Angular Introduction

Agenda

What is a Single Page Application? – SPA

Angular Introduction

Angular Command Line Interface – Angular CLI

Angular Technology Stack

Node.js, npm, WebPack

Debugging Angular

Bootstrapping Angular / Project Configuration using Node.js

Updating Angular Projects

What is a Single Page Application - SPA

What is a Single Page Application (SPA)

A web application that is implemented using a single *.html-file

- Loads the content (template) of the URL-Segments (Routes) into a Div-Like container

Providing a more fluent user experience - similar to a desktop application

In a SPA, either all necessary assets like HTML, JavaScript, and CSS –

- Are retrieved with a single page load, or
- Resources are dynamically loaded and added to the page, usually in response to user actions

Most SPA‘s are implemented using JavaScript Frameworks like

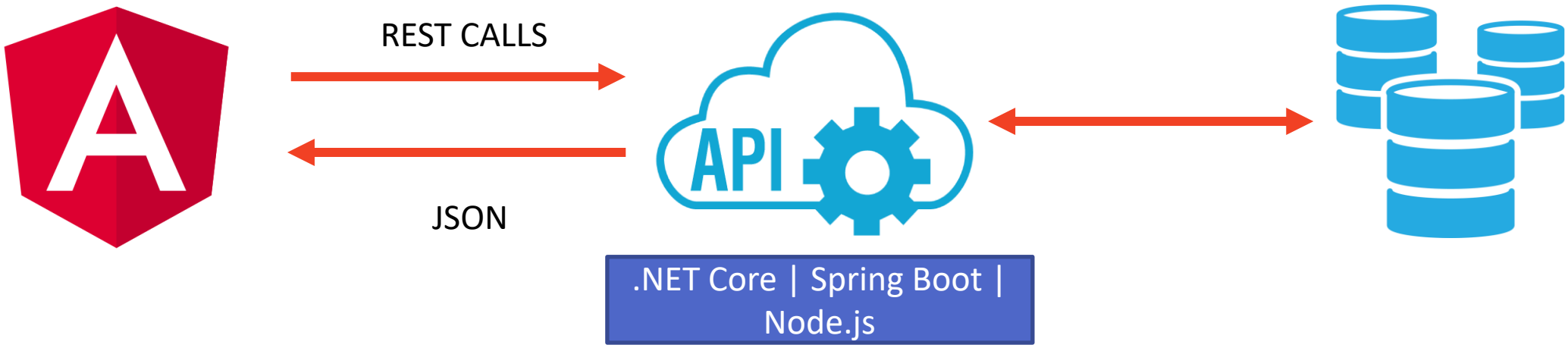
- Angular -> Google
- React -> Facebook
- Vue.js -> Started by Evan You, Ex-Google from Angular JS

Angular SPA Architecture

Provides the Front-End / User Interface

- Runs in a Container / Blob Storage / on a Web Server
- Is secured be Token Based Authentication ->
  - Forwards the token in the Header of the HttpRequest to the REST Api that consumes the same Identity Provider

Typically consumes one or more REST Apis that return JSON



Comparing SPA / Multi Page Application

SPA advantages over MPA:

- Faster page loading times with lazy loading strategies
- Improved user experience because of background data loading from server
- No need to write code for server side page rendering (but new SPA frameworks can do even this)
- Decoupling of front-end and back-end development

SPA disadvantages to MPA:

- Heavy client frameworks which are required to be loaded to the client
- Full Code is exposed to potential malicious user
- SEO (search engine optimization) implications: since your pages are built in the browser, the search engine crawler will see a different version of the page than your users

Angular Introduction

What is ANGULAR?

- A Single Page App (SPA) Framework maintained by Google
- Enhances HTML by attaching directives, custom tags, attributes, expressions, templates within HTML.
- Encourage TDD & Client Side MVC/MVVM design pattern
- Current version 13.x - Released Nov 2020
- ½ year release cycle - sometimes with delay :-)

Angular vs Angular JS

Angular

- Some concepts of Angular JS, but otherwise Complete Rewrite
- Typically implemented in TypeScript - Dart and JS possible
- Uses Node.js as Developer Runtime
- Heavily based on Observables

Angular JS (1.x)

- Implemented mostly in pure JavaScript
- Standalone Libraries

There is nothing like Angular JS 2, 6, 8 ... !

