multiverse

Contents

Name:
Date:

Angular

> We recommend doing this plan in combination with the TypeScript Learning Plan.

Angular is a front-end web framework for building dynamic web applications using TypeScript, a superset of JavaScript. It provides a set of powerful tools and features for building complex, scalable, and maintainable web applications. Angular was initially developed by Google in 2010 and is currently maintained by a team of developers and contributors.

Angular is particularly well-suited for building single-page applications (SPAs) and dynamic web applications that require a lot of user interaction and real-time data updates. It's also useful for creating large, enterprise-level applications with complex business logic and data processing requirements. Many popular websites and applications, such as Google Cloud Platform, Netflix, and PayPal, use Angular for their front-end development.

The first version of Angular, called AngularJS, was released in 2010. It gained popularity quickly and became one of the most widely-used front-end frameworks for building web applications. However, AngularJS had some limitations, particularly in terms of performance and scalability, so a new version of Angular, called Angular 2, was released in 2016. Angular 2 introduced significant changes and improvements, including a modular architecture, improved performance, and support for mobile development. Since then, Angular has continued to evolve and improve with each new version.

Angular is a comprehensive framework that includes many built-in features and tools, such as a powerful templating system, reactive programming support, and robust testing tools. It uses TypeScript, a statically-typed language that provides additional safety and productivity features for developers. Angular also emphasizes modularity and code organization, making it easier to maintain and scale large codebases. Additionally, Angular has a large and active community of developers and contributors, which helps to ensure that the framework is continuously improving and evolving.

Learning Plan Tasks

1. Introduction to TypeScript and Angular
2. Components Templates and Directives
3. Forms and HTTP
4. Advanced Angular Concepts
5. Advanced Topics and Best Practices
6. Build Something Using Angular

#checkoutTheDocs

- **Angular**: Documentation
- **Angular**: Getting Started
- **Angular**: Tutorial - Your First Angular App
- **Angular University** - Numerous Angular courses and tutorials
- **Free Code Camp**: Angular Course

- **TypeScript**: The TypeScript Handbook

1. Introduction to TypeScript and Angular

This section covers the following topics:

- Introduction to TypeScript and how it extends JavaScript
- Overview of Angular framework and its benefits
- Setting up a development environment for TypeScript and Angular
- Creating a basic Angular application

Tasks

1. Introduction to TypeScript
    - Introduction to the TypeScript language and its benefits
    - Overview of TypeScript features, such as static typing, interfaces, and classes
    - Install TypeScript and set up a development environment
    - Write a basic TypeScript program and compile it using the TypeScript compiler.
2. TypeScript Fundamentals
    - Understand TypeScript types, including `number`, `string`, `boolean`, `any`, and `void`
    - Declaring variables and using type annotations
    - Using interfaces to define object shapes and class contracts
    - Understand functions in TypeScript, including function types, parameters, and return types
    - Write a TypeScript program that uses types, variables, interfaces, and functions.
3. Introduction to Angular
    - Overview of the Angular framework and its benefits
    - Understand Angular architecture, including modules, components, and services
    - Install Angular and setting up a development environment
    - Create an Angular project using the Angular CLI and explore the project structure.
4. Angular Components
    - Understand Angular components and their role in the application
    - Create a basic Angular component
    - Using templates to display data in a component
    - Understand data binding and event binding in Angular
    - Create an Angular component and use data binding to display data in the template.
5. Angular Templates
    - Creating templates using Angular's built-in syntax
    - Understand template expressions, statements, and directives

- Using structural directives like `ngIf`and `ngFor`
- Create an Angular component with a template that uses `ngIf`and `ngFor`.

6. Angular Directives
    - Understand directives in Angular
    - Creating custom directives
    - Using built-in directives like `ngStyle`and `ngClass`
    - Create a custom directive and use built-in directives to manipulate the appearance of an Angular template.

2. Components, Templates, and Directives

This section covers the following topics:

- Understand components and how they work in Angular
- Creating and using templates to display data
- Using directives to manipulate the DOM and add logic to templates
- Working with built-in directives like `ngIf`, `ngFor`, and `ngSwitch`

Tasks

1. Angular Components
    - Review of Angular components and how they work in the application
    - Understand component lifecycle hooks
    - Creating nested components in Angular
    - Create a nested Angular component and use lifecycle hooks.
2. Angular Templates
    - Understand template syntax in Angular
    - Using interpolation and property binding to display data
    - Using event binding to handle user interactions
    - Create an Angular template with interpolation and event binding.
3. Structural Directives
    - Understand structural directives like `ngIf` and ngFor
    - Using `ngSwitch` to conditionally display content
    - Use `ngIf`, `ngFor`, and `ngSwitch` to conditionally display content in an Angular template.
4. Attribute Directives
    - Understand attribute directives in Angular
    - Creating custom attribute directives
    - Create a custom attribute directive in Angular.
5. Services and Dependency Injection
    - Understand services and how they work in Angular
    - Using dependency injection to provide services to components
    - Creating a basic service in Angular
    - Create a service in Angular and use dependency injection to provide it to a component.

6. Routing in Angular
    - Understand routing in Angular
    - Creating routes in an Angular application
    - Using the Angular router to navigate between routes
    - Create routes in an Angular application and use the Angular router to navigate between them.

