**Angular: The Big Picture**

***Introduction***

Angular, What, and Why

Hello Angular

Versions of Angular

***Benefits & Features of Angular***

Universal Benefits

Subjective & Situational Benefits

Basic Features

Advanced Features

***Angular Architecture***

One Way Data Flow

Dependency Injection

Components

Directives

Templates

Zone.js & Change Detection

Rendering Targets

***Tooling***

The Angular CLI

Server-side Rendering

Mobile & Native Frameworks

Testing Tools

AOT Compiler

Editors

***Tips, Tricks, & Gotchas***

Gotchas

Tips & Tricks

***Angular: Present & Future***

Present

Angular vs. Other Frameworks

Future

**Angular: Getting Started**

***Introduction***

Anatomy of an Angular Application

Get the Most from This Course

Sample Application

Course Outline

***First Things First***

Selecting a Language

Selecting an Editor

Setting up Our Environment

Setting up an Angular Application

Installing an Angular Application

Running an Angular Application

About Modules

***Introduction to Components***

What Is a Component?

Creating the Component Class

Defining the Metadata with a Decorator

Importing What We Need

Demo: Creating the App Component

Bootstrapping Our App Component

Demo: Bootstrapping Our App Component

***Templates, Interpolation, and Directives***

Building a Template

Building the Component

Using a Component as a Directive

Binding with Interpolation

Adding Logic with Directives: ngIf

Adding Logic with Directives: ngFor

***Data Binding & Pipes***

Property Binding

Handling Events with Event Binding

Handling Input with Two-way Binding

Transforming Data with Pipes

***More on Components***

Defining Interfaces

Encapsulating Component Styles

Using Lifecycle Hooks

Building Custom Pipes

Filtering a List

Checklists and

8. Building Nested Components

Introduction

Building a Nested Component

Using a Nested Component

Passing Data to a Nested Component Using @Input

Passing Data from a Component Using @Output

***Services and Dependency Injection***

How Does It Work?

Building a Service

Registering the Service

Injecting the Service

***Retrieving Data Using HTTP***

Observables and Reactive Extensions

Sending an HTTP Request

Demo: Sending an Http Request

Exception Handling

Subscribing to an Observable

Demo: Subscribing to an Observable

***Navigation and Routing Basics***

Generating Code and Handling Undefined

How Routing Works

Configuring Routes

Demo: Configuring Routes

Tying Routes to Actions

Placing the Views

***Navigation and Routing Additional Techniques***

Passing Parameters to a Route

Demo: Passing Parameters to a Route

Activating a Route with Code

Protecting Routes with Guards

Demo: Protecting Routes with Guards

***Angular Modules***

What Is an Angular Module?

Bootstrap Array

Declarations Array

Exports Array

Imports Array

Providers Array

Feature Modules

Demo: Feature Modules

Shared Module

Revisiting AppModule

***Building, Testing, and Deploying with the CLI***

Angular CLI Overview

ng new

ng serve

ng generate

ng test

ng build

***Final Words***

Recapping Our Journey

Learning More

What Is Angular? (Revisited)

