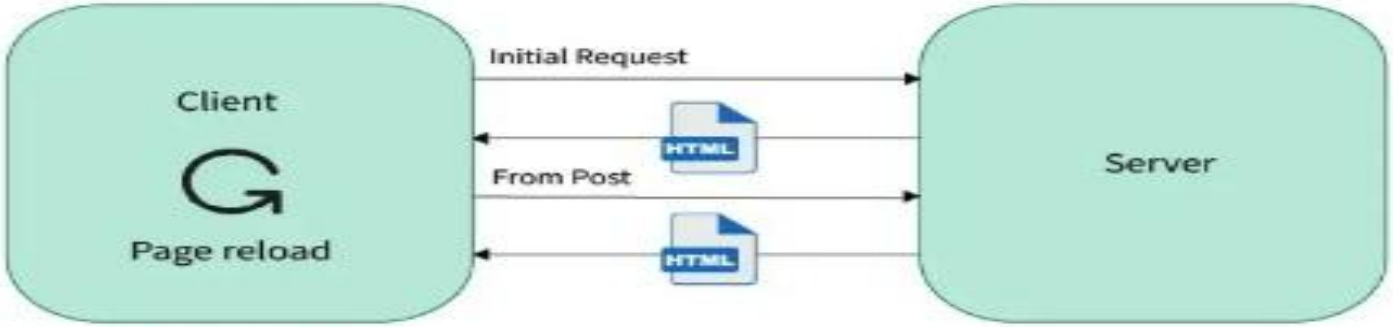
- Unlike traditional websites, SPAs use modern technologies to enhance the user experience by minimizing interruptions and providing a smoother interface.

- Users can interact with the application seamlessly, similar to using desktop software. The main advantage is the elimination of full-page reloads, resulting in a more responsive and engaging web experience.

Extra : Single Page Application (SPA)

Traditional Page Lifecycle

Once the user accesses the page and performs any kind of action on that page, the page gets reloaded with the changes that were done on the server-side.



Single Page Lifecycle

Once the user accesses the page and performs any kind of action on that page, they get an almost-instant reaction from the page (think of Facebook, when you comment on someone's post.)

Role of Angular JS in Web Development

- Built-in **dependency injection (DI)** simplifies the management of services and components, making the code **modular, testable, and reusable**. [Dependency Injection]

- AngularJS is ideal for building SPAs where the content dynamically updates **without refreshing the entire page**, ensuring a smooth user experience.[Single Page Application]

- AngularJS was designed with testing in mind—offering support for both **unit testing and end-to-end testing.** [Testing Support]

Role of Angular JS in Web Development

- The built-in routing system enables navigation between views in an SPA, without requiring full-page reloads, mimicking the feel of a desktop application. [Routing]

- AngularJS uses standard **HTML as the templating language**, which is easy for designers and developers to work with and enhances collaboration. [Template Support]

- It provides a rich set of services (e.g., $http, $location) for handling **AJAX requests, URL routing**, and more, minimizing external library needs. [Built-in Services]

AngularJS Architecture

- AngularJS is a **structural framework for dynamic web applications**. It follows the **Model-View-Controller (MVC)** architecture to organize and manage application components.

- **1. Model:** Represents the data of the application. It's plain JavaScript objects (**POJOs**).The model is the source of truth and is updated by controllers or services.

AngularJS Architecture

- **2. View:** Represents the UI (HTML) that users interact with. Uses AngularJS expressions and directives to bind data and logic to the DOM. Automatically reflects model changes via two-way data binding.

- **3. Controller:** A JavaScript function that defines application behavior. Interacts with the model and updates the view via $scope. Acts as a bridge between model and view.

AngularJS Architecture

- **4. Scope:** An object that connects the controller and the view. Watches expressions and propagates model changes to the view. $scope is the execution context for expressions.

- **5. Services:** Reusable business logic components. Used for operations like HTTP requests, data transformation, etc. Angular provides built-in services (like $http, $timeout) and allows creating custom services.