- Kind of "I dont know anything about Angular Outing ..." :-)

Why Angular

Angular has a large Ecosystem helping you fulfilling a variety of tasks

Scaffolding -> CLI
- Schematics

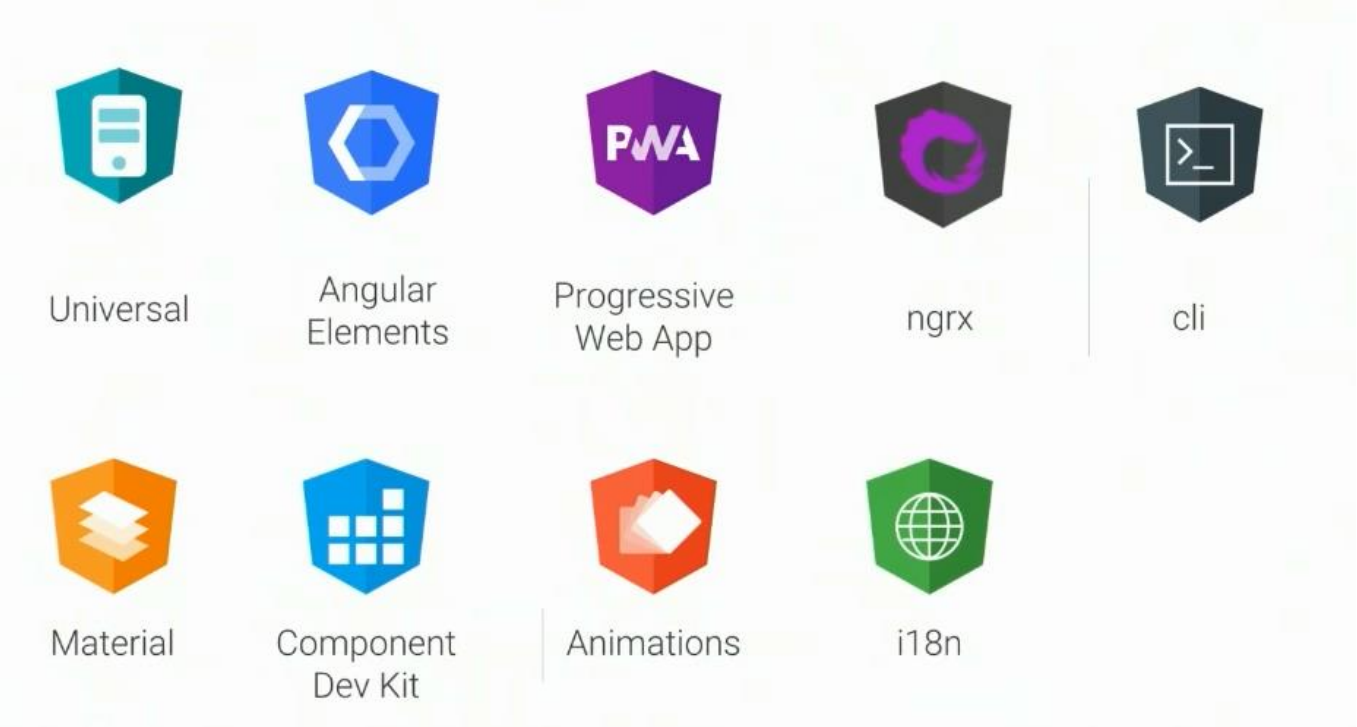
UI -> Material

State management -> NGRX

MicroPages -> Elements

Offline -> Progressive

….



