Technical course content

Contents

Angular 2
The Basics 2
Debugging 2
Components & Databinding Deep Dive 2
Directives Deep Dive 3
Using Services & Dependency Injection 3
Changing Pages with Routing 4
Understanding Observables 4
Handling Forms in Angular Apps 5
Using Pipes to Transform Output 6
Making Http Requests 6
Authentication & Route Protection in Angular 6
Deploying an Angular App 7
Unit Testing in Angular Apps 7
Angular Material 7
AG-grid usage (3rd party package) 8
State management 8
Angular style guide 8
Web API 9
Framework 9
Introduction to Web API 9
Getting Started with the project 9
Building the Web API 9
Security 9
Advanced topics 9
C# Advanced topics 9
3rd Party nuget packages usage 10
Optimization 10
Test Driven Development (TDD) 10

Angular

The Basics

- Module Introduction
- How an Angular App gets Loaded and Started
- Components are Important!
- Creating a New Component
- Understanding the Role of AppModule and Component Declaration
- Using Custom Components
- Creating Components with the CLI & Nesting Components
- Working with Component Templates
- Working with Component Styles
- Fully Understanding the Component Selector
- What is Databinding?
- String Interpolation
- Property Binding
- Property Binding vs String Interpolation
- Event Binding
- Bindable Properties and Events
- Passing and Using Data with Event Binding
- Important: FormsModule is Required for Two-Way-Binding!
- Two-Way-Databinding
- Combining all Forms of Databinding
- Understanding Directives
- Using ngIf to Output Data Conditionally
- Enhancing ngIf with an Else Condition
- Styling Elements Dynamically with ngStyle
- Applying CSS Classes Dynamically with ngClass
- Outputting Lists with ngFor
- Getting the Index when using ngFor

Debugging

- Understanding Angular Error Messages
- Debugging Code in the Browser Using Sourcemaps

Components & Databinding Deep Dive

- Module Introduction
- Splitting Apps into Components
- Property & Event Binding Overview

- Binding to Custom Properties
- Assigning an Alias to Custom Properties
- Binding to Custom Events
- Assigning an Alias to Custom Events
- Custom Property and Event Binding Summary
- Understanding View Encapsulation
- More on View Encapsulation
- Using Local References in Templates
- @ViewChild() in Angular 8+
- Getting Access to the Template & DOM with @ViewChild
- Projecting Content into Components with ng-content
- Understanding the Component Lifecycle
- Seeing Lifecycle Hooks in Action
- Lifecycle Hooks and Template Access
- @ContentChild() in Angular 8+
- Getting Access to ng-content with @ContentChild

Directives Deep Dive

- Module Introduction
- ngFor and ngIf Recap
- ngClass and ngStyle Recap
- Creating a Basic Attribute Directive
- Using the Renderer to build a Better Attribute Directive
- More about the Renderer
- Using HostListener to Listen to Host Events
- Using HostBinding to Bind to Host Properties
- Binding to Directive Properties
- What Happens behind the Scenes on Structural Directives
- Building a Structural Directive
- Understanding ngSwitch

Using Services & Dependency Injection

- Module Introduction
- Why would you Need Services?
- Creating a Logging Service
- Injecting the Logging Service into Components
- Creating a Data Service

- Understanding the Hierarchical Injector
- How many Instances of Service Should It Be?
- Injecting Services into Services
- Using Services for Cross-Component Communication
- Services in Angular

Changing Pages with Routing

- Module Introduction
- Why do we need a Router?
- Understanding the Example Project
- Setting up and Loading Routes
- Navigating with Router Links
- Understanding Navigation Paths
- Styling Active Router Links
- Navigating Programmatically
- Using Relative Paths in Programmatic Navigation
- Passing Parameters to Routes
- Fetching Route Parameters
- Fetching Route Parameters Reactively
- Passing Query Parameters and Fragments
- Retrieving Query Parameters and Fragments
- Setting up Child (Nested) Routes
- Using Query Parameters - Practice
- Configuring the Handling of Query Parameters
- Redirecting and Wildcard Routes
- An Introduction to Guards
- Protecting Routes with canActivate
- Protecting Child (Nested) Routes with canActivateChild
- Using a Fake Auth Service
- Controlling Navigation with canDeactivate
- Passing Static Data to a Route
- Resolving Dynamic Data with the resolve Guard
- Understanding Location Strategies

Understanding Observables

- Module Introduction
- Install RxJS

- Analyzing Angular Observables
- Getting Closer to the Core of Observables
- Building a Custom Observable
- Errors & Completion
- Observables
- Understanding Operators
- Subjects