Closing

**Angular Fundamentals**

***Getting Started with Angular***

Practice Exercises

Introduction to Typescript

Comparing Angular to AngularJS

A Conceptual Overview of Angular

Here's What We'll Be Building

Installing Git and Node

Getting Started with the Angular CLI

Bootstrapping an Angular App

A Brief Look at the App Module

Accessing Static Files

***Creating and Communicating Between Angular Components***

Creating Your First Data-bound Component

Using External Templates

Communicating with Child Components Using @Input

Communicating with Parent Components Using @Output

Using Template Variables to Interact with Child Components

Styling Components

Exploring Angular’s CSS Encapsulation

Adding a Site Header

***Exploring the Angular Template Syntax***

Interpolation, Property Bindings, and Expressions

Event Bindings and Statements

Repeating Data with ngFor

Handling Null Values with the Safe-Navigation Operator

Hiding and Showing Content with ngIf

Hiding Content with the [Hidden] Binding

Hiding and Showing Content with ngSwitch

Styling Components with ngClass

Styling Components with ngStyle

***Creating Reusable Angular Services***

Why We Need Services and Dependency Injection

Creating Your First Service

Wrapping Third Party Services

***Routing and Navigating Pages***

Adding Multiple Pages to Your App

Adding Your First Route

Accessing Route Parameters

Linking to Routes

Navigating from Code

Guarding Against Route Activation

Guarding Against Route De-activation

Pre-loading Data for Components

Styling Active Links

Lazily Loading Feature Modules

Organizing Your Exports with Barrels

***Collecting Data with Angular Forms and Validation***

Using Models for Type Safety

Creating Your First Template-based Form

Using the Data from Your Template-based Form

Validating Template-based Forms

Creating Your First Reactive Form

Validating Reactive Forms

Using Multiple Validators in Reactive Forms

Diving Deeper into Template-based Forms

Editing Data with Two-way Bindings

Diving Deeper into Reactive Forms

Creating Custom Validators

***Communicating Between Components***

Passing Data into a Child Component

Passing Data out of a Child Component

***Reusing Components with Content Projection***

Content Projection

Multiple Slot Content Projection

***Displaying Data with Pipes***

Using Built-in Pipes

Creating a Custom Pipe

Sorting and Filtering Overview

Creating a Filtering Display

Filtering Data

Sorting Data

***Understanding Angular's Dependency Injection***

Using Third Party Global Services - The Problem

Angular Dependency Injection Lookup

Using Angular's InjectionToken

Using Angular's @Inject Decorator

The useClass Provider

The useExisting and useFactory Providers

***Creating Directives and Advanced Components in Angular***

Implementing the Session Search

Adding jQuery

Creating a Modal Component

Fixing Template Parse Errors

Creating Directives - The Trigger Directive

Binding an ID

Routing to the Same Component

Using the @ViewChild Decorator

Creating Settings on Components

***More Components and Custom Validators***

Creating a Voting Component

Adding Voting Functionality

Hiding Functionality Before Authentication

Using @Input Setters

Creating a Custom Validator

Adding a Validator to Angular's Validators

Implementing a Multi-field Validator

***Communicating with the Server Using HTTP, Observables, and Rx***

Preparing to Store Data on the Server

Moving Data Storage to the Server

Listening to Resolved Data Changes

Using POST and PUT

Using QueryString Parameters

Using DELETE

Integrating Authentication with the Server

Persisting Authentication Status across Page Refreshes

Saving User Data to the Server

Implementing Logout

***Unit Testing Your Angular Code***

Installing Karma

Unit Testing Services

Testing Mock Calls

Testing Components with Isolated Tests

***Testing Angular Components with Integrated Tests***

Setting up for Integrated Tests

Testing Components with Deep Integrated Tests

Creating Mock Services

Using DebugElement

Testing Components with Shallow Integrated Tests

***Taking an Angular App to Production***

Linting Overview

Installing TSLint in VSCode

Using TSLint with VSCode

Linting from the Command Line

Going to Production - Overview

Angular's Ahead of Time Compiler

Creating your First Build

Basic Deployment

Demo: Serving up a Build

Build Flags

The Effects of Prod Mode

Optimistic Bundle Downloading

**Angular CLI**

***Overview***

Introduction

Motivation

Sneak Peek inside the Angular CLI

Test Driving the Angular CLI

Looking Ahead

***Angular CLI Setup and Verification***

Getting Started with the Angular CLI

Angular CLI Requirements

Installing and Verifying the Angular CLI

Angular CLI Installation

Generating and Exploring a New App

Angular Extensions for VS Code

Angular CLI Resources

***Generating a New Angular Application***

Overview

Commands for Generating a New App

Generating and Customizing a New App

Angular CLI JSON File

Common App Generation Flags

Configuring the Angular CLI

Linting

What's Next

***Generating Code from Blueprints***

Angular CLI Code Generation

Blueprints

Component Blueprints and Options

Generating Components

Declaring a Component in an NgModule

Generating Directives

Generating Services

Generating Classes, Interfaces, and Enums

Generating Pipes

Generating NgModules

Configuration and Resources for Out-of-the-Box Blueprints

***Generating Routing Features***

The NgModule Blueprint with Routing

Generating a New Routing App

Generating a New Module with Routing

Generating a Guard

What's Next?

***Building and Serving***

Building Angular

Exploring a Development Build and Its Bundles

Building Target and Environments

Production Builds

Serving Angular

Exploring Angular Serving Options

Adding New Capabilities

Adding Angular Material

Adding Scripts, Styles, and Assets

What's Next?