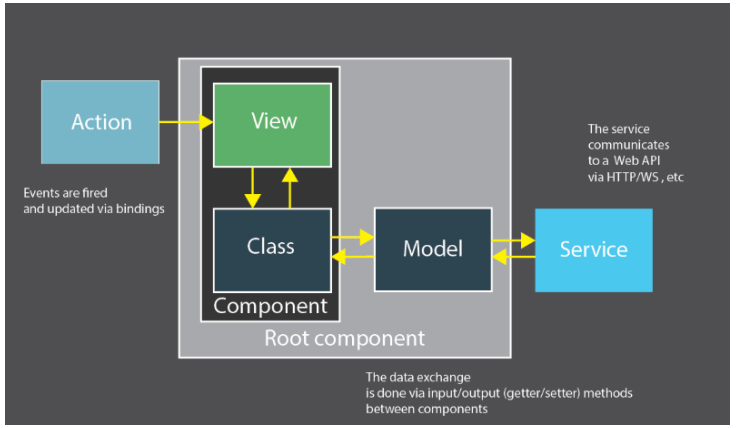
Component

An Angular App consists of one or more [nested] components

It defines:

- A TypeScript Class that acts as Controller
  - Defines Metadata like a selector using a Decorator -> @Component
- View: HTML | Inline
- Styles: CSS | SCSS
- @Input | @Output are used to exchange data with parent Components
- …

```
@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.scss']
})
export class HomeComponent implements OnInit {
```



Routing

Routing is the process to switch from one view to another

- ... From one component to another
- Routing is achieved using Angular Router (V3)

Main Routing is configured in app-routing.module.ts

- Each module can have its own routing

Router Links are used for navigation

```
const routes: Routes = [
  {
    path: '',
    component: HomeComponent,
  },
  {
    path: 'skills',
    component: SkillsListComponent,
  },
  {
    path: 'skills/:id',
    component: SkillsEditComponent
  },
```

```
<nav>
  <a routerLink="/">Home</a>
  <a routerLink="/skills">Skills</a>
</nav>
<div>
    <router-outlet></router-outlet>
</div>
```

Angular CLI

What is Angular CLI

Command Line Interface used to manage Angular projects

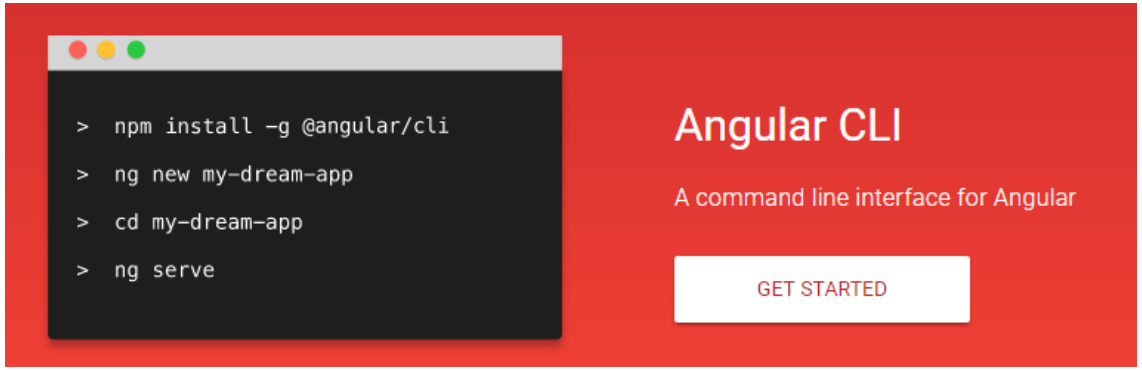
Installation

- npm install –g @angular/cli

Common Commands

- ng new -> Scaffolding a new project
- ng generate component -> create a component
- ng serve - > run the app
- ng add / ng update -> add packages / update proj

Documentations @ https://cli.angular.io/



ng new

Creates a new Angular project

Some interesting options:

- -- routing
  - Generates a separate routing module
- -- skip-install
  - Skips installation of npm files
- --prefix
  - Sets the prefix of selectors – default: "app"

ng generate

Used to create Angular artifacts like

- class
- component
- directive
- enum
- guard
- interface
- module
- pipe
- service
- library

General Syntax:

- ng generate component [name]
- ng g c [name]

