**Angular JS**

**Introducing Angular JS**

**Starting out with Angular JS**

**Basic AngularJS**

**Introduction to AngularJS**

AngularJS is a JavaScript framework developed by Google for building dynamic, single-page web applications (SPAs). It extends HTML by adding directives and allows for data binding, making the development process smoother and more efficient.

Let’s go through a simple example step by step to understand AngularJS.

**Step 1: Setup**

To get started with AngularJS, you'll need to include the AngularJS library in your HTML file. You can use a CDN link to include AngularJS.

Here’s how you can set up your HTML file:

html

Copy

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>AngularJS Example</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

</head>

<body>

<div ng-app="myApp" ng-controller="myCtrl">

<h1>{{ greeting }}</h1>

<p>{{ message }}</p>

<input type="text" ng-model="name" placeholder="Enter your name">

<button ng-click="changeMessage()">Click me!</button>

</div>

<script src="app.js"></script>

</body>

</html>

**Step 2: Create AngularJS Module and Controller**

Next, you need to define the AngularJS application. This is done by creating a module, and a controller to handle the logic. Create a file called app.js.

javascript

Copy

// Create an AngularJS module

var app = angular.module("myApp", []);

// Create a controller for the module

app.controller("myCtrl", function($scope) {

// Define a simple greeting message

$scope.greeting = "Hello, AngularJS!";

// Define a dynamic message

$scope.message = "Welcome to the AngularJS tutorial.";

// Define a function that changes the message when the button is clicked

$scope.changeMessage = function() {

$scope.message = "Hello, " + $scope.name + "!";

};

});

**Step 3: Explanation of Code**

1. **HTML Structure**:
   * ng-app="myApp": This directive initializes the AngularJS application, linking it to the module myApp we defined in app.js.
   * ng-controller="myCtrl": This connects the HTML to the myCtrl controller from app.js.
   * {{ greeting }} and {{ message }}: These are **expressions** that AngularJS will automatically replace with values from the controller.
   * ng-model="name": This binds the input field to a variable name on the AngularJS scope, making it two-way bound. The value of name will automatically update as the user types in the input field.
   * ng-click="changeMessage()": This triggers the changeMessage() function in the controller when the button is clicked.
2. **JavaScript (app.js)**:
   * **Module**: angular.module("myApp", []) creates an AngularJS module. The empty array [] means there are no dependencies.
   * **Controller**: app.controller("myCtrl", function($scope) {...}) defines a controller named myCtrl. The controller contains data ($scope.greeting, $scope.message) and a method ($scope.changeMessage) that are bound to the view.
   * **$scope**: $scope is an object that acts as a bridge between the view (HTML) and the controller (JavaScript). Any data stored in $scope is accessible in the HTML using AngularJS expressions.

**Step 4: Run the Application**

To test the application:

1. Save the HTML file as index.html.
2. Save the JavaScript file as app.js.
3. Open the index.html file in a web browser.

You should see:

* A greeting message: "Hello, AngularJS!"
* A welcome message: "Welcome to the AngularJS tutorial."
* An input field where you can type your name.
* A button that, when clicked, changes the message to greet the user by name.

**2)Directives and Controllers**

### Directives and Controllers in AngularJS

AngularJS allows us to create dynamic and powerful web applications using **directives** and **controllers**. Let's go through both concepts step by step with examples.

### 1. ****Directives in AngularJS****

Directives are markers on DOM elements (like attributes, elements, classes, etc.) that tell AngularJS to attach specific behavior to those elements.

AngularJS provides many built-in directives, but you can also create your own custom directives.

#### Built-in Directives

Here are some of the most common built-in directives:

* ng-app: Initializes the AngularJS application.
* ng-model: Binds the value of an HTML element to the scope variable.
* ng-repeat: Repeats an element for each item in an array or object.
* ng-if: Conditionally includes or removes an element from the DOM.
* ng-click: Specifies an expression to evaluate when an element is clicked.

#### Example: Using ng-repeat and ng-click Directives

Let's create an example where we display a list of items and use ng-repeat to repeat the list items dynamically.