Handling Forms in Angular Apps

- Module Introduction
- Template-Driven (TD) vs Reactive Approach
- TD: Creating the Form and Registering the Controls
- TD: Submitting and Using the Form
- TD: Understanding Form State
- TD: Accessing the Form with @ViewChild
- TD: Adding Validation to check User Input
- Built-in Validators & Using HTML5 Validation
- TD: Using the Form State
- TD: Outputting Validation Error Messages
- TD: Set Default Values with ngModel Property Binding
- TD: Using ngModel with Two-Way-Binding
- TD: Grouping Form Controls
- TD: Handling Radio Buttons
- TD: Setting and Patching Form Values
- TD: Using Form Data
- TD: Resetting Forms
- Practicing Template-Driven Forms
- Introduction to the Reactive Approach
- Reactive: Setup
- Reactive: Creating a Form in Code
- Reactive: Syncing HTML and Form
- Reactive: Submitting the Form
- Reactive: Adding Validation
- Reactive: Getting Access to Controls
- Reactive: Grouping Controls
- Reactive: Arrays of Form Controls (FormArray)
- Reactive: Creating Custom Validators
- Reactive: Using Error Codes
- Reactive: Creating a Custom Async Validator
- Reactive: Reacting to Status or Value Changes
- Reactive: Setting and Patching Values

Using Pipes to Transform Output

- Introduction & Why Pipes are Useful
- Using Pipes
- Parametrizing Pipes
- Chaining Multiple Pipes
- Creating a Custom Pipe
- Parametrizing a Custom Pipe
- Understanding the "async" Pipe

Making Http Requests

- Module Introduction
- How Does Angular Interact with Backends?
- The Anatomy of a Http Request
- Backend Setup
- Sending a POST Request
- GETting Data
- Using RxJS Operators to Transform Response Data
- Using Types with the HttpClient
- Outputting Posts
- Showing a Loading Indicator
- Using a Service for Http Requests
- Services & Components Working Together
- Sending a DELETE Request
- Handling Errors
- Using Subjects for Error Handling
- Using the catchError Operator
- Error Handling
- Setting Headers
- Adding Query Params
- Introducing Interceptors
- Manipulating Request Objects
- Response Interceptors

Authentication & Route Protection in Angular

- Module Introduction
- How Authentication Works

- Adding the Auth Page
- Creating & Storing the User Data
- Reflecting the Auth State in the UI
- Adding the Token to Outgoing Requests
- Attaching the Token with an Interceptor
- Adding Logout
- Adding an Auth Guard

Deploying an Angular App

- Module Introduction
- Deployment Preparation & Steps
- Using Environment Variables
- Deployment Example

Unit Testing in Angular Apps

- Introduction to the Jasmine testing framework
- Introduction to Jasmine spies
- Unit Testing of simple Angular Services
- Jasmine testing best practices
- Introduction to Angular testing utilities
- Testing of complex Angular services
- Mocking of Angular HTTP requests
- Unit Testing of Angular Components
- Asynchronous Angular Testing with fakeAsync and async
- Mocking of Observable-based Services
- Unit Testing of Presentational Components
- Unit Testing of Smart or Container Components
- Simulation of user interaction in unit tests
- Angular CLI Test Coverage Reports

Angular Material

- Set up Angular Material from scratch
- navigation and containers, side menu, tab container
- display data with cards, use some buttons
- commonly used form controls and buttons - create course form
- inputs and text areas
- radio buttons, checkboxes, tooltips
- date picker, date/time formatting
- Displaying form errors
- Angular Data Table in depth
- Expandable table rows
- pagination and loading indicator
- Overlays, best practices to use them
- Drag and Drop

- Responsive design with mat-grid-list
- Responsive breakpoint observers
- Tree component
- Virtual scrolling - handling large amounts of data
- Custom Themes

AG-grid usage (3rd party package)
https://www.ag-grid.com/angular-data-grid/

- Interface/API
- Columns
- Rows
- Layout & styling
- Client-Side data
- Selection
- Filtering
- Rendering
- Editing
- Import & Export
- Accessories
- Components
- Scrolling

State management
https://ngrx.io/docs

- Introduction to State Management
- The Store Architecture in Detail
- NgRx Key Concepts
- Actions and Action Creators
- Reducers
- NgRx Effects
- Selectors
- Adding Authentication to an NgRx Application
- NgRx Entity and the Entity Format
- NgRx DevTools
- NgRx Time Travelling Debugger
- NgRx Runtime checks and Store Immutability
- NgRx Router Store
- NgRx Data and Entity State Management
- NgRx Best Practices

Angular style guide
https://angular.io/guide/styleguide

Web API

Framework

- Dotnet 5 or 6
- C# 8, 9, 10 features

Introduction to Web API

- Understanding REST
- ASP.net WEB API
- Micro services development

Getting Started with the project

- Create new Web API with latest dotnet stable version
- Web API controllers
- Serialization
- Routing

Building the Web API

- HTTP Verbs
- GET, POST, PUT, DELETE, PATCH implementation
- URI Mapping
- Filters & Attributes
- Model binding & formatters

Security

- SSL usage
- CORS
- Authentication
- Authorization
- OAUTH

Advanced topics

- Dependency Injection
- HTTP caching
- Redis caching usage
- API versioning
- Entity framework core
- LINQ usage

C# Advanced topics

- Generics
- Lamba expressions
- LINQ
- Extension methods
- Nullable types
- Dynamic
- Exception Handling
- Asynchronous programming with Asyn / Await
- Parallel programming with threads

3rd Party nuget packages usage

- Logging **-** Serilog, log4net, nlog implementation
- Fluent validation
- NSwag for Open API

Optimization

- Code optimization
- Memory optimization

Test Driven Development (TDD)

- TDD with unit tests (XUnit)