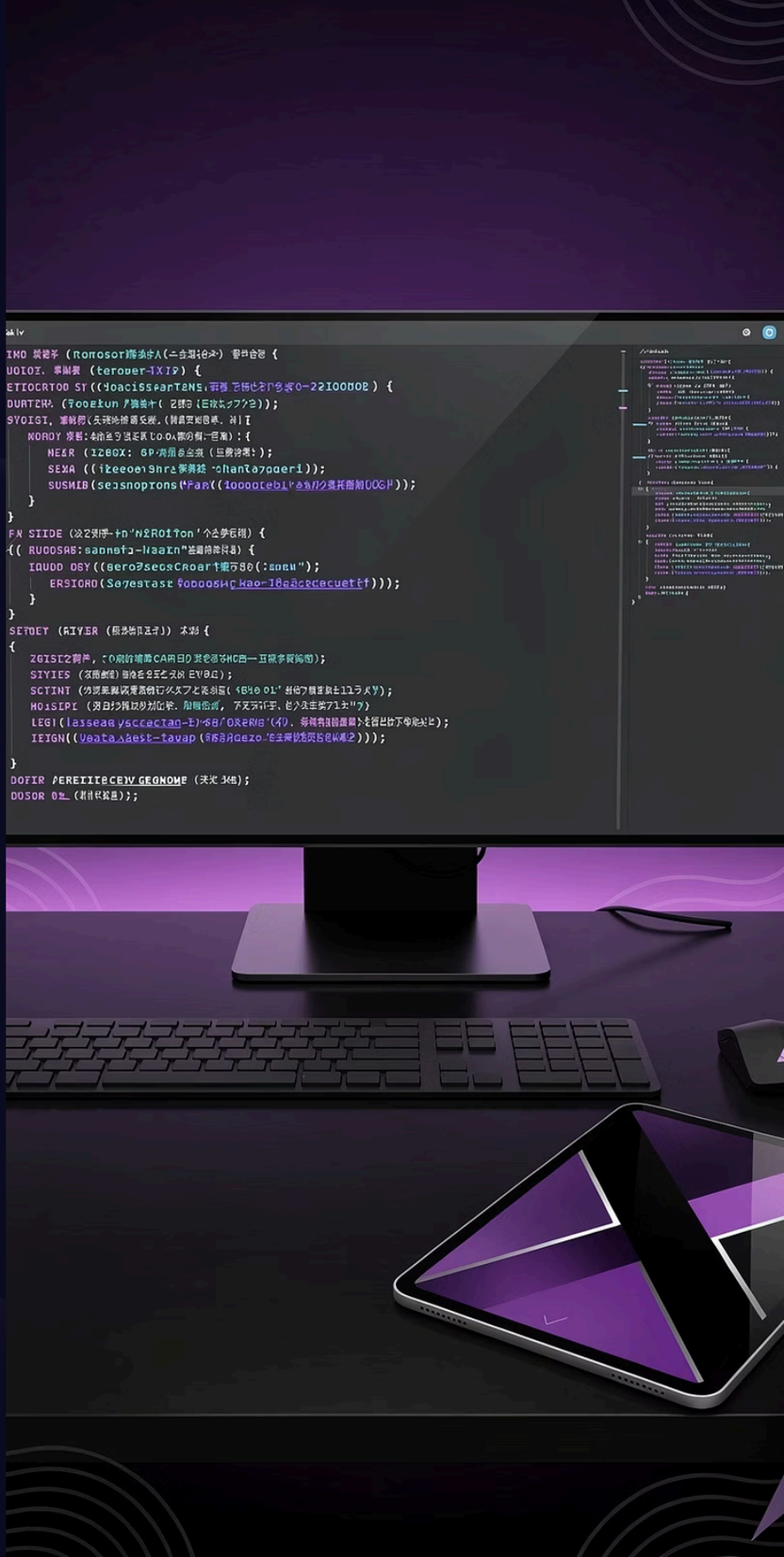
AngularJS Course

Build Dynamic Single Page Applications Using the Angular Framework — a comprehensive, hands-on journey from core concepts to real-world project development.

14 MODULES

BEGINNER TO INTERMEDIATE

SPA DEVELOPMENT



What You'll Learn

This course is structured into 14 focused modules that take you from AngularJS fundamentals to building fully functional single-page applications. Here's your learning roadmap at a glance.

01

Foundations

Modules 1–3: Introduction, setup, and core syntax including directives, expressions, and data binding.

02

Core Framework

Modules 4–7: Directives, controllers, scope, modules, and form handling with validation.

03

Services & Routing

Modules 8–10: Filters, services, factories, and configuring client-side routing for SPAs.

04

Advanced & Projects

Modules 11–14: HTTP calls, events, advanced concepts, and hands-on project work.

