COM661 Full Stack Strategies and Development

FE20. Front-end Documentation

Aims

- To summarise the role of software documentation
- To introduce the Compodoc library for generating documentation for Angular applications
- To install and prepare Compodoc
- To generate the default Compodoc documentation
- To demonstrate the specification of bespoke documentaiton using JSDoc comments
- To present JSDoc tags for providing context to documentation
- To introduce the Compodoc Documentation Coverage facility

Table of Contents

20.1 Software Documentation and Compodoc

Software documentation is a collection of written and illustrated information that explains how to use or how a piece of software works. It can be embedded within the software or provided as a separate document. Software documentation can take many forms and can include

- User documentation that provides instructions for the users of the software
- Technical documentation that provides detailed implementation information
- Process documentation that describes the steps and procedures used to develop, test and maintain the software
- Software Design Documents that outline the application's main components and interfaces
- API documentation that specifies how to invoke endpoints and how to interpret the response
- Release notes that describe the version history, releases of features, and bugs that have been found and fixed
- System documentation that describes the main requirements to use the software.

In recent years, many electronic tools and frameworks have emerged for specifying, structuring, and publishing documentation, often integrated into the environment in which the actual source code is written. One such example is **Compodoc**, which is a documentation tool for Angular applications (as well as supporting other JavaScript-based frameworks) that uses a combination of code analysis and structured comments to generate comprehensive documentation that can either be published independently or hosted alongside the application. In this practical, we will introduce the **Compodoc** tool and investigate the documentation that it produces.

20.1.1 Installation

Compodoc is an Open-Source project led by Vincent Ogloblinsky and contributed to by a team of volunteer developers on Github. It can easily be installed into an Angular application by using **npm** to install the library into the application using the following command.

```
C:/BizFE> npm install @compodoc/compodoc --save
```

To run **Compodoc**, we need to create a file within the top-level project folder called ***ts.config.doc.json*** and with the following contents.

File: BizFE/tsconfig.doc.json
```
{
    "include": ["src/**/*.ts"],
    "exclude": ["src/test.ts", "src/**/*.spec.ts"]
}
```

The `include` and `exclude` attributes specify which parts of the application should be documented and which parts should be ignored. Typically, an Angular development, we specify that all TypeScript files within the **src** folder should be documented, with the exception of test files.

Next, we add an execution command to the `scripts` object within the application's ***package.json*** file. This can take a variety of forms and there are many command line options that can be included, but those that we choose to include here are described as follows

`-p tscomfig.doc.json`	to provide a pointer to the configuration file
`-s`	to serve the documentation from a web server that will launch at localhost:8080
`-w`	to watch the source files and automatically re-build the documentation when one of the source files changes.

The code that should be added to ***package.json*** is presented in the code box below.

File: BizFE/package.json

```
{
  "name": "biz-fe",
  "version": "0.0.0",
  "scripts": {
    ...
    "compodoc": "npx compodoc -p tsconfig.doc.json -s -w"
  }

...
```

Do it now!	Install the **compodoc** library into your Angular application using npm and provide the code presented in the code boxes above for ***tsconfig.doc.json*** and ***package.json***.