***Running Unit and End to End Tests***

Running Tests with the Angular CLI

Testing Angular

Executing Unit Tests

CLI Testing Options

Code Coverage

Debugging Unit Tests

End to End Tests

Executing End to End Tests

Wrap Up

***Tooling Features***

Updating Angular

Running ng update

Updating Our Demo

Multiple Projects

Workspaces

Generating Libraries

Creating a Logger Library

Adding to Angular

Angular Console

Wrap Up

**Angular Forms**

***Introduction***

Angular's Form Technologies

Creating the Demo Application

***Form Basics in Angular***

Angular's FormsModule

Creating a Form Component

Using Bootstrap for Styling

Checkboxes and Radios

Select and Options

Other Form Controls

***Data Binding in Angular Forms***

Using NgForm

NgModel

Creating a Data Model

Two-way Data Binding

Copying Form Data

***Form Validation***

HTML5 Field Validation

CSS Classes for Validation

NgModel Properties for Validation

Styling Forms with Validation Errors

Submitting Forms

Handling Form Control Events

***HTTP Form Posting and Data Access***

Creating a Data Service

Form Posting Using Observables

HTTP Access Using HttpClient

Posting a Form

Handling POST Errors

Retrieving Data for Select Elements

***Third-party Form Controls***

Form Resources at angular.io

Installing and using ngx-bootstrap

Working with Buttons

Dates and Date Ranges

Timepicker

Rating Control

**Angular Reactive Forms**

***Introduction***

Angular Forms

Get the Most from This Course

Demo Form and Sample Application

Course Outline

***Template-driven vs. Reactive Forms***

Form Building Blocks

Form Directives

Template Syntax for Forms

Template-driven Form

Template-driven Form: Template

Template-driven Form: Component

Complex Scenarios

***Building a Reactive Form***

The Component Class

The Component Class: Demo

The Angular Module

The Template

The Template: Demo

Using setValue and patchValue

Simplifying with FormBuilder

***Validation***

Setting Built-in Validation Rules

Adjusting Validation Rules at Runtime

Custom Validators

Custom Validation with Parameters

Cross-Field Validation: Nested FormGroup

Cross-field Validation: Custom Validator

***Reacting to Changes***

Watching

Reacting: Adjusting Validation Rules

Reacting: Displaying Validation Messages

Reactive Transformations

***Dynamically Duplicate Input Elements***

Steps

Define the Input Element(s) to Duplicate

Define a FormGroup

Refactor

Create a FormArray

Loop Through the FormArray

Duplicate the Input Elements

***Reactive Forms in Context***

Sample Application

Sample Application: Architecture

Sample Application: Code

Routing to a Form

Reading a Route Parameter

Setting a canDeactivate Guard

Refactoring to a Custom Validation Class

***Create, Read, Update, and Delete (CRUD) Using HTTP***

Data Access Service

Data Access Service: Demo

Faking a Backend Server

Populating the Form with Data

Populating the Form with Data: Http Get

Populating the Form with Data: Subscribe

Saving Edits

Saving Edits: Demo

Creating New Items

Creating New Items: Demo

Deleting an Existing Item

Deleting an Existing Item: Demo

***Final Words***

Recapping Your Journey

Learning More

Closing

**Angular Routing**

***Introduction***

Sample Application: Demo

Sample Application: Architecture

How Routing Works

Get the Most from This Course

Course Outline

***Routing Basics***

Setting up Routing

Defining the Base Path

Importing the Angular Router

Configuring Application Routes

Placing the Template

Activating Routes

Using HTML 5 or Hash-based URLs

Checklist and

***Routing to Features***

Setting up for Feature Routing

Route Path Naming Strategies

Activating a Route with Code

Activating a Route with Code: Demo

Accessing Feature Routes

Defining a Routing Module

***Route Parameters***

Configuring a Parameterized Route

Populating Route Parameters

Populating Route Parameters: Demo

Reading Route Parameters: Snapshot

Reading Route Parameters: Snapshot Demo

Reading Route Parameters: Observable

Defining Optional Route Parameters

Defining Query Parameters

Reading Query Parameters

***Prefetching Data Using Route*** ***Resolvers***

Providing Data with a Route

Using a Route Resolver

Building a Route Resolver Service

Resolver Error Handling

Adding a Resolver to a Route Configuration

Reading Resolver Data - Snapshot

Reading Resolver Data - Observable

***Child Routes***

Using Child Routes

Configuring Child Routes

Placing the Child View

Activating Child Routes

Obtaining Data for Child Routes

Validating Across Child Routes

***Grouping and Component-less Routes***

Grouping Routes

Component-less Routes

***Styling, Animating, and Watching Routes***

Styling the Selected Route

Animating Route Transitions

Watching Routing Events

Reacting to Routing Events

***Secondary Routes***

Using Secondary Routes

Defining a Named Router Outlet

Configuring Secondary Routes

Activating Secondary Routes: RouterLink

Activating Secondary Routes: In Code

Clearing Secondary Outlets

***Route Guards***

Using Route Guards

CanActivate Guard

Sharing Data with a Guard

CanActivateChild Guard

CanDeactivate Guard

***Lazy Loading***

Building and Serving Our Files

Preparing for Lazy Loading

Lazy Loading

CanLoad Guard

Preloading Feature Modules

Preloading Feature Modules: Demo

Custom Preloading Strategy

***Final Words***

Recapping Our Journey

Learning More

**Angular Component Communication**

***Course Overview***

***Introduction***

Get the Most from This Course

Sample Application

Course Outline

***Communicating with a Template***

Binding and Structural Directives

Notifying the Component of User Changes

Two-way Binding, the Long Way

Getters and Setters

Guidelines and Summary

***ViewChild and ViewChildren***

ViewChild

Demo: ViewChild and Accessing the Native Html Element

ViewChildren

ViewChild and Angular Forms

Demo: ViewChild and Angular Forms

ViewChild and ngIf

Guidelines and Summary

***Communicating with a Child Component***

Building Child Components

Parent to Child Communication

Input Property

Watching for Changes

Template Reference Variable

ViewChild Decorator

Guidelines and Summary

***Communicating with a Parent Component***

Introduction

Child to Parent

Output Property

Guidelines and Summary

***Communicating Through a Service***

***Introduction***

Managing State

Component to Itself

Property Bag Service

Component to Component

Service Scope and Lifetime

Guidelines and Summary

***Communicating Through a State Management Service***

Sharing Entity State

State Management Service

Demo: Retrieving State

Demo: Maintaining State

Displaying Concurrent Components

Keeping State in Sync

Change Detection

Guidelines and Summary

***Communicating Through Service Notifications***

Service Notifications

Subject

Demo: Subject

Demo: More on the Subject

Subject vs. BehaviorSubject

Angular Component Communication

Demo: Cleaning up a Subscription

Guidelines and Summary

***Communicating Using the Router***

Routing Basics

Route Parameters

Guidelines and Summary

***Final Words***

Recap

Learning More

Closing

**Angular Http Communication**

***Configuring an Application to Make HTTP Requests***

Introduction and Overview

Demo: Book Tracker Project Overview

The Role of RxJS

Demo: Preparing to Use HttpClient in a Project

***Consuming REST Services***

Introduction and Overview

What is a REST Service?

Subscribing to Observables

Demo: Retrieving a Collection

Demo: Retrieving a Single Item

Using RxJS Operators

Demo: Transforming Data with RxJS

Demo: Creating, Updating, and Deleting Data

***Advanced HTTP Requests and Error Handling***

Introduction and Overview

Handling HTTP Errors

Demo: Handling HTTP Errors

Retrieving Data with Resolvers

Demo: Retrieving Data over HTTP with Resolvers

***Creating Interceptors***

Introduction and Overview

What Are Interceptors?

Uses for Interceptors

Defining and Providing Interceptors

Demo: Creating an Interceptor

Demo: Intercepting Responses and Using Multiple Interceptors

***Caching HTTP Requests with Interceptors***

Introduction and Overview

Benefits and Types of Caching

Caching with Interceptors

Demo: Create a Service to Store Cached HTTP Requests

Demo: Create an Interceptor to Cache HTTP Requests

***Testing HTTP Requests***

Introduction and Overview

Angular's Default Unit Testing Tools

Structure of Angular HTTP Unit Tests

Demo: Testing HTTP Requests and Responses

Demo: Testing HTTP Errors

**Angular Services**

***The Role Services Play in an Angular Application***

What Is an Angular Service?

Demo: Book Tracker Project Overview

***Creating and Using Services***

Introduction and Overview

Parts of a Service

Delivering Services to Components

Demo: Manually Creating a Logger Service Demo: Creating a Service with the Angular CLI

Sharing Data with Services

Demo: Sharing Data with Service

***Understanding and Configuring Dependency Injection***

Introduction and Overview

What Is Dependency Injection?

Provider Tokens and Recipes

Demo: Multiple Ways to Provide Services

Injectors and Metadata

Hierarchical Injectors

Demo: Hierarchical Injectors

Deciding Where to Provide Services

Demo: Providing Feature Services

Demo: Creating a Core Module

***Creating Asynchronous Services***

Introduction and Overview

What Are Asynchronous Services?

Observables

Demo: Processing an Asynchronous HTTP Request with an Observable

Demo: Abstracting Away HTTP Errors

Promises

Demo: Asynchronously Executing a Task with a Promise

Understanding async/await

Demo: Handling a Promise with async/await

***Consuming Common Built-in Services***

Introduction and Overview

Demo: Finding and Using Built-in Services

Demo: Implementing a Centralized Error Handler

**Angular ngrx: Getting Started**

***Introduction***

What Is NgRx?

Why Use NgRx?

Getting the Most from This Course

Sample Application

Course Outline

***The Redux Pattern***

Store

Actions

Reducers

Advantages of the Redux Pattern

***First Look at NgRx***

Demo: Setting up the Sample Application

Installing the Store

Initializing the Store

Demo: Initializing the Store

Defining the State and Actions

Building a Reducer to Process Actions

Demo: Building a Reducer to Process Actions

Dispatching an Action to Change State

Subscribing to the Store to Get State Changes

***Developer Tools and Debugging***

Installing the Tools

Using the Tools

***Strongly Typing the State***

Defining Interfaces for Slices of State

Extending the State Interface for Lazy Loaded Features

Strongly Typing the State

Setting Initial State Values

Building Selectors

Demo: Building Selectors

Demo: Using Selectors

Composing Selectors

***Strongly Typing Actions with Action Creators***

Building Action Creators

Demo: Building Action Creators

Using Strongly Typed Actions

Demo: Using Strongly Typed Actions

Using Actions and Selectors for Component Communication

Demo: Communicating with the Edit Component

Defining Actions for Complex Operations

Demo: Defining Actions for Complex Operations

***Working with Effects***

What Are Effects?

Defining an Effect

Demo: Defining an Effect

SwitchMap versus MergeMap

Registering an Effect

Using an Effect

Unsubscribing from Observables

Exception Handling in Effects

***Performing Update Operations***

Identifying the State and Actions

Strongly Typing the State and Building Selectors

Strongly Typing the Actions with Action Creators

Dispatching an Action

Building the Effect

Processing the Success and Fail Actions

Demo: Processing the Success and Fail Actions

***Architectural Considerations***

Folder by Feature or Function

Container Presentational Component Pattern

Demo: Container Component

Demo: Presentational Component

Change Detection OnPush

Creating a Barrel with Index.ts Files

***Final Words***

Recap

Additional NgRx Libraries

Learning More

Closing

**RxJS in Angular: Reactive Development**

***Introduction***

What Is RxJS?

What Is Reactive Development?