AngularJS Architecture

- **6. Directives:** Special tokens in the DOM (ng-model, ng-repeat, ng-if) that tell AngularJS to do something. Can be built-in or custom to extend HTML functionality.

- **7. Templates:** HTML with AngularJS-specific syntax and expressions ({{ }}). Defines the view and is rendered dynamically based on the model.

AngularJS Architecture

- **8. Dependency Injection (DI):** AngularJS has a built-in DI system. Automatically injects services, factories, and values into components like controllers and directives.

- **9. $digest Cycle:** A mechanism by which Angular checks and updates the DOM when the model changes. Detects changes by comparing current and previous model states.

- **10. Routing (ngRoute or UI-Router):** Enables building Single Page Applications (SPAs) by navigating between different views without reloading the page. Maps URL routes to views and controllers.

AngularJS Expressions

- AngularJS expressions can be written inside double braces:

➢{{ expression }}.

- AngularJS expressions can also be written inside a directive:

➢ng-bind="expression".

➢AngularJS will resolve the expression and return the result exactly where the expression is written.

AngularJS Expressions

- AngularJS expressions are much like JavaScript expressions: They can contain literals, operators, and variables.

- Coding Example

- If you remove the **ng-app directive**, HTML will display the expression as it is, without solving it.

AngularJS Expressions

- AngularJS numbers are like JavaScript numbers.
- Coding Example
- AngularJS strings are like JavaScript strings.
- Coding Example

AngularJS Vs JavaScript Expressions

- Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.
- Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.
- AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.
- AngularJS expressions support filters, while JavaScript expressions do not.

AngularJS Modules

- An AngularJS module defines an application.
- The module is a container for the different parts of an application.
- The module is a container for the application controllers.
- Controllers always belong to a module.

AngularJS Modules

- A module is created by using the AngularJS function “angular.module”.

```
<script>
var app = angular.module("myApp", []);
</script>
```

- The "myApp" parameter refers to an HTML element in which the application will run.

- Now you can add controllers, directives, filters, and more, to your AngularJS application.

- It is common in AngularJS applications to put the module and the controllers in JavaScript files.

AngularJS Directives

- AngularJS lets you extend HTML with new attributes called **Directives**.

- AngularJS has a set of built-in directives which offers functionality to your applications.

- AngularJS also lets you define your own directives.

AngularJS Directives

- AngularJS directives are extended HTML attributes with the prefix “ng-”.
- The ng-app directive initializes an AngularJS application.
- The ng-init directive initializes application data.
- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

AngularJS Directives

- In addition to all the built-in AngularJS directives, you can create your own directives.

- New directives are created by using the .directive function.

- Coding Example

- However, when naming a directive, you must use a camel case name, w3TestDirective, but when invoking it, you must use - separated name, w3-test-directive.

AngularJS Model

- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

- With the ng-model directive you can bind the value of an input field to a variable created.

- Coding Example

AngularJS Model : Two Way Binding

- The binding goes both ways i.e. If the user changes the value inside the input field, the AngularJS property will also change.

- Coding Example

- The ng-model directive can provide type validation for application data (number, e-mail, required).

- Coding Example

AngularJS Controllers

- AngularJS controllers **control the data** of AngularJS applications. AngularJS controllers are regular **JavaScript Objects**.

- The **ng-controller** directive defines the application controller.

- Coding Example

AngularJS Controllers

- In larger applications, it is common to store controllers in external files.

- Coding Example

AngularJS Scope

- The scope is the binding part between the **view** and the **controller.**
- The scope is an object with the available properties and methods.
- The scope is available for both the view and the controller.
- When you make a controller in AngularJS, you pass the $scope object as an argument.
- It is important to know which scope you are dealing with, at any time.

AngularJS Scope

- For larger applications there can be sections in the HTML DOM which can only access certain scopes.

- By making use of the ng-repeat directive, each repetition can be given access to the current repetition object.

- Coding Example

AngularJS Filters

- Filters can be added in AngularJS to format data.