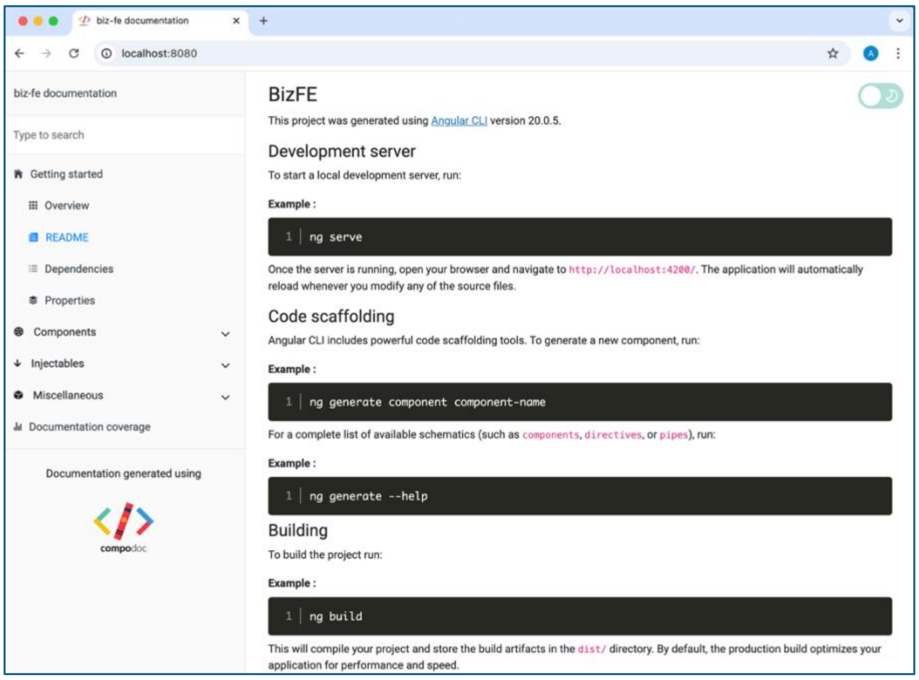
20.1.2 Creating the Documentation

With **Compodoc** installed in our application, we can run it to build the default documentation. Providing the new `script` entry in ***angular.json*** above means that we can run **Compodoc** from a terminal prompt by the command

```
C:/BizFE> npm run compodoc
```

This causes **Compodoc** to scan and analyse the application code and build the default documentation. By default, the documentation is created within a top-level documentation folder within our application, but specifying the `-s` flag in the `script` entry also launches a local web server that displays the documentation at http://localhost:8080.

The default application documentation is shown in Figure 20.1 below. If you navigate through the menu options presented, you can see that all Components, Services and other elements are recognised and described, and that each class, method, property, parameter and other code elements are presented. **Compodoc** is able to create all of this by simply reading the code that we provide, but the added value in documentation is where we provide our own insights about why decisions have been taken, the purpose of each code element, and other contextual information.



BizFE

This project was generated using Angular CLI version 20.0.5.

Development server

To start a local development server, run:

Example :

```
ng serve
```

Once the server is running, open your browser and navigate to `http://localhost:4200/`. The application will automatically reload whenever you modify any of the source files.

Code scaffolding

Angular CLI includes powerful code scaffolding tools. To generate a new component, run:

Example :

```
ng generate component component-name
```

For a complete list of available schematics (such as `components`, `directives`, or `pipes`), run:

Example :

```
ng generate --help
```

Building

To build the project run:

Example :

```
ng build
```

This will compile your project and store the build artifacts in the `dist/` directory. By default, the production build optimizes your application for performance and speed.

Figure 20.1 Compodoc Server Running

Do it now!	Run **Compodoc** using the command **`npm run compodoc`** at a terminal window. (Note, you can open a second terminal on the folder so that you can also run **`ng serve`** on the application at the same time. Navigate through the information provided and see how all of the various code elements are represented.

20.2 Specifying Documentation

The default documentation generated by **Compodoc** produces full coverage of the application code and makes all code elements available within an easy-to-navigate framework. However, as well as code structure analysis to present components, services, methods, parameters, properties and the various other code elements, **Compodoc** can also capture developer intent and context by using the structure and location of comments to add additional value to the documentation.

20.2.1 JSDoc Comments

JSDoc is a mark-up language used to annotate JavaScript source files. Using comments that contain JSDoc, developers can add descriptive detail about the code, providing general information as well as specifying the interface to a code element in terms of the parameters required and the return value generated. In **Compodoc**, comments in JSDoc format are automatically scanned and are included in the documentation produced.

JSDoc comments are provided in the following form

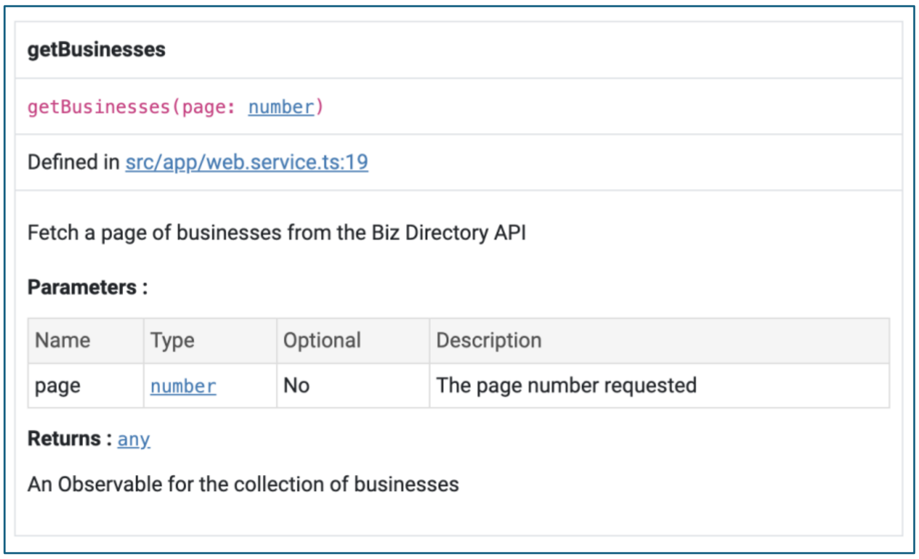
```
/**
*
* JSDoc Comments must begin with /** and end with */
* Each line of the comment block must begin with *
*
*/
```