See Expected Output: --dry-run

Do not create specs: --skipTests= false

```
PS H:\Playground\screenshot> ng g c demos/flexbox -m app.module.ts --skipTests=false --dry-run
CREATE src/app/demos/flexbox/flexbox.component.html (22 bytes)
CREATE src/app/demos/flexbox/flexbox.component.spec.ts (635 bytes)
CREATE src/app/demos/flexbox/flexbox.component.ts (280 bytes)
CREATE src/app/demos/flexbox/flexbox.component.scss (0 bytes)
UPDATE src/app/app.module.ts (485 bytes)
```

ng serve / ng build

ng serve

- Compiled output is served from memory, not from disk
- Does NOT include all project files
- Runs on http://localhost:4200 by default
- Runs in watch mode

ng build

- Writes output to dist/ folder

ng add

- Adds packages just like npm install, and also
- Using Schematics, also used by Angular CLI, to creates requires config files and registrations in
  - angular.json, app.module.ts
  - ...
- Available for
  - Angular Material,
    - ng add @angular/material
  - NgRx State Management
    - ng add @
- Possible to write custom schematics: -> ng add @briebug/jest-schematic

Angular Technology Stack

Angular Technology Stack

Runtime / Package Management

- Node.js, NPM

Language

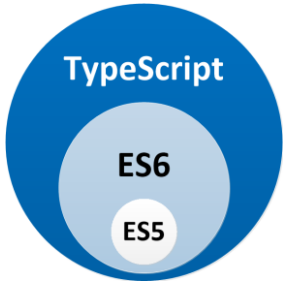
- TypeScript (ES 6, Dart)

Templating / Dependencies

- Angular CLI

Bundling

- Webpack



Common Editors

Any Editor that has integrated Support for Node.js

Editor choice is often result of available Plugins

- Visual Studio Code
  - Make sure you have Angular Language Service Extension installed
- IntelliJ IDEA & WebStorm from JetBrains
- Stackblitz
  - Online Editor used for prototyping and Online Questions
- Any other editor of choice that supports Node based Development

Visual Studio Code

Free, Open Source, lightweight cross plattform Editor

Built on top of GitHubs Electron platform using TypeScript

Out-of-box integration of GitHub & Node.js

Optimized to enable almost any kind of Application and Task

- HTML, TS, JS, C#, Java, Angular, PowerShell, Phyton, ...
- Docker, DevOps Pipelines, Cloud, …

Extensions (Plugins) for almost anything

Get from https://code.visualstudio.com/

VS Code Shortcuts & Settings

Ctrl + p	→ Quick open
Ctrl + Shift + f	→ Find in Files
Ctrl + K S	→ Save All
Ctrl + ö	→ Open Terminal Window
Shift + #	→ Toggle Comment
Alt + Shift + A	→ Toggle Block Comment

Complete VS Code Shortcut Guide:

https://code.visualstudio.com/docs/getstarted/keybindings

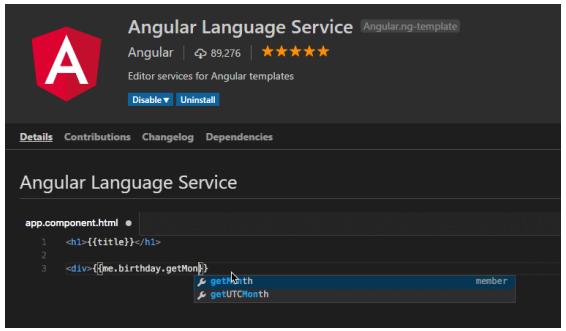
https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf

