Angular

Duration: 5 days

Description:

Angular is a powerful JavaScript framework for building dynamic web applications. As part of our course outline, you will dive deep into the world of Angular technology. You will learn the fundamentals of Angular, including its syntax, data binding, and directives. Gain hands-on experience in creating robust and scalable web applications using Angular's features and tools. Master the art of building responsive user interfaces and harness the full potential of Angular to create modern, interactive web experiences. By the end of this course, you will be equipped with the knowledge and skills to confidently develop applications using Angular technology.

Learning Objectives:

- Understand how Angular is different than traditional web development frameworks
- Code using TypeScript language features
- Develop an application from scratch using Angular
- Understand and use Angular Forms, Observables, Dependency Injection, and Routing
- Retrieve, update, and delete data using Angular's Http service
- Create, build, and deploy an Angular application using the Angular CLI
- Develop dynamic Model-driven forms that are easier to unit test
- Reactive Programming Observable with RxJs

Prerequisites:

- All participants should have experience developing Web Applications
- Participants should have good knowledge of JavaScript
- Understanding of Web Services would be an added advantage

Contents

- Template Input Variables
- The NgSwitch Directives
- Template Reference Variables
- Input and output properties Components Deep Dive
- Component Lifecycle Hooks
- Implementing the Lifecycle methods
- Component Communication
- Sharing data between components

Services

- Creating Services
- Using a services to access data
- Using a service to encapsulate business logic

Dependency Injection

- Understanding Dependency Injection
- Angular's Dependency Injection System
- Registering
- Injecting

Template-driven Forms

- NgSubmit Directive
- FormsModule
- NgForm, NgModel, and NgModelGroup Directives
- Validation Directives

RXJS

- Reactive Programming
- Observables and Observer
- Operators
- Errors and Exceptions

Communicating with the Server using the Http Service

- Deciding between Promises or Observables (RxJS)
- Making Http GET Requests
- Making Http POST and PUT Requests
- Issuing a Http DELETE Request
- Consuming RESTful Services
- WebSockets(Introduction)

Single Page Applications using Router

- Importing the RouterModule and Routes
- Configuring Routes
- Displaying Components using a RouterOutlet
- Navigating with RouterLink and RouterLinkActive Directives or the Router
- Accessing parameters using ActivedRoute
- Organizing your code into Modules
- Lazy-loading Angular Modules

- Location Strategies
- Nested or Child Routes
- Route Guards

Security

- How to Prevent Cross-site Scripting (XSS)
- Trusting values with the DOMSanitizer
- HTTP Attacks

Advanced Components

- Component Styles using MetaData properties: Styles and StyleUrls
- Change Detection Strategies
- Component Lifecycle Hooks

Model-driven Forms (Reactive Forms)

- ReactiveFormsModule
- AbstractControl, FormControl, FormGroup, and FormArray
- FormBuilder
- Validators
- Creating Dynamic Forms

Attribute Directives

- Creating a custom Attribute Directive using ElementRef, Render
- Creating a custom structural Directive

Pipes

- Built-in Pipes: Using, Passing Parameters, Chaining
- Creating a custom Pipe using PipeTransform
- Pure and Impure pipes

Web Workers

Ivy and Bazel

Differential Loading

Testing

- Understanding Jasmine.js
- Writing specifications in Jasmine.js
- Learning built-in matchers
- Covering before and after
- Use Karma for browser testing
- Testing Angular components
- Creating End to End testing with protractor
- Configuring and using Karma

Creating, Building, and Deploying an Angular Application

- Manually
- Using the Angular CLI with Ahead-Of-Time (AOT) Compilation

- Pre-loading and lazy loading

Using Libraries

- Exploring Angular UI Components
    - Ng-bootstrap
    - PrimeNG
    - Angular Material