Getting the Most from This Course

***RxJS Terms and Syntax***

Observer/Subscriber

Observable Stream (Observable)

Starting the Observable Stream/Subscription

Stopping the Observable Stream

Creation Functions

Creation Functions: Demo

***RxJS Operators***

RxJS Operator Overview

RxJS Operator: map

RxJS Operator: tap

RxJS Operator: take

RxJS Operators: Demo

Operator Internals

***Going Reactive***

Sample Application

Working with the Async Pipe

Handling Errors

Handling Errors: Catch and Rethrow

Handling Errors: Demo

Improving Change Detection

Declarative Pattern for Data Retrieval

***Mapping Returned Data***

Mapping an Http Response

Mapping the Emitted Array

Mapping Array Elements

Transforming Array Elements

***Combining Streams***

Combination Operators/Functions

RxJS Creation Function: combineLatest

RxJS Creation Function: forkJoin

RxJS Operator: withLatestFrom

Combining Streams to Map an Id to a String

Combining Streams to Map an Id to a String: Demo

***Reacting to Actions***

Filtering a Stream

Filtering a Stream: Demo

Filtering a Stream: Demo II

Data Stream vs. Action Stream

Subject and BehaviorSubject

Reacting to Actions

Starting with an Initial Value

***Reacting to Actions: Examples***

Reacting to a Selection: Data Stream

Reacting to a Selection: Action Stream

Reacting to an Error

Reacting to an Add Operation

Reacting to an Add Operation: Demo

***Caching Observables***

Why Caching?

Patterns for Data Caching

Patterns for Data Caching: Demo

Summary and Considerations

***Higher-order Mapping Operators***

Higher-order Mapping Operators

RxJS Operator: concatMap

concatMap: Demo

RxJS Operator: mergeMap

mergeMap: Demo

RxJS Operator: switchMap

switchMap: Demo

***Combining All the Streams***

Related Data Streams

Related Data Streams: Get It All

Related Data Streams: Just in Time

Just in Time: Demo

Get It All vs. Just in Time

Ancillary Streams

Combining All the Streams

***Final Words***

Recapping Our Journey

Key Points, Tips, and Common Issues

A Few More Terms

**Angular Redux Manage State**

***Introduction to Redux***

What You'll Learn in This Course

Getting Your Starter Code up and Running

Constructing a Redux Store

Subscribing to State Changes and Dispatching Actions

Calculating New State in Your Reducer Function

Creating and Configuring Redux Middleware

***Moving State to Redux***

Getting the Existing Angular 2 App up and Running

Implementing a Basic Integration of Redux and Angular 2

Configuring the Redux Developer Tools

Handling User Input Through a Redux Action

Integrating with Redux Using NgRedux

Defining Actions with Access to Injected Services

Calling into Angular Services from Redux Actions

***Immutability and Testing***

The Problem with Unexpected State Mutation

Making Objects Read-only

Enforcing Immutability

Testing a Reducer's Initial State'

**Structuring Angular Application with Libraries**

***Introduction***

Getting the Most out of This Course

Demo: Introducing the Sample App

Course Outline

***Understanding the Problem of Growing Applications***

Introduction

Architectural Blocks of an Angular Application

Placing Libraries in a Growing Codebase

***Building a Typescript Library***

Introduction Version

Setting up the Environment

Introduction to the Javascript Module System

Demo: Creating Your First Javascript Module

Short Comparison of Javscript Modules Systems

Demo: Implement First Typescript Module

Introduction Packages and NPM

Modules vs. Packages

Demo: Implement First Typescript Package

***Creating and Using Your First Angular Library***

Requirements of a Library

Angular Package Format

Examine the ng-packagr

Understanding the Libs Entryfile

Demo: Create a Library with ng-packagr

Demo: Examine the Demo Application

Demo: Creating a Library with the Angular CLI

Demo: Extracting a Component into the Library

Demo: Extracting a Service into a Library

Demo: Passing Configuration into a Library

Demo: Using Dependency Injection to Switch the Libraries Service

***Building, Packing, and Using the Library in Another Application***

Intoducing Lint and Test Commands

Demo: Linting an Angular Library

Demo: Testing Angular Libraries

Introducing Install, Link, and Pack Commands

Demo: Installing Libraries Locally with NPM Install

Demo: Using Angular Libraries with NPM Link

Demo: Using Angular Libraries with NPM Pack

***Deploying the Library to NPM***

Dealing with Versions and Names of Packages

Demo: Versioning and Preparing the Libraries for Publishing

Demo: Publishing the Libraries to NPM

Demo: Installing and Using the Libraries

**Reusable Angular Services Configuration Management**

***Course Overview***

***Create a Class to Hold Global Settings***

Modules in This Course

Configuration Management Architecture

Create Settings and Service Classes

Create the Product Class

Create the Product Page

Create the Product Detail Component

Update App Component and Run the Sample

Refactor Classes for Efficiency

***Read Settings from a JSON File***

Add Bootstrap Styles to Project

Wrap Bootstrap Around Product Detail Page

Create a Default Settings Page

Create a Settings Page Component Class

Add Routing Module

Add Navigation Menu

Add and Read from JSON File

Add Exception Handling

***Store Settings in Local Storage***

Add saveSettings Method to AppSettingsService Class

Call saveSettings from Settings Page

How to Retrieve Settings

Retrieve Settings from Local Storage

Delete Settings from Local Storage

***Retrieve Settings from SQL Server via a Web API Call***

Create an AppSettings Table

Create ASP.NET Core Web API Project

Add the Entity Framework to the Project

Create Entity Framework Entity Class

Create Database Context Class

Build Get() Controller Method

Convert Camel Case to Pascal Case

Add CORS to the Web API Project

Call Web API from Angular Application

**Reusable Angular Services Logging**

***Course Overview***

***Building a Simple Logging Service***

Goals and Architecture of Logging

Modules in This Course

What's Wrong with console.log()?