MODULE 1

Introduction to AngularJS

What is AngularJS?

AngularJS is a **JavaScript-based open-source front-end framework** developed by Google. It extends HTML with new attributes and makes web pages more responsive through dynamic data binding and a declarative programming model.

Key Comparisons

AngularJS vs Vanilla JS

AngularJS abstracts DOM manipulation, offering two-way binding and MVC architecture out of the box.

AngularJS vs Angular

Angular (2+) is a complete rewrite using TypeScript and components. AngularJS uses JavaScript and controllers.

✅ Advantages

Two-way data binding, dependency injection, reusable components, large community, Google-backed.

⚠️ Limitations

Performance overhead on large apps, steep learning curve for beginners, superseded by Angular 2+.

🌟 Features

MVC pattern, directives, filters, routing, services, and a powerful templating engine.



MODULE 2

Setting Up AngularJS

Getting started with AngularJS is straightforward. You can integrate it into any HTML page in minutes using either a CDN link or a local file download.

1

Download or CDN

Get AngularJS from **angularjs.org** or include it via a CDN script tag — no build tools required.

2

HTML Integration

Add the `ng-app` attribute to your HTML element to bootstrap AngularJS in the page.

3

First App

Write your first AngularJS expression, bind a model input, and watch two-way data binding in action live.

💡 **Quick Tip:** Use the CDN approach during learning — it requires zero configuration and works immediately in any browser.

MODULE 3

AngularJS Basics

The core of AngularJS lies in its built-in directives and expression system. These are the building blocks you'll use in every application you create.



ng-app

Defines the root element of the AngularJS application. Every AngularJS app must have this directive to bootstrap the framework.



ng-model

Binds the value of an HTML input element to a variable in the `$scope`, enabling two-way data binding.



ng-bind

Replaces the content of an HTML element with the value of a given expression — a one-way binding alternative to `{{ }}`.



ng-init

Initializes application data directly in the HTML template. Useful for setting default values without a controller.

AngularJS **expressions** (written as `{{ expression }}`) evaluate data and display results in the view, supporting arithmetic, strings, objects, and arrays.

MODULE 4

Directives in AngularJS

Directives extend HTML with custom behavior. AngularJS ships with a rich set of built-in directives, and you can also create your own for reusable UI logic.



ng-repeat

Loops over a collection and renders HTML for each item — the go-to directive for rendering lists and tables.



ng-if

Conditionally renders DOM elements. Removes or recreates elements based on a Boolean expression.



ng-show / ng-hide

Toggles element visibility using CSS without removing the element from the DOM.



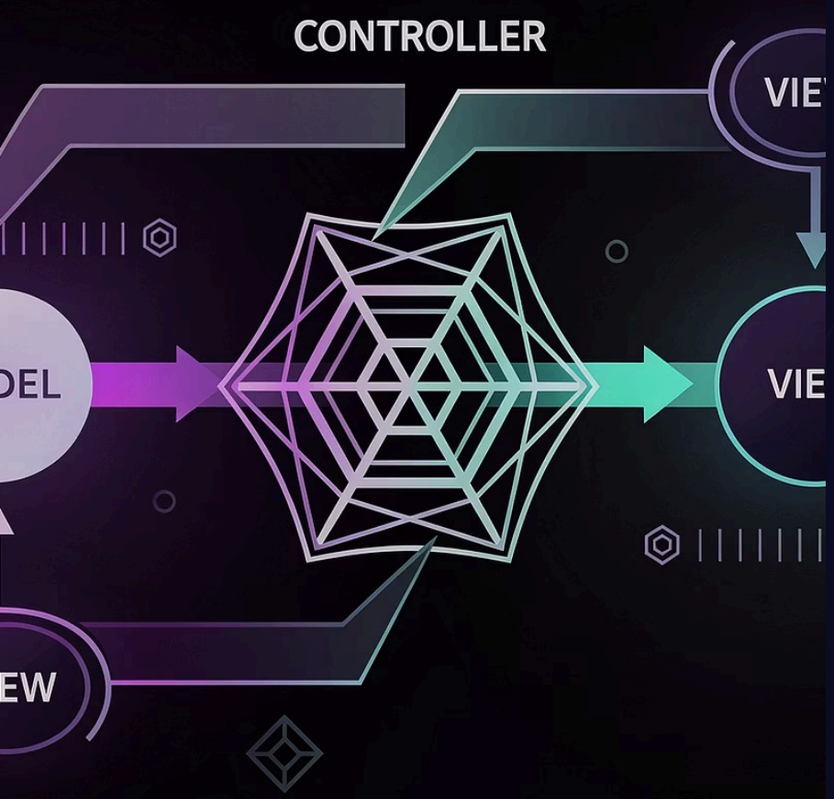
ng-class / ng-style

Dynamically apply CSS classes or inline styles based on scope variables or expressions.