In addition, JSDoc provides a range of tags that are used to identify different code elements. For example, in the JSDoc comment added to the Web Service method **`getBusinesses()`**, shown in the code box below, we use the **`@param`** tag to provide a description of the page parameter to the method and the **`@returns`** tag to provide a description of the value returned from the method.

File: services/web-service.ts

```
...

    /**
     * Fetch a page of businesses from the Biz Directory API
     * @param page The page number requested
     * @returns An Observable for the collection of businesses
     */
    getBusinesses(page: number) {
        return this.http.get<any>(
            'http://localhost:4999/api/v1.0/businesses?pn=' +
            page + '&ps=' + this.pageSize);
    }

...
```

The effect of the JSDoc comment can be seen by saving **`web-service.ts`** and observing the documentation re-built to reflect the new addition. Now, the comment above the **`getBusinesses()`** method is added to the documentation for the method to incorporate the additional contextual information. This is illustrated in Figure 20.2, below, which shows the revised **Compodoc** documentation for the Web Service method **`getBusinesses()`**.



getBusinesses

`getBusinesses(page: number)`

Defined in src/app/web.service.ts:19

Fetch a page of businesses from the Biz Directory API

Parameters :

Name	Type	Optional	Description
page	number	No	The page number requested

Returns : any

An Observable for the collection of businesses

Figure 20.2 Documentation Generated from JSDoc Comment

Do it now!	Add the JSDoc comment shown above to the **`getBusinesses()`** method in the file ***web-service.ts***. Verify that the **Compodoc** documentation has been automatically updated with the new information.

There are only a few formatting rules to follow when providing JSDoc comments

- Each line must contain an asterisk, and the asterisks must be vertically aligned
- Each asterisk must be followed by a space or newline (except the first and the last)
- The only characters before the asterisk on each line must be whitespace characters
- If a newline is required in the rendered documentation, then a blank line (except for the compulsory asterisk) must be included in the comment. Comment text on successive lines will be rendered on the same line of the documentation (word-wrap permitting).

In the example above, we also introduce the tags **`@param`** and **`@returns`** to describe the method parameters and return value. These are example of a range of contextual JSDoc tags that can be used in the documentation and that are understood by **Compodoc**, including those shown below. A full set can be seen in the JSDoc documentation linked from Section 20.3 in this document.

`@author`	The name of the developer
`@constructor`	Marks a method as a constructor
`@param`	Documents a method parameter. A data type indicator can also be added between **`{}`** braces such as **`@param { number } page`** for the **`page`** parameter passed to the **`WebService getBusinesses()`** method.
`@private`	Denotes that an element is private to the class in which it is defined
`@returns`	Documents a return value
`@since`	Allows the history of an element to be describes, e.g. **`@since v2.3.1`**
`@todo`	Documents a feature that is missing or under development