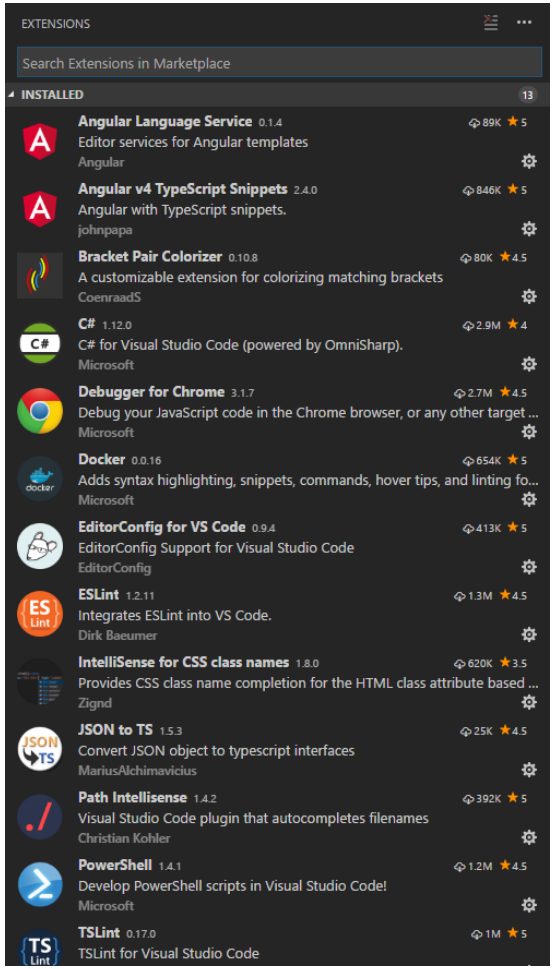
https://github.com/Microsoft/vscode-tips-and-tricks

Useful VS Code Extensions

Extensions make VS Code development easier

- Angular Language Service
- Several Angular Snippets Extensions
- Angular Flex Layout Snippets
- Debugger for Chrome
- Angular Console -> Scaffolding
- Angular Schematics
- ...





Extension Management CLI

Extensions can be managed using UI or by cli commands:

List

- code --list-extensions

Install (VSIX install possible)

- code --install-extension NAME ie: Angular.ng-template

Uninstall

- code --uninstall-extension NAME ie: ms-vscode.csharp

Node.js

Node.js as Runtime for Dev Toolset

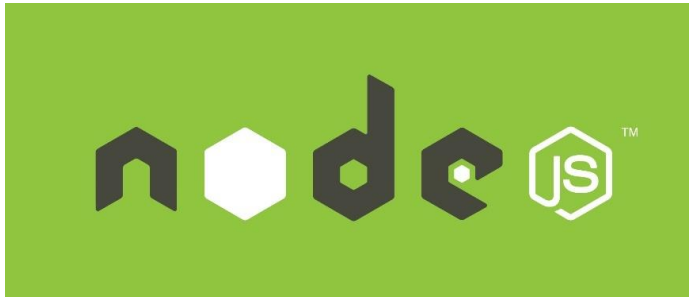
Built on Chrome's JavaScript V8 runtime

Used for building fast and scalable network applications

Uses event-driven, non-blocking I/O model that is lightweight and efficient

Used to build / as:

- I/O bound, Data Streaming Applications
- JSON APIs
- Single Page Applications
- Runtime for Developer Toolset



Nodes Role in Angular Dev

Node.js acts as a Runtime Host for Dev Toolset

It provides:

- Package Management
- TS / SASS Compilation
- Bundling
- ...

package.json

The configuration file for node.js

Defines libs used at runtime or devtime

packages are saved to node_modules

- npm install xxx --save | -S
- npm install xxx --save-dev | -D

npmjs.com is the package repository for node

Node.js Basic Setup

Install Node.js from https://nodejs.org/

From any folder execute

- npm install –g @angular/cli
- installs Angular CLI globally
- npm install –g typescript

From within the root folder of the project execute

- npm i ... short for install
  - installs all dependencies listed in package.json

Node Versions Management

Sometimes you need to run different Node Versions on the same machine

- Different Versions of e.g SPFx (Sharepoint) where created using different Node.js Versions and sometimes do not compile well when using other versions
- Drop 0.x – 1.4.x were create using Node.js version 6.3.1
- Drop 1.5.1 – 1.6x support Node.js version 8.11.3

nvm-windows supports running several Node versions on the same windows machine

- https://github.com/coreybutler/nvm-windows
- nvm install 6.3.1
- nvm use 6.3.1
- For mac / linux use : http://nvm.sh/