Create the LogService

Create Page to Test the LogService

Run and Test the Simple Log Service

***Adding Different Levels of Logging***

Add LogLevel Enum and Properties to LogService

Add shouldLog() Method

Add writeToLog() Method

Test the shouldLog and writeToLog Methods

Add Methods for Each Enumeration

Format Optional Parameters

***Creating a Flexible Log Publishers Service***

Create LogEntry Class and Properties

Write the buildLogString() and formatParms() Methods

Rewrite writeToLog() Method to Use LogEntry Class

Create LogPublisher Base Class

Create LogConsole Class to Log Data to Console

Create LogPublishersService Class

Modify LogService Class to Use the Log Publishers Service

***Building a Local Storage Publisher***

Create the Structure of the LogLocalStorage Class

Create the log() and clear() Methods

Add LogLocalStorage to Publishers Array and Test

Add clear() Method to LogService and Test

Log a Product Object Using Your Log Service

Create getAll() Method in LogLocalStorage Class

Test Retrieving All Values from Local Storage

***Building a Web API Publisher***

Create Web API Project, LogLevel Enum, and LogEntry Class

Build the LogController Class

Enable CORS in the Web API Project

Start Creating the LogWebApi Publisher

Create Constructor and log() Method

Build the clear() and handleErrors() Methods

Add LogWebApi to Log Publishers Service

Test and See the Call to the Web API

***Reading Publishers from a JSON Configuration File***

Build the JSON Configuration File

Create LogPublisherConfig Class

Add Import Statements in Log Publishers Service Class

Add handleErrors() Method

Add getLoggers() Method

Read All Publishers from JSON File

Run and Test Reading the Publishers

**Angular Unit Testing**

***Course Introduction***

The Demo Application

Testing Overview

Mocking

Unit Tests in Angular

Tools of Unit Testing with Angular

Installing and Running the Demo

Writing Your First Unit Test

Running Your Unit Tests

Writing Good Unit Tests

***Isolated Unit Tests***

Testing a Pipe

Testing a Service

Testing a Component

Mocking to Isolate Code

Testing Interactions

***Shallow Integration Tests***

Debugging Techniques with Angular and Karma

The TestBed

Using NO\_ERRORS\_SCHEMA

Testing Rendered HTML

NativeElement vs. DebugElement

More Complex Shallow Integration Tests

Mocking an Injected Service

Mocking Child Components

Dealing with Lists of Elements

***Deep Integration Tests***

Creating a Deep Integration Test

Finding Elements by Directive

Integration Testing of Services

Implementing a Test with Mocked HTTP

***Testing DOM Interaction and Routing Components***

Triggering Events on Elements

Emitting Events from Children

Raising Events on Child Directives

Interacting with Input Boxes

Testing with ActivatedRoute

Dealing with ngModel

Mocking the RouterLink

Testing the RouterLink

***Advanced Topics***

Adding Async Code

Basic Async Testing

Using the fakeAsync Helper Function

Using the async Helper Function

**Building Reusable Angular Components Schematics**

***Getting Started with Angular CLI*** ***Schematics***

Setup

The Power of Schematics

Creating a New Schematic

Adding a Schema

Running a Schematic

***Working with Template Files in a Schematic***

Adding Template Files to a Schematics Project

Templating Filenames

Using Template Expressions Inside Files

Additional Template Features

Conditionally Templating Files

***Building Extra Capabilities into a Schematic***

Getting the Project Workspace File

Updating Existing Files in the Target Project

Chaining Multiple Rules Together

Parsing Files for Correct Insertion Locations

Running NPM Commands in the Target Project

Running External Schematics

***Testing a Schematic***

Getting the Default Test Passing

Testing File Creation

Testing File Contents

Testing a Scheduled Task

***Post Production Tasks***

Testing a Schematic in Another Project Locally

Publishing a Schematic

**Securing Angular Applications with Open ID Connect and Auth2**

***Angular App Security Big Picture***

Security Design Considerations

Client vs. Server Security

Angular App Security Architecture

Authentication and Authorization

Terminology

OpenID Connect And OAuth 2 Protocols

Identity Provider Options

Client Library Options

A Tour through the Demo Application

***Authenticating with OpenID Connect***

Why Use OpenID Connect?

JWT Tokens and OpenID Connect Flows

A Word About the oidc-client Library

Getting Started with the App Code

Getting the Client App Running

Getting the Back End Code Running

Adding oidc-client and an Authorization Service

Adding Login to the App

Adding the Post-login Redirect Page

Creating the User Object and Checking Logged In Status

Inspecting JWT Tokens

Switching to Auth0 and Setting up Tenants, Apps, Users, and APIs

Changing the Client Configuration for Auth0

Addressing Configuration Differences between Auth0 and IdentityServer4

***Authorizing Calls to Your Backend APIs with OAuth2***

OAuth 2 Terminology/Roles

OAuth 2 Grant Types

OAuth 2 Token Types

Demo Overview and Current Status

Adding Authentication to the API

Adding Access Tokens Manually to HTTP Requests

Adding Access Tokens Automatically with an HTTP Interceptor

Adding Access Tokens Automatically with an HTTP Interceptor

Filtering Data in the API Based on Caller Identity

Adding Access Control Checks Based on Permissions

Handling Authorization Errors in the Client App

***Enhancing the User Experience with Client Security Context***

Silent Renewal of Access Tokens

Enabling Silent Renew in Your Angular App

Providing a User Security Context to the Client App

Hide, Show, or Disable UI Based on Security Context

Block Navigation to Unauthorized Routes with Route Guards

Single Sign-On Across Multiple Apps

**Angular security Using JSON Web Tokens**

***Review and Install the Sample Application***

Course Introduction and Goals

Prerequisites and Modules in this Course

Security Goals and Security Architecture

Machine Setup and Getting Started

Create the Sample Project and Workspace

Add Sample Files into Sample Project

Run the Web API and Angular Project

***Authenticating a User***

Security Architecture

Angular User Classes

Build Angular User Classes

Security Service

Build the Security Service

The Login Page

Build the HTML for the Login Page

Build the Login Component

Add Login Route and Login Menu

***Secure UI Elements and Guard Routes***

How to Secure Menus

Secure the Menus

How to Secure Buttons

Secure the Buttons

How to Create an Authorization Route Guard

Create the Authorization Route Guard

How to Redirect to Login Page

Redirect to Login Page

Redirect Back to Requested Route

***Call Web API to Authenticate and Authorize***

Security Architecture

Entity Framework and Authorization Classes

Create Entity Framework Classes

The User Authorization and Security Manager Class

Build the AppUserAuth Class

Build the Security Manager Class

The Security Controller

Build the Security Controller Class

Changes to Security Service to Call the Web API

Call the Web API from Angular

**Use JSON Web Tokens to Secure Web API Methods**

Add Authorize Attribute

Add JWT Packages to Web API Project

How to Configure JSON Web Token System

Add Settings to JSON File and Read the Settings

How to Setup Authentication in the Web API Project

Setup the Authentication in Startup.cs

Create Singleton of Settings

Inject Settings into Controller and Security Manager

How to Create a JWT Token

Create the BuildJwtToken Method

Return Bearer Token to Angular

***Working with Bearer Tokens in Angular***

Pass Bearer Token to Web API

Secure the Entire Product Controller

What Is an HTTP Interceptor?