Custom Directives

Create reusable HTML components with custom behavior using the `directive()` method on a module.



MODULE 5

Controllers and Scope

Controllers are the backbone of AngularJS applications. They connect your data model with the view through the `$scope` object — AngularJS's glue between controller logic and HTML templates.

Controller Concept

A controller is a JavaScript constructor function that augments the `$scope` with properties and methods used by the view.

$scope Object

The `$scope` is the data-binding glue. Variables defined on it are accessible in the HTML template and update automatically when changed.

Nested Controllers

Controllers can be nested to create hierarchical scopes. Child scopes inherit from parent scopes through prototypal inheritance.

Controller Methods

Functions defined on `$scope` inside a controller can be called directly from the HTML using `ng-click` and other event directives.

MODULE 6

Modules in AngularJS

What is a Module?

A module is a **container for the different parts** of your application — controllers, services, filters, directives, and configuration. Every AngularJS application has at least one module.

Create a module using:

```
angular.module('myApp', []);
```

The second argument is an array of **module dependencies** — other modules your app relies on.

Why Modules Matter

- **Separation of concerns** — each feature can live in its own module for cleaner architecture.
- **Reusability** — modules can be shared across multiple applications without modification.
- **Testability** — isolated modules are easier to unit test independently.
- **Dependency management** — declare what your module needs and AngularJS handles injection automatically.

MODULE 7

Forms in AngularJS

AngularJS provides powerful, built-in form handling that goes far beyond standard HTML forms. Two-way data binding and declarative validation make form development faster and more reliable.



Form Setup & Two-Way Binding

Use `ng-model` on inputs to bind form fields directly to `$scope` variables — changes in the input instantly update the model and vice versa.



Built-In Validation

AngularJS adds validation states automatically. Use `required`, `ng-minlength`, `type="email"`, and `type="number"` to enforce input rules declaratively.



Showing Error Messages

Use `ng-show` combined with form state properties like `$invalid`, `$dirty`, and `$touched` to display contextual error messages to users.

MODULE 8

Filters in AngularJS

Filters format and transform data before displaying it in the view. They can be applied directly in templates using the pipe `|` syntax, or programmatically inside controllers.



Text Filters

`uppercase` and `lowercase` transform strings. Example: `{{ 'hello' | uppercase }}` outputs **HELLO**.



currency & date

Format numbers as currency (`$1,234.56`) or dates in custom patterns like `'MM/dd/yyyy'` with a single pipe.



orderBy & filter

`orderBy` sorts arrays by a property. The `filter` filter searches and returns matching items from a collection — perfect for search boxes.



Custom Filters

Register a custom filter using `module.filter('name', fn)` to encapsulate any transformation logic and reuse it across templates.

MODULE 9

Services and Factories

Services are singleton objects that provide shared functionality across your application — perfect for business logic, data fetching, and utilities that multiple controllers need to access.

Built-In Services

AngularJS ships with services like `$http`, `$timeout`, `$interval`, and `$location` for common tasks.



Factory

A factory returns an object or function. It's the most flexible recipe — ideal when you need to compute or configure what gets returned.