Debugging Angular

Debugging Angular

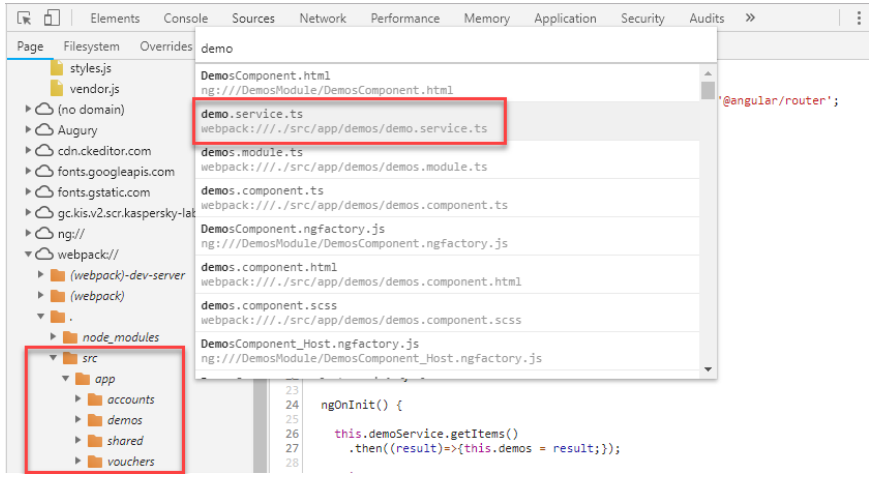
Use "Ctrl + P" and FILENAME in Chrome Dev tools to directly navigate to a file you want to debug

- TypeScript Debugging done using Source Maps

For Debugging in VSCode

- Install Debugger for Chrome Ext
- Create launch.json in .vscode-folder
- Add Debug config
- Run ng serve before Debugging!

```
{
  "name": "ng serve",
  "type": "chrome",
  "request": "launch",
  "url": "http://localhost:4200/",
  "webRoot": "${workspaceFolder}"
},
```



Angular 9+ Debug Statements

@angular/core/global exposes a set of functions which are useful for debugging the current state of your application

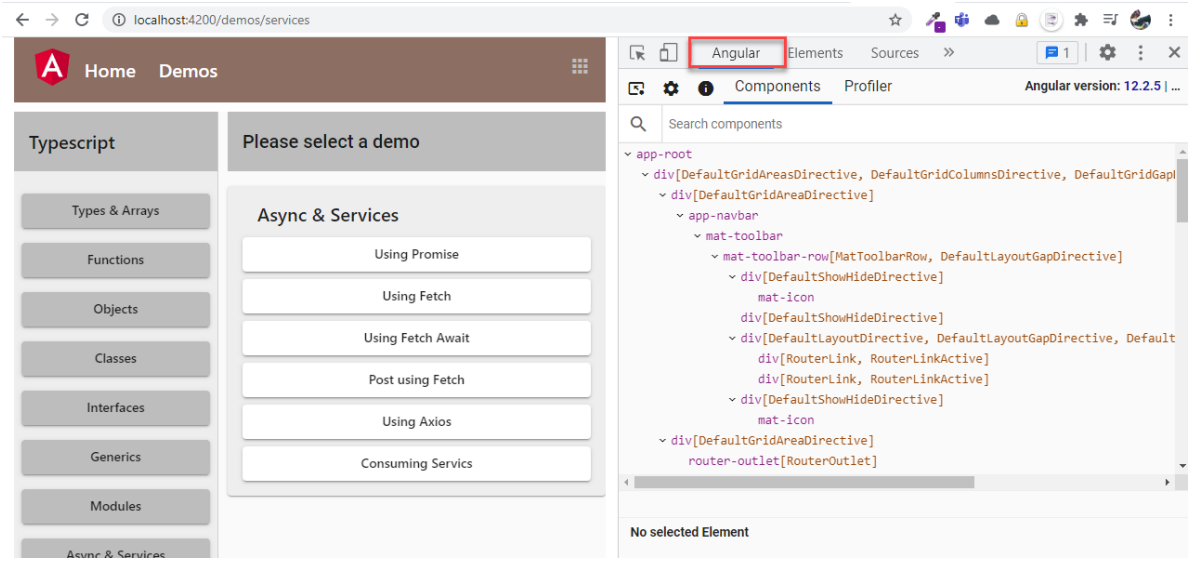
`ng.applyChanges`	Marks a component for check (in case of OnPush components) and synchronously performs change detection on the application this component belongs to.
`ng.getComponent`	Retrieves the component instance associated with a given DOM element.
`ng.getContext`	If inside an embedded view (e.g. `*ngIf` or `*ngFor`), retrieves the context of the embedded view that the element is part of. Otherwise retrieves the instance of the component whose view owns the element (in this case, the result is the same as calling `getOwningComponent`).
`ng.getDirectives`	Retrieves directive instances associated with a given DOM element. Does not include component instances.
`ng.getHostElement`	Retrieves the host element of a component or directive instance. The host element is the DOM element that matched the selector of the directive.
`ng.getInjector`	Retrieves an `Injector` associated with an element, component or directive instance.
`ng.getListeners`	Retrieves a list of event listeners associated with a DOM element. The list does include host listeners, but it does not include event listeners defined outside of the Angular context (e.g. through `addEventListener`).
`ng.getOwningComponent`	Retrieves the component instance whose view contains the DOM element.
`ng.getRootComponents`	Retrieves all root components associated with a DOM element, directive or component instance. Root components are those which have been bootstrapped by Angular.