Create the HTTP Interceptor

Register and Test the HTTP Interceptor

Security Policies

Add Authorization to Web API

Test the Security Policy

***Using Claims-based Authorization***

Changes to Make to the Angular Security Classes

Remove Individual Properties and Add Claims Array

Check Claims in Security Service and Route Guard

Changes to Make to the Web API Classes

Make Changes to Server-side C# Classes

How to Create a Structural Directive

Build and Test the Structural Directive

Secure Menu Items

Handling Multiple Claims per Element

Modify Code to Handle Multiple Claims

Remove Code from Components

**Angular Best Practices**

***Getting Started with Angular***

Setting up Our Environment

Getting Started with the CLI

File Naming

Folder Structure

One Item per File

***General Coding Best Practices***

The Single Responsibility Principle

Symbol Naming Best Practices

Using Immutability

Small Functions

***Angular Module Organization***

Introduction to Angular Module Organization

Creating a Core Module

Creating Shared Modules

Creating Feature Modules

***Angular Components Best Practices***

Prefixing Component Selectors

Separating Component, CSS, and Template Files

Decorating Input and Output Properties

Delegating Complex Logic to Services

Component Member Sequence

Implementing Life Cycle Hook Interfaces

When to (and Not to) Create Components

***Angular Services Best Practices***

Marking Services as Injectable

Using Services for Data Retrieval

Service Injector Best Practices

***Performance Best Practices***

Introduction

Ahead-of-time Compilation and the CLI

Lazy Loading Feature Modules

Monitoring Bundle Sizes

Improving Performance with OnPush Change Detection

Pure and Impure Pipe Performance

**Angular Architecture and Best Practices**

***Introduction***

Prerequisites to Maximize Learning

Key Concepts and Learning Goals

Sample Application and Software Requirements

***Planning the Application Architecture***

Architecture Considerations

Architecture Planning Template

Architecture Planning Template Example - Part 1

Architecture Planning Template Example - Part 2

The Angular Style Guide

Other Considerations

***Organizing Features and Modules***

Organizing Features

Feature Modules

Core and Shared Modules

Core and Shared in Action

Creating a Custom Library

Consuming a Custom Library

Putting All the Modules Together

***Structuring Components***

Container and Presentation Components

Container and Presentation Components in Action

Passing State with Input and Output Properties

Input and Output Properties in Action

Change Detection Strategies

Reference vs. Value Types

Cloning Techniques

Cloning in Action

Cloning with Immutable.js

Component Inheritance

Component Inheritance in Action

***Component Communication***

Component Communication

Understanding RxJS Subjects

RxJS Subjects in Action - Part 1

RxJS Subjects in Action - Part 2

Creating an Event Bus Service

Using an Event Bus Service

Creating an Observable Service

Using an Observable Service

Unsubscribing from Observables

7. State Management

Introduction

The Need for State Management

State Management Options

Angular Services

NgRx

NgRx in Action

ngrx-data

ngrx-data in Action

Observable Store

Observable Store in Action

State Management Review

***Additional Considerations***

Functions vs. Pipes

Functions and Pipes in Action

Using a Memo Decorator

HttpClient and RxJS Operators

Key Security Considerations

HTTP Interceptors

**Building responsive SPA Framework**

***Course Overview***

***Introduction***

Tooling for this Course

The Course Demo Projects and Angular CLI

***User Interface Layout***

Project and Framework Setup

The SPA Framework Module

Creating the Framework Module

Understanding Flexboxes

Developing the Framework Body Class

Create the TitleBar Component

***Configuration, Styling, and Responsiveness***

Creating a Framework Configuration Service

Creating the TopBar Component

Styling the Framework with CSS

Creating the Status Bar

Building a Screen Service for Responsive Features

Making the TitleBar Responsive

Building Responsive Directives

Unsubscribing from Observables

***Menus and Navigation***

A Menu Service

Menu Data

The Menu Component

Showing Menu Items

Routing and Navigation

routerLink and routerLinkActive

Creating a Vertical Menu

Small Screen Menus

Fixing Menu Issues

***Advanced Menu Features***

Switching Menu Orientation

Popup Menu Overview

Menu Data and Routes

Creating the Popup Menu Component

Showing Popup Menus

Cleaning up the Popup Menu

Routing for Popup Menus

Showing Active Routes in Popups

Arrow Indicators for Menu Items

Animated Popup Menus

Testing on Small Devices

***User Controls***

Routing and Child Routes

The Sign-in Component

Building a Registration Form

Dependency Injection and Class-interfaces

Implementing a Class-interface

Dependency Injection Tokens

Submitting the Sign-in Form

Handling Sign-in Errors

Using Route Guards to Authorize Routes

Signing Out

***Forms and CRUD Operations***

View Models and a Data Service

Showing a Country List

Showing a Country Detail Page

Dynamic Forms

The Dynamic Form Template

Dynamic Fields

Building a Read-only Details Form

Building an Update Form

Creating a New Country

Deleting a Country

Viewing Dynamic Forms on Small Devices

***Building Responsive Content Panels and a Dashboard***

Content Projection

Creating a Panel Component

Creating the Country List

Creating the Country Panel

A Responsive Country Panel

Creating a Responsive Panel with Background Image

Creating the Dashboard

Multi-slot Content Projection for Panels

**Angular Materials**

***Introduction***

What We'll Be Building

Course Overview

References and Links

***Material Design***

Key Principles

***Getting Started***

Overview and Tooling

Setting up Angular CLI

Installing Angular Material

Our First Material Components

Creating a Material Master Module

From CSS to SCSS

***Layout Component***

Our Goal

Flexbox Basics

Application Routes

Creating a Demo Module

Setting up the Routes

Flexbox Demo

Scaffolding the App

The Sidenav Component

Styling the Sidenav

Configuring the Sidenav

Adding Responsiveness

Creating a Toggle Button

***Displaying Real Data***

Our Goal

Introducing the Data Service

Feching Data

Navigation Lists

Loading SVG Avatars

Routing to Users

User Cards

Refactoring Routing Issues

Introducing Tabs

***Using Data Tables***

Our Goal

MatTable Features

Generating our Notes Component

Implementing the MatTable

Adding Pagination

Adding Filtering

Adding Header Sorting

***Dialogs and Popups***

Our Goal

Adding a Toolbar Menu

Dialog Basics

Creating our First Dialog

Scaffolding a Form

Customizing the Form

Adding Form Validation

Using the DatePicker

Saving the User

Snackbar Notifications

***Wrapping Up***

Understanding Themes

Creating Custom Themes

Toggling Themes

RTL and LTR Support

Final Words

**Angular Apps Prime Ng**

***Course Introduction***

What Is PrimeNG?

Getting the Most from This Course

Why Non-designers Love PrimeNG

Tour of the Sample App

Getting the Source

***PrimeNG Fundamentals***

What You'll Learn

Installing the Angular CLI

Working with Themes

Cloning the Sample App

Three Step Process

The Prime Grid System

Working with Icon Fonts

Building a Statistics Component

Implementing a Menu

***Charts for Visual Effect***

What You'll Learn

Getting Configured

Your First Pie Chart

Pie Data Models

Working with Dynamic Data

Polar Area Charts

Bar Charts

Line and Radar Charts

Mixing Chart Types

Interactive Charting

Deeper Customization

Realtime Charting

***Forms Unmasked***

What You'll Learn

Reactive Forms

Your First PrimeNG Reactive Form

Reactive Form Markup

The Great Validation Refactor

Styling Submit Buttons

Working with Masked Input

Working with TextAreas

Getting Rich with Text Editing

Fun with Calendars

Radio Buttons

Fieldsets

Choosing Devs with Dropdowns

Choosing Devs with Multiselect

Choosing Devs with a Listbox

Using Custom Item Templates

Using Spinners

Using Sliders

Using Ratings

***DataTables***

What You'll Learn

Your First PrimeNG DataTable

Datatable Markup

Sorting Tables

Filtering Tables

Custom UI Templates for Filtering

Re-order and Resize Columns

Facets: Customising Headers and Footers

CSV Export

Inplace Editing

Row Selection

Context Menu

Scrolling and Pagination

Lazy Loading

***Dialogs, Tabs, and Other Wizardry***

***Intro***

What You'll Learn

Working with Tabs

Creating Dynamic Tabs

Handling Tab Changes

Working with Dialogs

Dialog Headers and Footers

Implementing Wizard Steps

Switching Step Panes

Adding a Schedule

Tree Controls

Working with Maps

DataGrids

Tooltips

Confirmation Dialogs

Toasting Growls

***Advanced Topics***

Introduction

What You'll Learn

Drag and Drop

Using Galleria for Profile Images

Implementing Draggable

Selecting Images

Diving into PrimeNG Source

Unit Testing PrimeNG Components

Where to from Here?