**HTML (index.html)**

html

Copy

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>AngularJS Directives Example</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

</head>

<body>

<div ng-app="myApp" ng-controller="myCtrl">

<h1>My Favorite Fruits</h1>

<ul>

<!-- ng-repeat directive to loop through the fruits array -->

<li ng-repeat="fruit in fruits" ng-click="showFruit(fruit)">

{{ fruit }}

</li>

</ul>

<p ng-if="selectedFruit">You selected: {{ selectedFruit }}</p>

</div>

<script src="app.js"></script>

</body>

</html>

**JavaScript (app.js)**

javascript

Copy

// Create an AngularJS module

var app = angular.module("myApp", []);

// Create a controller for the module

app.controller("myCtrl", function($scope) {

// Define an array of fruits

$scope.fruits = ["Apple", "Banana", "Cherry", "Date", "Elderberry"];

// Function to handle the selection of a fruit

$scope.showFruit = function(fruit) {

$scope.selectedFruit = fruit; // Update the selected fruit

};

});

### Explanation of Code:

* **ng-repeat**: Loops through the fruits array and creates a list item (<li>) for each fruit. It displays the name of each fruit and binds the click event to the showFruit function.
* **ng-click**: This directive triggers the showFruit(fruit) function when a fruit is clicked.
* **ng-if**: The ng-if directive displays the message "You selected: [fruit]" only when a fruit is selected.

When you click on any fruit name, it will display the selected fruit below the list using ng-if.

### 2. ****Controllers in AngularJS****

A **controller** in AngularJS is a JavaScript function that is responsible for handling the logic of the application. It manages the data ($scope) and behavior (functions) that the view will use.

Controllers are attached to the view via the ng-controller directive.

#### Example: Controller to Manage Input and Output

Let’s take a simple example where we manage a user's input and display it dynamically.

**HTML (index2.html)**

html

Copy

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>AngularJS Controller Example</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

</head>

<body>

<div ng-app="myApp" ng-controller="myCtrl">

<h1>AngularJS Controllers</h1>

<p>Enter your name:</p>

<input type="text" ng-model="userName">

<p>Hello, {{ userName }}!</p>

<button ng-click="reset()">Reset</button>

</div>

<script src="app2.js"></script>

</body>

</html>

**JavaScript (app2.js)**

javascript

Copy

// Create an AngularJS module

var app = angular.module("myApp", []);

// Create a controller for the module

app.controller("myCtrl", function($scope) {

// Function to reset the user's name

$scope.reset = function() {

$scope.userName = ''; // Clears the input field

};

});

### Explanation of Code:

* **ng-model="userName"**: This directive binds the input field to the userName variable in the controller. Any changes in the input field will automatically update the userName in the scope.
* **{{ userName }}**: This expression displays the value of userName in real-time as the user types.
* **ng-click="reset()"**: This directive calls the reset() function when the button is clicked, which clears the userName field.

### 3. ****Custom Directive Example****

You can create your own custom directives in AngularJS to extend the functionality of HTML.

#### Example: Creating a Custom Directive

Let’s create a custom directive that changes the background color of a div when hovered.

**HTML (index3.html)**

<!DOCTYPE html>

<html lang="en" ng-app="myApp"> <!-- Moved ng-app here -->

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>AngularJS Custom Directive</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

<style>

div {

width: 300px;

height: 200px;

border: 2px solid black;

display: flex;

justify-content: center;

align-items: center;

font-size: 18px;

text-align: center;

}

.hovered {

background-color: yellow !important;

}

</style>

</head>

<body>

<div ng-controller="myCtrl">

<h1>Custom Directive Example</h1>

<!-- Custom Directive in action -->

<div hover-color>

Hover over this box to change its background color!

</div>

</div>

<script src="app3.js"></script>

</body>

</html>

**JavaScript (app3.js)**

### // Create an AngularJS module

### var app = angular.module("myApp", []);

### // Create a controller to ensure AngularJS is working

### app.controller("myCtrl", function($scope) {

### console.log("Controller loaded!"); // Debugging log

### });

### // Create a custom directive called hoverColor

### app.directive("hoverColor", function() {

### return {

### restrict: 'A', // A for attribute directive

### link: function(scope, element, attrs) {

### console.log("Directive applied!"); // Debugging log

### element.on('mouseenter', function() {

### console.log("Mouse entered! Changing color."); // Debugging log

### element.addClass('hovered'); // Add class instead of setting CSS

### });

### element.on('mouseleave', function() {

### console.log("Mouse left! Resetting color."); // Debugging log

### element.removeClass('hovered');

### });

### }

### };

### });

### Explanation of Code:

* **hover-color**: This is the custom directive we created. We applied it as an attribute to the div.
* **Directive Definition**: In the directive, we defined a link function that adds event listeners to the div. When the mouse enters, the background color changes to yellow, and when the mouse leaves, the background color returns to transparent.

### Summary:

* **Directives**: AngularJS uses directives to add functionality to HTML elements. You can use built-in directives like ng-repeat, ng-click, and ng-if, or you can create custom directives.
* **Controllers**: Controllers in AngularJS manage the application's logic. They bind data and behavior to the view using $scope.
* **Custom Directives**: You can extend HTML by creating your own custom directives to add specific behavior to DOM elements.

These two concepts (directives and controllers) form the foundation of an AngularJS application.