Angular DevTools

Chrome Extension that allows to inspect the Angular App

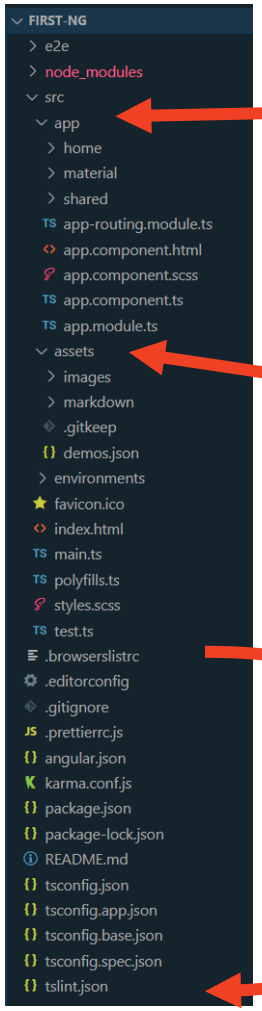
Use Angular Debug Statements

See guide at https://angular.io/guide/devtools



Bootstrapping & Configuration

Project Structure



App Code:

- app.module.ts
- app.component.ts

Assets:

- Used for shared assets - pictures, json, ...

Configuration:

- package.json, tsconfig.json, angular.json

angular.json

Configuration file for Angular CLI

Contains settings for

- root: root folder
- assets: assets to bundle
- styles: global styles
- scripts: global script imports
- defaults:
  - ng config schematics.@schematics/angular.component.spec false
- outDir: dir to compile to using ng-build

package.json

package.json is the configuration file for node.js

Consists of:

- Metadata
- Scripts
  - Automate custom Tasks
- Dependencies
  - Packages needed at runtime
- DevDependencies
  - Packages needed at runtime -> „Dev Tools“

```
{
  "name": "first-ng",
  "version": "0.0.0",
  ▷ Debug
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~10.0.9",
    "@angular/cdk": "^10.1.3",
    "@angular/common": "~10.0.9",
    "@angular/compiler": "~10.0.9",
    "@angular/core": "~10.0.9",
    "@angular/flex-layout": "^10.0.0-beta.32",
    "@angular/forms": "~10.0.9",
    "@angular/material": "^10.1.3",
    "@angular/platform-browser": "~10.0.9",
    "@angular/platform-browser-dynamic": "~10.0.9",
    "@angular/router": "~10.0.9",
    "rxjs": "~6.5.5",
    "tslib": "^2.0.0",
    "zone.js": "~0.10.3"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~0.1000.6",
    "@angular/cli": "~10.0.6",
    "@angular/compiler-cli": "~10.0.9",
    "@types/node": "^12.11.1",
    "@types/jasmine": "~3.5.0",
    "@types/jasminewd2": "~2.0.3",
    "codelyzer": "^6.0.0",
    "jasmine-core": "~3.5.0",
    "jasmine-spec-reporter": "~5.0.0",
    "karma": "~5.0.0",
    "karma-chrome-launcher": "~3.1.0",
    "karma-coverage-istanbul-reporter": "~3.0.2",
    "karma-jasmine": "~3.3.0",
    "karma-jasmine-html-reporter": "^1.5.0",
    "protractor": "~7.0.0",
    "ts-node": "~8.3.0",
    "tslint": "~6.1.0",
    "typescript": "~3.9.5"
  }
}
```