3. Forms and HTTP

This section covers the following topics:

- Creating forms in Angular and handling user input
- Validating user input with built-in validators
- Making HTTP requests to external APIs and displaying the results in your application

Tasks

1. Reactive Forms
    - Understand reactive forms in Angular
    - Creating a basic reactive form in Angular
    - Using validators to validate form input
    - Create a reactive form in Angular and add validators to it.
2. Template-driven Forms
    - Understand template-driven forms in Angular
    - Creating a basic template-driven form in Angular
    - Using built-in directives to validate form input
    - Create a template-driven form in Angular and use built-in directives to validate form input.
3. Observables and RxJS
    - Understand observables and how they work in Angular
    - Using `RxJS` operators to manipulate observables
    - Create an observable in Angular and use `RxJS` operators to manipulate it.
4. HTTP Requests in Angular
    - Understand how to make HTTP requests in Angular
    - Using Angular's built-in `HttpClient` module to make requests
    - Handling errors and using `RxJS` operators to manipulate responses
    - Make HTTP requests in an Angular application using the `HttpClient` module.
5. Authentication and Authorization
    - Understand authentication and authorization in Angular
    - Using JSON Web Tokens (JWT) for authentication and authorization
    - Implementing a basic authentication and authorization system in an Angular application

- Implement a basic authentication and authorization system in an Angular application using JWT.

6. Deployment and Best Practices
    - Understand deployment options for Angular applications
    - Best practices for building and deploying Angular applications
    - Deploy an Angular application using a popular hosting service, and review best practices for building and deploying Angular applications.

4. Advanced Angular Concepts

This section covers the following topics:

- Managing application state and lazy loading
- Angular Material components
- Animation
- Internationalisation
- Testing applications

Tasks

1. NgRx
    - Understand the `NgRx` library for managing application state in Angular
    - Creating a basic store in `NgRx`
    - Using `NgRx` effects to manage side effects in the application
    - Use `NgRx` to manage state in an Angular application.
2. Lazy Loading
    - Understand lazy loading in Angular and how it can improve application performance
    - Configuring lazy loading in an Angular application
    - Implement lazy loading in an Angular application.
3. Angular Material
    - Understand the Angular Material library for UI components in Angular
    - Using Angular Material components in an Angular application
    - Use Angular Material components to create a UI in an Angular application.
4. Animations in Angular
    - Understand how to add animations to an Angular application
    - Using Angular's built-in animation library to create animations
    - Add animations to an Angular application using Angular's built-in animation library.
5. Internationalisation (i18n)
    - Understand internationalisation (i18n) in Angular and how to support multiple languages in an application
    - Using Angular's i18n tools to create translations for an application
    - Add support for multiple languages to an Angular application using Angular's i18n tools.

6. Testing in Angular
    - Understand testing in Angular and the different types of tests available
    - Write unit tests for Angular components, services, and pipes
    - Use end-to-end testing with Protractor
    - Write unit tests and end-to-end tests for an Angular application.

5. Advanced Topics and Best Practices

This section covers the following topics:

- Overview of advanced topics like observables, pipes, and testing
- Best practices for building scalable and maintainable Angular applications
- Tips and tricks for debugging and troubleshooting your application

Tasks

1. Server-side Rendering (SSR)
    - Understand server-side rendering (SSR) in Angular and how it can improve performance and SEO
    - Setting up an Angular application for SSR
    - Implement server-side rendering in an Angular application.
2. Progressive Web Apps (PWA)
    - Understand progressive web apps (PWA) and how to build them with Angular
    - Configuring an Angular application to be a PWA
    - Create a progressive web app using Angular.
3. Angular Performance Optimization
    - Understand performance optimization in Angular
    - Using Angular's built-in tools to improve performance
    - Optimize the performance of an Angular application using Angular's built-in tools.
4. Debugging and Troubleshooting
    - Understand how to debug and troubleshoot Angular applications
    - Using the Angular CLI to generate debugging information
    - Debug and troubleshoot an Angular application.
5. Advanced Techniques
    - Understand advanced techniques for working with Angular
    - Using dynamic component loading to improve performance and user experience
    - Use dynamic component loading to improve the performance and user experience of an Angular application.
6. Wrap-Up and Next Steps
    - Review of key concepts learned during the week
    - Best practices for continuing to learn and work with Angular

Build Something with Angular!

You have already built some very powerful initial applications. After completing your Angular Learning Plan, try to build out at least one of the applications below (or something else that you want to build!):

- **To-do list application**: Build a to-do list application that allows users to add, edit, and delete tasks. You can use Angular's built-in forms and validation features, as well as TypeScript interfaces and classes to define your data models.
- **Weather app**: Build a weather app that displays current weather data for a given location. You can use a weather API to fetch the data, and use Angular's routing and HTTP features to create a dynamic and responsive UI.
- **Shopping cart**: Build a shopping cart application that allows users to add items to their cart, view their cart, and check out. You can use Angular's component and service features to create a modular and reusable application, and use TypeScript classes and interfaces to define your data models.
- **Recipe app**: Build a recipe app that allows users to browse and search for recipes, and save their favourite recipes for later. You can use Angular's forms and validation features to create a dynamic and user-friendly search interface, and use TypeScript classes and interfaces to define your data models.
- **Chat Application**: Build a real-time chat application that allows users to chat with each other in real-time. You can use Angular's component and service features to create a modular and reusable application, and use TypeScript classes and interfaces to define your data models.