**Constructing UI Angular**

***Getting Started***

***Understanding the Available Building Blocks***

Introduction

Components

@Input Decorator on a Property Setter

Checking for @Output Property Subscribers

Manipulating the UI with Directives

Altering Data Presentation with Pipes

***Querying the UI***

Templates and TemplateRefs

Querying the UI via ViewChild(ren) and ContentChild(ren)

TempateRef Selectors

ViewContainerRefs

HTMLElement as ElementRefs

ElementRef Selector Options - Read

ElementRef Selector Options - Descendants

ElementRef via Dependency Injection

QueryLists for Collections of UI Items

***Manipulating the UI***

View Encapsulation

Shadow Piercing - Bypassing Shadow DOM

Building Custom Pipes - Dynamic Display Manipulation

Directive Selectors - More Than Just Attributes

Host Binding - Manipulating a Parent from a Child

Host Listening - Responding to a Parent From a Child

NativeElement and Renderer2 are NOT Bad

Getting and Using NativeElement Properly

JQuery as an Angular Anti-pattern

***Creating a Dynamic UI***

Using Built-in Content Projection

Wrapping Content in ngContainer

Building Templates with ngTemplate

ngIf-Else with ngTemplate

Rendering Content with ngTemplateOutlet

Dynamic Content with ngTemplateOutletContext

Component Inheritance

Dynamic Component Creation

Overview of the Widget Framework

Creating Dynamic Components Declaratively with ngComponentOutlet

Creating Dynamic Components Programmatically with ComponentFactoryResolver

***Keeping Your DOM Clean***

Component Attribute Selectors

ngIf to Minimize DOM Size

ngSwitch Bloat

Shadow DOM

***Improving Performance of the UI***

A Better UI for Async Loading

Pipes and Performance

A Better ngFor

Instant Caching with Memoizing

A First Look at Angular's Lifecycle Hooks

Working with Angular's Lifecycle Hooks

***Security in the Angular UI***

Understanding the Threat

Default Security

Bypassing the Default Security

***Bringing It All Together: Building Dynamic Forms***

Introduction / What We're Building

Defining the Form in Metadata

Generating the Field UI

Loading and Saving Data

Validating the Field Data

***Wrap up: Use Cases and Final Thoughts***

Error Handling Scenarios

Personalization Scenarios

UI Goodness Scenarios

**Styling Angular Applications**

***Introduction***

Working with the Demos

***How Styles Work in Angular***

***Web Components***

Angular Component Structure

View Encapsulation

View Encapsulation Modes: None & Native

Component Styles

Emulated CSS Selectors

***Scalable, Maintainable CSS/SCSS Architecture in Angular***

Global Styles: A Traditional Approach

Global Styles: A More Modular Approach

Naming Conventions

Predictable Sizing with Relative Units

CSS Selectors & Style Overrides

Component CSS Structure & Organization

Local Mixins & Variables

Styling the Pre-boostrap Loading Screen

***Creating Component Themes***

Component Themes: Layouts & Colors

Adding Classes & Using :host

Using :host-context

Conditional Content Display

Using Custom Properties (CSS Variables)

**Integrating Angular with Nodejs Restful Web Services**

***Course Introduction***

Pre-requisites to Maximize Learning

Learning Goals

Server-side Technologies and Concepts

Client-side Technologies and Concepts

Running the Application

Running the Application with Docker

***Exploring the Node.js and Angular Application***

Exploring the Project Structure

Application Modules

Configuring Node.js Routes

Configuring the ES Module Loader

Angular Modules, Components, and Services

***Retrieving Data Using a GET Action***

Creating a GET Action to Return Multiple Customers

Creating a GET Action to Return a Single Customer

Making GET Requests with an Angular Service

Displaying Customers in a Grid

Displaying a Customer in a Form

Converting to a 'Reactive' Form

***Inserting Data Using a POST Action***

Creating a POST Action to Insert a Customer

Making a POST Request with an Angular Service

Modifying the Customer Form to Support Inserts

Exploring the 'Reactive' Form

***Updating Data Using a PUT Action***

Creating a PUT Action to Update a Customer

Making a PUT Request with an Angular Service

Modifying the Customer Form to Support Updates

Exploring the 'Reactive' Form

***Deleting Data Using a DELETE Action***

Creating a DELETE Action to DELETE a Customer

Making a DELETE Request with an Angular Service

Modifying the Customer Form to Support Deletes

Exploring the 'Reactive' Form

***Data Paging, HTTP Headers, and CSRF***

Adding a Paging Header to a RESTful Service Response

Accessing Headers and Data in an Angular Service

Adding Paging Support to a Component

Adding a Paging Component

CSRF Overview

Adding CSRF Functionality with csurf

Using a csurf Token in an Angular Service

**Designing Restful Web APIs**

***What Is REST?***

Course Overview

History of Distributed Computed

HTTP in a Nutshell

HTTP in Action

What is REST

An Example of a Well Designed API

What We've Learned

***Designing a RESTful API***

Designing for REST

Demo: Using URIs

Design Verbs

Demo: Using Verbs

Idempotency in Action

Understanding Idempotency

Designing Results

Demo: Designing Your Results

Formatting Results

Demo: Formatting Results

Hypermedia

What We've Learned

***Handling More Complex Scenarios in Your API***

Designing Associations

Demo: Associations

Designing Paging

Demo: Paging

Error Handling

Demo: Error Handling

Designing Caching

Demo: Caching with ETags

Functional APIs

Demo: Functional APIs

Asynchronous APIs

What We've Learned

***Versioning Your API***

Should You Version Your APIs

Designing Versioning

Demo: Versioning Strategies

What We've Learned

***Locking Down Your API***

APIs and Security

Cross Domain Security

Authentication and Authorization

Authentication Types

Understanding OAuth

What We've Learned

**RxJS: The Big Picture**

***Course Overview***

***Introduction to Reactive Programming and RxJS***

Introduction

What Is RxJS?

What Problems Does RxJS Solve?