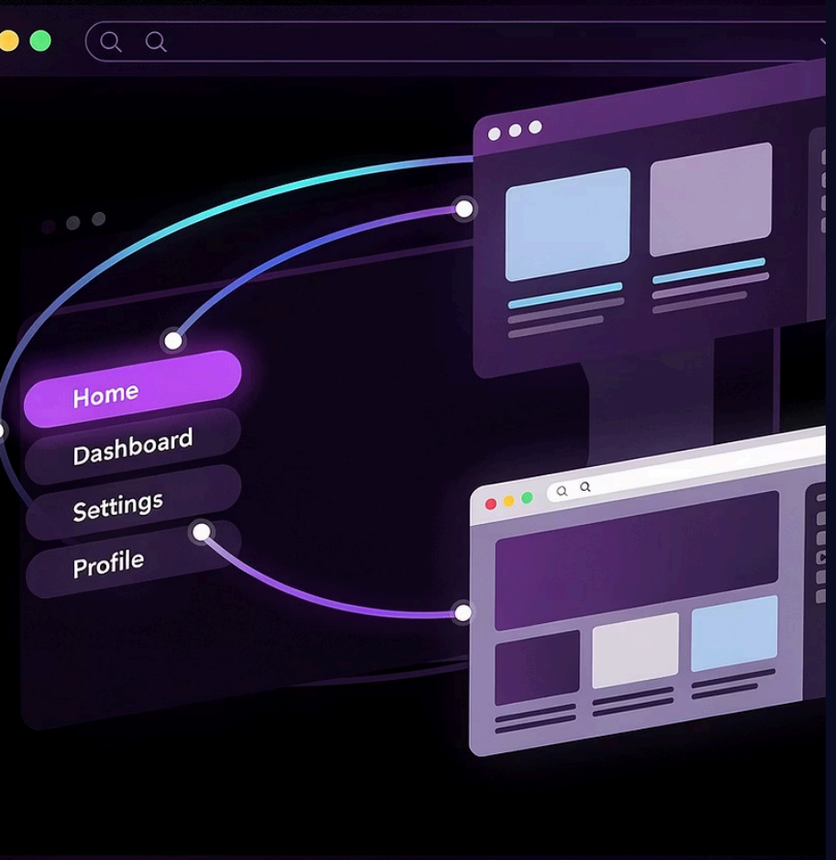
Service

The `service` recipe instantiates using `new`. You define properties on `this` — great for class-style patterns.



Provider

The most powerful recipe — configurable before the app starts. Factories and services are built on top of providers internally.



MODULE 10

Routing & Single Page Applications

Routing is what makes AngularJS apps truly single-page. Instead of full page reloads, `ngRoute` swaps views dynamically — giving users a fast, app-like experience.

1. **Include ngRoute**

   Add the `angular-route.js` script and declare `ngRoute` as a module dependency.

2. **Configure Routes**

   Use `$routeProvider` to map URL paths to templates and controllers in the `.config()` block.

3. **Route Parameters**

   Pass dynamic segments in URLs using `:id` syntax and read them via `$routeParams` in the controller.

4. **Multiple Views**

   Use `ng-view` as a placeholder in your layout — AngularJS injects the correct template for each route automatically.

MODULE 11

HTTP and API Calls

The $http Service

AngularJS's built-in `$http` service is used to communicate with remote servers over HTTP. It returns a **promise** that resolves with the response data.

Supported methods:

- GET
- POST
- PUT
- DELETE

Fetching Data from a Server

```
$http.get('/api/students')
  .then(function(response) {
    $scope.students = response.data;
  })
  .catch(function(error) {
    console.error('Error:', error);
  });
```

Error Handling

Use `.catch()` to handle failed requests gracefully. Always inform users when data cannot be loaded and consider retry strategies for critical data.

MODULE 12

Events in AngularJS

AngularJS makes it easy to respond to user interactions by attaching event handlers directly in the template using built-in event directives — no manual `addEventListener` calls needed.



Click & Mouse Events

Use `ng-click`, `ng-dblclick`, `ng-mouseover`, and `ng-mouseleave` to respond to mouse interactions. Each directive calls a `$scope` function.



Keyboard Events

`ng-keydown`, `ng-keyup`, and `ng-keypress` capture keyboard input. Combine with `$event` to inspect the specific key pressed.



Form Events

`ng-submit` handles form submissions cleanly. `ng-change` fires whenever an input value changes — great for live search or instant validation feedback.



Event Handling in Controller

Define handler functions on `$scope` in the controller. The `$event` object can be passed from the template to access native event properties.

MODULE 13

Advanced Concepts

Mastering these advanced topics separates competent AngularJS developers from exceptional ones. These are the internals that power everything under the hood.



Dependency Injection

AngularJS's DI system automatically provides services to controllers and other components. Declare dependencies as function parameters and AngularJS resolves them at runtime.



Digest Cycle

The digest cycle is AngularJS's change detection mechanism. It iterates over all `$watch` listeners and updates the DOM when scope values change.



$watch & $rootScope

`$watch` monitors scope variables for changes. `$rootScope` is the top-level scope accessible across the entire application for shared state.



Promises & Custom Components

AngularJS uses `$q` for promise-based async programming. Custom components encapsulate templates and logic into reusable, self-contained units.

MODULE 14

Project Work

Apply everything you've learned by building five real-world applications. Each project is designed to reinforce specific skills while producing a portfolio-worthy result.

✅ To-Do List App

Build a task manager with add, complete, and delete features using `ng-repeat` and two-way binding.

🔐 Login Form

Create a login form with real-time validation for email format, required fields, and password strength feedback.

🧮 Marks Calculator

Build a student grade calculator that computes totals, averages, and letter grades dynamically from form inputs.

🗃️ CRUD Application

A full create-read-update-delete app using `$http`, controllers, and routing for multi-view navigation.

🔍 Product Catalog

A product listing with live search using the `filter` filter and `orderBy` for sorting by price or name.

Your AngularJS Journey Starts Here

You now have a complete roadmap — from writing your first `ng-app` to deploying a full CRUD single-page application. Each module builds on the last, giving you a solid, practical foundation in AngularJS development.

14	5	40+
Modules	Projects	Topics
Covering every essential AngularJS concept	Real-world apps to build your portfolio	From basics to advanced internals

🚀 **Ready to begin?** Start with Module 1 and work through each module sequentially. The skills compound — every concept you learn accelerates the next one.