- AngularJS provides filters to transform data:

`currency` Format a number to a currency format.
`date` Format a date to a specified format.
`filter` Select a subset of items from an array.
`json` Format an object to a JSON string.
`limitTo` Limits an array/string, into a specified number of elements/characters.
`lowercase` Format a string to lower case.
`number` Format a number to a string.
`orderBy` Orders an array by an expression.
`uppercase` Format a string to upper case.

AngularJS Filters

- Filters can be added to expressions by using the pipe character “|”, followed by a filter.

- Coding Example

- Filters are added to directives, like ng-repeat, by using the pipe character “|”.

- Coding Example

AngularJS Filters

- By setting the ng-model directive on an input field, we can use the value of the input field as an expression in a filter.

- Coding Example

AngularJS Services

- In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application. AngularJS has about 30 built-in services.

- Example Services: $location ; $http ; $timeout ;

- The **$location** service has methods which return information about the location of the current web page.

- Coding Example

AngularJS Services

- The **$timeout** service is AngularJS' version of the window.setTimeout function.

- Coding Example

AngularJS Tables

- The **ng-repeat** directive is perfect for displaying tables.
- Coding Example
- One can make use of CSS to add style as well to the displayed table.
- Coding Example
- One can make use of Angular Filters “|” as well to order data in tables.

AngularJS Events

- AngularJS has its own HTML events directives.
- You can add AngularJS event listeners to your HTML elements by using one or more pre-defined directives.
- Examples

ng-cut
ng-dblclick
ng-focus
ng-keydown
ng-keypress
ng-keyup

ng-blur
ng-change
ng-click
ng-copy
ng-mousedown
ng-mouseenter

AngularJS Events

- Mouse events occur when the cursor moves over an element.
- Coding Example
- ng-click directive defines AngularJS code that will be executed when the element is being clicked.
- Coding Example

AngularJS Forms

- Forms in AngularJS provides data-binding and validation of input controls.

- Input controls provides data-binding by using the **ng-model** directive.

- One can also apply ng-model directive to a checkbox.

- Coding Example

AngularJS Forms

- One can also bind radio buttons to your application with the ng-model directive.

- Coding Example

- AngularJS can be used to bind default values to form elements as well as change them dynamically.

- Coding Example

AngularJS Forms : Validation

- AngularJS offers client-side form validation.
- AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.
- AngularJS also holds information about whether they have been touched, or modified, or not.
- You can use standard HTML attributes to validate input, or you can make your own validation functions.

AngularJS Forms : Validation

- Coding Example

- Email Validation can also be used in angular forms.

- Coding Example

- AngularJS is constantly helps updating the state of both the form and the input fields.

AngularJS Forms : Validation

- Input fields have the following states:

$untouched **The field has not been touched yet**

$touched **The field has been touched**

$pristine **The field has not been modified yet**

$dirty **The field has been modified**

$invalid **The field content is not valid**

$valid **The field content is valid**

AngularJS Forms : Validation

- Validation of Forms can be implemented in multiple ways.

- Coding Examples (6 and 7)

Debugging AngularJS applications using developer tools.

Category	Popular Ways	Popular Tools
Inspecting Scopes & Controllers	Use DevTools Console → angular $scope()	**Chrome DevTools** (built-in)
Dependency Injection Debugging	Access services with angular.element like (‘$http')	**Chrome DevTools Console**
Error & Log Tracking	Use $log.debug(), $log.error()	**AngularJS $log service**
Digest Cycle	Count watchers, use one-time binding	**ng-stats** (performance directive)
Template & Data Binding Debugging	Check $error, $valid on forms; inspect data bindings	**ng-inspector** (browser extension directive)
Performance Monitoring	Record with DevTools Performance tab, find repeated digests	**Chrome DevTools**
API/HTTP Debugging	Monitor $http & $resource requests in Network tab	**Chrome DevTools**
Exception & Error Analysis	Customize $exceptionHandler to log more context	**AngularJS built-in services**
Application State Debugging	Visualize scopes, dependencies, bindings, and performance	**AngularJS Chrome Extension**
End-to-End Debugging & Testing	Debug failing UI flows via automated browser tests	**Protractor** (end to end testing tool)

Thank You