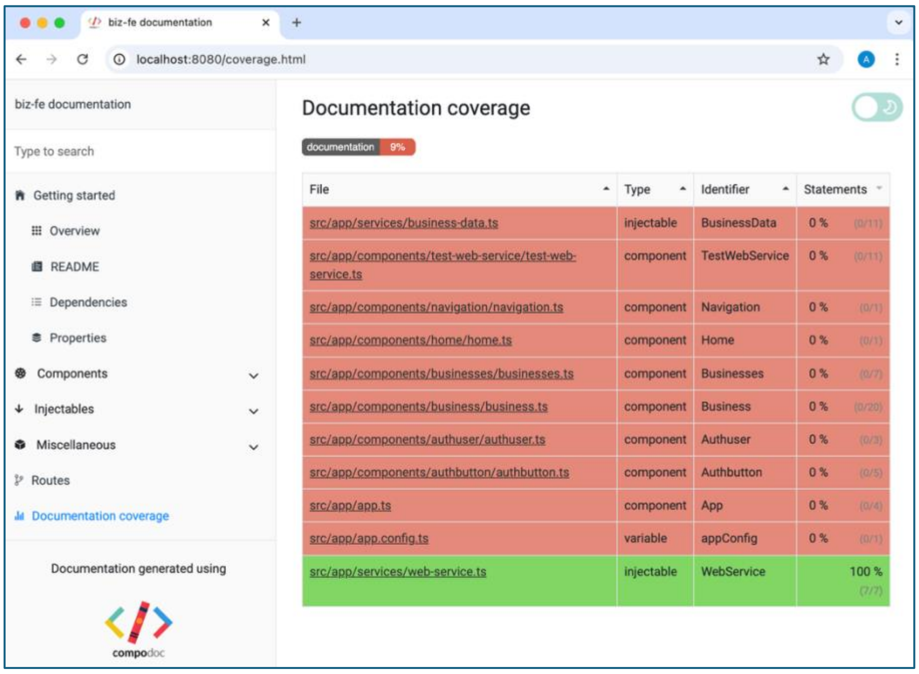
Do it now!	Extend the documentation provided for the **`WebService`**, by adding JSDoc comments for all methods and properties. The code box below provides an extract of this for you.

File: services/web-service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

/**
 * The Web Service provides access to the endpoints of the
 * Biz Directory API.
 */
@Injectable({
  providedIn: 'root'
})
export class WebService {

  /**
   * The default page size to be returned
   */
  pageSize: number = 3;

  /**
   * The constructor for the Web Service
   * @param http Injecting the HttpClient to the WebService
   * class
   */
  constructor(private http: HttpClient) {}

    ...
```

20.2.2 Documentation Coverage

One of the most useful features of **Compodoc** is the Documentation Coverage report that identifies those elements of the documentation for which good coverage has been achieved and those for which additional work remains to be done. When the documentation for the **`WebService`** is complete, as partially shown in the code box above, the Documentation Coverage report for the ***BizFE*** application is as shown in Figure 20.3, below, where we have achieved 100% coverage for the **`WebService`**, but not yet started the documentation of any of the other elements.



Documentation coverage

documentation 9%

File	Type	Identifier	Statements
src/app/services/business-data.ts	injectable	BusinessData	0 % (0/11)
src/app/components/test-web-service/test-web-service.ts	component	TestWebService	0 % (0/11)
src/app/components/navigation/navigation.ts	component	Navigation	0 % (0/1)
src/app/components/home/home.ts	component	Home	0 % (0/1)
src/app/components/businesses/businesses.ts	component	Businesses	0 % (0/7)
src/app/components/business/business.ts	component	Business	0 % (0/20)
src/app/components/authuser/authuser.ts	component	Authuser	0 % (0/3)
src/app/components/authbutton/authbutton.ts	component	Authbutton	0 % (0/5)
src/app/app.ts	component	App	0 % (0/4)
src/app/app.config.ts	variable	appConfig	0 % (0/1)
src/app/services/web-service.ts	injectable	WebService	100 % (7/7)

Figure 20.3 Documentation Coverage

Do it now!	View the Documentation Coverage report for the ***BizFE*** application. Provide new documentation as JSDoc comments for other Components and Services and verify how the coverage figure rises as documentation elements are added.

20.3 Further Information

- https://compodoc.app/
  Compodoc Home Page

- https://github.com/compodoc/compodoc
  Compodoc GitHub Repository

- https://fr.linkedin.com/in/vincentogloblinsky/en
  Vincent Ogloblinsky (Compodoc developer)

- https://medium.com/@erickzanetti/how-to-implement-compodoc-in-your-angular-project-0a2d0a05d61f
  How to Implement Compodoc in your Angular Project

- https://www.youtube.com/watch?app=desktop&v=k1EDQfkMu2c
  Angular Project Documentation with Compodoc (YouTube)

- https://www.youtube.com/watch?v=6cjaJBMmZLs
  Effortless Software Documentation with Compodoc

- https://alternativeto.net/software/compodoc/about/
  Compodoc – The Missing Documentation Tool for Angular

- https://medium.com/@martink_rsa/js-docs-a-quickstart-guide-da6ce5df4a73
  JSDocs: A Quickstart Guide

- https://compodoc.app/guides/documentation-coverage.html
  Compodoc Documentation Coverage

- https://blog.angulartraining.com/how-to-generate-documentation-for-your-angular-porjects-c58cd4b4664b
  How to Generate Documentation for your Angular Projects