tsconfig.json & tsconfig.app.json

Indicates the root of a typescript project

- tsconfig.app.json inherits from tsconfig.json
  - tsconfig.json inherits from tsconfig.base.json (ng10+)
- tsconfig.app.json used in multi project workspace

Specifies the compiler options

Most common used settings

- sourceMap / target
- lib
  - Sets ECMA Script Syntax versions for newer language features

```
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "module": "esnext",
    "moduleResolution": "node",
    "importHelpers": true,
    "target": "es2015",
    "typeRoots": [
      "node_modules/@types"
    ],
    "lib": [
      "es2018",
      "dom"
    ]
  },
  "angularCompilerOptions": {
    "fullTemplateTypeCheck": true,
    "strictInjectionParameters": true
  }
}
```

src/main.ts

Bootstraps the Angular Application

- Injected into index.html

Loads the AppModule & the RootComponent

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

app.module.ts

An ngModule acts as a Container that organizes / provides artifacts for an app that must be registered in the module so that they can be used

Consists of sections:

- Imports
  - Other Modules that provide artifacts that you might want to use
- Declarations
  - Components, Pipes, Directives
- Providers
  - Data- and Utility-Services
- Bootstrap
  - What to boot ...

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Updating Angular

ng update

Updating Angular can be done with CLI using „ng update“

- ng update @angular/cli [--allow-dirty] [--force]
- ng update also works on some 3rd party libs
  - ng update @angular/material @angular/flex-layout

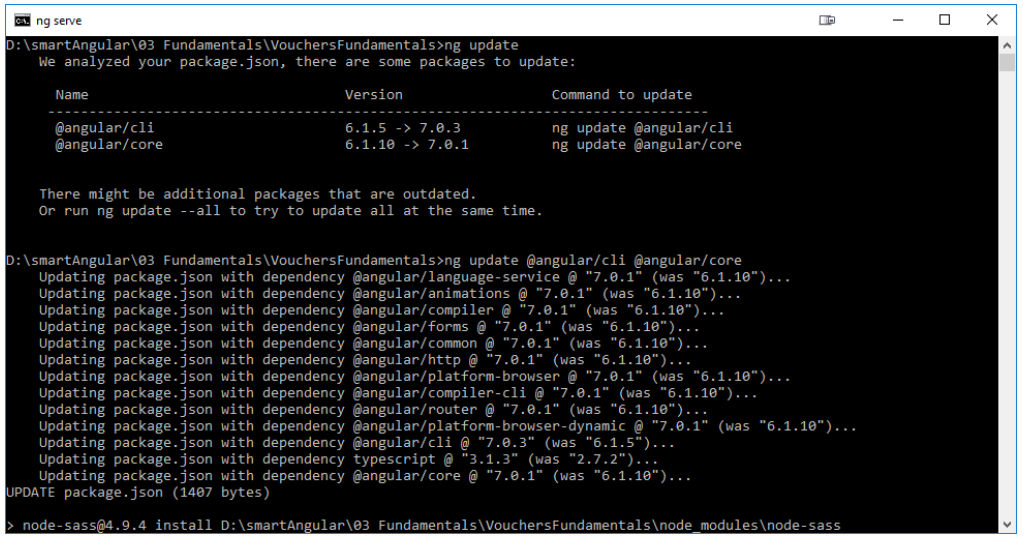
Updates Angular Libs & TypeScript

- Removes burden of doing it manually

Uses Angular Schematics to modify config files

- Might apply Schematics to migrate code

npm audit fix might fix vulnerabilities in packages



Angular Update Guide

Available at https://update.angular.io/

The Application tells you the steps needed to upgrade