Class 11 - Getting Started

Class 11 Course Content

Lesson Outline

Today we will learn:

1. What Angular is.
2. What a Single-Page-Application is.
3. How to create your first Angular app using the Angular CLI.
4. How to edit our first app.
5. How to add Bootstrap to our app.
6. How this course is structured.
7. How to get the most out of this course.
8. A Basic Introduction to Typescript.

Lesson Notes

- **Angular**: *Angular* is a Javascript framework that allows you to create reactive Single-Page-Applications (SPA's).
- **SPA:** A *Single-Page-Application* is an app composed of one file the user receives. The server will send the client ONE Html File that can dynamically change using javascript. Creating a SPA allows a very interactive experience and much faster loading times. The website will look and feel like a native mobile app.
- **Angular Versions:** There are two major *Angular versions*. Angular 1, typically called AngularJS, and Angular 2+. Angular 2 and beyond is a complete rewrite entirely different from AngularJS. We will be using Angular 12 in this course, which will be very similar to all versions from Angular 2, 3, 4 up to 12. Every six months, we get a new *version* (not rewrite) of Angular.
- **CLI:** The *Command Line Interface* is a terminal that accepts text input to execute operating system functions.
- **IDE:** The *Integrated Development Environment* is a tool developers use to edit source code. Many IDE's have advanced features that enhance code quality and developer productivity. We will use Microsoft's Free IDE, Visual Studio Code in this course.

Project Steps

STEP 1: What is Angular/SPA Demo

BookIt Application:

- Click through all the links and explore the project.

- Note how the refresh bar never changes. Our application is a SPA!
- Right-click the page and click View Page Source.

STEP 2: Installing Angular

Terminal:

- *Note*: to avoid specific issues, one may need to run the Terminal as an administrator.
- Ensure everyone has npm version 6 installed on their machine by running `sudo npm install -g npm@6`.
- Check your node version by running `npm -v`.

You can ignore the errors as long as you get a successful install.

- Install the Angular CLI by running `sudo npm install -g @angular/cli`.

STEP 3: Creating Your First Angular App Using the Angular CLI

New Terminal:

- Navigate to the folder you want to place your project in by using `cd myFolder`.
- Create a new project by typing `ng new my-first-app --no-strict`.

No whitespace in your app name or the word "test".

- type "n" to skip using Angular Routing, and select CSS by clicking "enter".

Jasmine Error? Change "jasmine-core" to version 3.7.1 and "karma-jasmine-html-reporter" to 1.6.0 and save it. Then go back to Terminal and go to your project and run npm install --force. Now it works, and you can run ng serve.

- Once all the dependencies are installed, run `cd my-first-app` to move into the correct folder, and then `ng serve` to start a development server for your project.
- Go to the browser and navigate to "http://localhost:4200" to view the Starting Angular Template.

STEP 4: Editing Our Application

IDE - Opened to the folder previously created:

- Open your application folder in VSC
- View the `package.json` file, which contains all the packages and dependencies your application needs to run.
- Navigate to the src folder and the app folder. This will be where you will spend the majority of your time editing.

Ensure your development ng serve process is still running. Keep this running until you are done for the day. It will watch for changes and update your private server on save.

- Go to the "app.component.html" and update one of the tags. Click save. View your browser.
- Almost every feature or component of your app will have an HTML file, a CSS file, and a Typescript file. The HTML will be what the user sees. The CSS is how to style that markup. Furthermore, the Typescript is where to put the logic and dynamic variables for that component.
- View the page source and show the `<app-root></app-root>` in the body and how that is because of our "selector" in the typescript file.
- Change the title variable to "name" and add your name.
- Delete everything in the HTML besides a paragraph outputting your name variable.
- Above your name variable, create an input like so: `<input type="text" [(ngModel)]="name" />`. The "ngModel" directive tells Angular to listen to anything entered in that input, store it in the name variable, listen to the name variable, and update based upon that. It is a two-way connection. If you change the input, the name will change. If you change the name, the input will change.
- View the error in your browser console.
- Navigate to the app.module.ts file and `import { FormsModule } from "@angular/forms"` and add to the imports array.
- Test your application.

STEP 5: Adding Bootstrap to Our Application

Terminal:

- Close your development process using `CTRL + C`, and type: `npm install --save bootstrap@4`.

angular.json file:

- To use this, we need to let Angular know we installed it by going to the angular.json file.
- Navigate to the styles array and add a new import above the "src/styles.css" as "node_modules/bootstrap/dist/css/bootstrap.min.css",
- Save the file and re-run `ng serve` in the correct directory.

app.component.html file:

- To check if this worked, go to the app.component.html file, add: `<div class="container"> </div>` wrapping all the content, and check your browser to make sure it worked!

Angular Course Information

How This Section is Structured

Getting Started: Build and Edit our First Application.

The Basics: How Angular works together. File Structure.

Components & Databinding: Two key features of Angular. The building blocks and the actions/reactions to user events.

Directives: Learn Angulars Directives and Build Your Own.

Services & Dependency Injection: Core feature that allows many different components to speak to each other. Learn the basics of Application State Management.

Routing: How to navigate between "pages" of your application by managing the changing of URL's.

Observables: A Concept allowing you to work with Asynchronous Code.

Forms: Learn how to handle user input.

Pipes: Transform the output displayed at runtime.

HTTP: How to connect to a server/database using HTTP Requests.

Authentication: How to create, store, and update users of your application.

Optimizations & NgModules: How to improve performance and manage the modules in our code.

Deployment: How to put your website on an actual server with a url that anyone can visit.

Animations & Testing: Bonus features that will take your application to another level.

Getting the Most Out of This Section

- Watch all of the videos.
- Do all of the Assignments.
- Do the course Project.
- Write down your questions.
- When stuck, docs/google => debugger/walk-through-code => peers => coaches.

Additional Notes

Class Exercise

1. Using the Terminal, create a new project titled "exercise_class-one".
2. In the "app.component.html" file, add an h1 element and a paragraph element. Add content in both elements.

3. Create a variable in the "app.component.ts" file `name: sting = "William". (Use your own name).
4. Display this variable inside the h1 element we created earlier.
5. Push your angular project to GitHub using GitHub Desktop.

Bonus: Install Bootstrap and then include styling to your HTML elements.

Bonus: add an input element and include "ngModel" to dynamically change the "name" variable.

Note: You will have to import the FormsModule to the app.module from @angular/core and include it within the imports array.

Resources

- CLI Overview and Command Reference
- Our Group Github Repository Notes