The Observer Pattern

***Benefits of RxJS***

Introduction

A Better Asynchronous API

Programming Language Choice

Operators

Maintenance and Enhancements

***Relationship to Other Libraries and Frameworks***

Libraries Versus Frameworks

RxJS with Angular

RxJS with React and Vue

Adopting RxJS Gradually

***RxJS Building Blocks***

Introduction

Observables and Observers

Operators

Subjects

Schedulers

Demo: Creating Observables and Observers

Demo: Observing Events and Applying Operators

Next Steps

**RxJS: Getting Started**

***Course Overview***

***Reactive Programming Basics***

What Is RxJS?

Imagining Data as a Stream

RxJS Building Blocks

Demo Project Overview

Compatibility Packages for RxJS 5

***Creating Observables***

Observables and Observers

Instantiating a New Observable with the Constructor

Creating Observables from Existing Data

Creating Observables to Handle Events

Making AJAX Requests with RxJS

***Subscribing to Observables with Observers***

Introduction

Understanding Observers

Creating and Using Observers

Executing Observables

Multiple Observers Executing a Single Observable

Managing Subscriptions

Cancelling Observable Execution with a Subscription

***Using Operators***

Introduction

Applying Operators

Categories of Operators

Reading a Marble Diagram

Importing and Using Common Operators

Handling Errors

Controlling the Number of Values Produced

***Creating Your Own Operators***

Why Create Your Own Operators?

Structure of an Operator

Creating New Operators with the Observable Constructor

Creating New Operators from Existing Operators

***Using Subjects and Multicasted Observables***

Introduction

What Are Subjects?

Producing Values with Subjects

Cold vs. Hot Observables

Using a Subject to Convert an Observable from Cold to Hot

Multicasting Operators

***Using Multicast Operators Instead of Subjects***

Specialized Subjects

Controlling Multicasted Output with Specialized Operators

***Controlling Execution with Schedulers***

What Are Schedulers?

RxJS Schedulers

Understanding Schedulers and the Event Loop

Using Schedulers with Observable Creation Functions

Applying a Scheduler with the observeOn Operator

***Testing Your RxJS Code***

Introduction

Using the TestScheduler

Observable and Subscription Marble Syntax

Structuring Unit Tests

Testing Observables and Subscriptions with Marble Diagrams

**Getting Started with Reactive Programming Using RxJS**

***Course Overview***

***Observers and Observables***

An Introduction

Reactive Programming

Installing RxJS with npm

Configuration and Setup

Creating Your First Observable and Observer

An Easier Observer

Using Observable.create

Going async with setTimeout

Using RxJS Operators

Importing Just What We Need

***Working with Observables***

Introduction

Processing Mouse Events

Sending Requests with XmlHttpRequest

Using flatMap to Process Inner Observables

Implementing Retry Logic with retry and retryWhen

Set up to Use the New Fetch Standard

Using fetch and Promises

***Working with Observable Data***

Introduction

Finding the Operator You Need

Dealing with Errors and Exceptions

Revisting the retryWhen Strategy

Unsubscribing for Cleanup

**Unit Testing RxJS with Marble Diagrams**

***Course Overview***

***Getting Started***

Introduction

Prerequisites

Course Repository

Why Learn Marble Testing?

Sample Application

***Learning Jasmine Marble Basics***

Introduction

Marble Diagrams: Introduction

Marble Diagrams: Examples

Jasmine Marbles: Introduction

Jasmine Marbles: Why This Library?

Jasmine Marbles: Learning Its Methods

Set up Your Test Project

Selecting Your Editor

VS Code Editor

Installing Node

Source Code Repository

Demo: Set up Your Test Project

Marble Syntax: Time Frame

Marble Syntax: Symbols

Marble Syntax: Empty and Never RxJS Observables

Demo: Writing Your First Marble Test

***Unit Testing with Hot and Cold Observables***

Introduction

Hot and Cold Observables Building Blocks of RxJS

Hot Observable Marble Diagrams

Demo: Hot Observable Marble Diagrams

Cold Observable Marble Diagrams

Demo: Cold Observable Marble Diagrams

Introduction: Marble Testing Hot and Cold Observables

Understanding Frames in Jasmine Marbles

Marble Testing for Cold Observable

Demo: Marble Testing Cold Observable: Part 1

Demo: Marble Testing Cold Observable: Part 2

Marble Testing Hot Observable

Understanding Subscription Model of Hot Observable

***Unit Testing by Mocking*** Observable Values and Testing RxJS Operators

Introduction

Demo: Marble Testing Hot Observable: Part 1

Demo: Marble Testing Hot Observable: Part 2

Mocking Observable Values

Demo: Mocking Observable String Values

Demo: Mocking Observable Number Values

Demo: Mocking Observable Array Values

Demo: Mocking Observable Object Values

Marble Testing RxJS Operators

Understanding Marble Diagrams of Concat Operator

Demo: Marble Testing with Concat Operator

Demo: Marble Testing Subscription Behavior of Concat Operator

Demo: Marble Testing Real-world Concat Operator Example

Understanding Marble Diagrams of Zip Operator

Demo: Marble Testing with Zip Operator

***Testing Business Code and Handling Errors & Race Conditions***

Introduction

Test Scheduler in Jasmine Marbles

Demo: Marble Testing Component by Mocking Service

Introduction Race Condition

Demo: Finding Race Condition with Marble Testing

Demo: Fixing Race Condition and Fixing Unit Test

Demo: Debounce Time with Marble Testing

Marble Syntax for Errors

Demo: Marble Testing for Error Handling

Module Summary

Course Summary

What to Do Next?

**Learning RxJS Operators by Example Playbook**

***Course Overview***

***RxJS 101***

***Buffering***

Introduction

buffer

bufferCount

bufferTime

bufferToggle

bufferWhen

window

windowCount

windowTime

windowToggle

windowWhen

***Error Handling***

catchError

throwIfEmpty

onErrorResumeNext

retry

retryWhen

timeout

timeoutWith

***Filtering to Multiple Results***

skip

skipLast

skipUntil

skipWhile

take

takeLast

takeUntil

takeWhile

distinct

distinctUntilChanged

distinctUntilKeyChanged

filter

sample

audit

throttle

***Filtering to One Result***

first

last

min

max

elementAt

find

findIndex

single

***Grouping Observables***

combineAll

concatAll

exhaust

mergeAll

withLatestFrom

***Grouping Values***

groupBy

pairwise

partition

switchAll

toArray

zipAll

***Observable Transformation***

repeat

repeatWhen

ignoreElements

finalize

***Time, Duration, & Scheduled***

auditTime

sampleTime

observeOn

subscribeOn

debounce

debounceTime

delay

delayWhen

throttleTime

timeInterval

timestamp

***Value Transformation***

concatMap

concatMapTo

defaultIfEmpty

endWith

startWith

exhaustMap

expand

map

mapTo

scan

mergeScan

pluck

reduce

switchMap / flatMap

mergeMapTo

switchMapTo

materialize

dematerialize

***Multicasting***

multicast

share

shareReplay

publish

publishBehavior

publishLast

publishReplay

***Utilities***

count

every

isEmpty

sequenceEqual

tap