Angular

Workshops part 1

avanade

Who we are?

Mateusz Dobrakowski – Frontend developer 4 years exp (as angular dev 2+ years of exp), loves games – video and board

Jakub Jarząbek – Fullstack developer 14 years exp (2+ years as angular dev). Loves watching F1 and books.

Elżbieta Wątroba – 2 years as angular dev. Loves books and movies

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Course agenda

Angular CLI - project setup with basics

Components - when to use, how to declare with its decorators

Role of AppModule and Component Declaration

Data bindings (string interpolation, property binding, event binding) - communication between components on different nesting levels

Directives - custom and build-in directives

Pipes - custom and build-in

Components Live cycle hooks

Services - why do we need them, declaration, injection

Routing

Guards

Observables

Forms with validation

HttpClient with error handling and interceptors

Authentication

NgRx - application state, reducers, actions, effects

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Agenda

- Prerequisites installation (if necessary)
- Quick demo
- Angular and Angular.js
- Basic configuration
- App module and app component
- Creating components
- One-way binding (two ways)
- Two-way binding

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

What do I need?

NodeJS, NPM

Angular CLI

Visual Studio Code (or other IDE but I show some tips and tricks for vscode)

avanade

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

How do I get it?

1. Node and node package manager – get package from https://nodejs.org/en/

Check if it works by: `node -v` and `npm -v`

2. Angular Cli help us speed up development and allow quickly boilerplate

```
npm install -g @angular/cli
```

3. `ng -version` to check if Angular is installed properly

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Creating new project

Create startup project:

```
ng new project_name
Would you like to add Angular routing? (y/N) y
Which stylesheet format would you like to use? SCSS
```

Start running basic server:

```
ng serve
```

Use Visual Studio code + Google Chrome, or any IDE you like.

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Sources

To see what's going on in the app while running:

-Open browser dev tools

-Go to Sources

-Hit `ctrl + p` to open any type script file from your project

This is possible because we did dev build which is using JIT compilation.

Of course build files for prod use AOT which is faster and protect your code from simple theft.

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Why do I even bother?

Pros:

-It's easy to build reausable code in app/several apps because we're focusing on reusable components

-Architecture similar on different projects because of convention over configuration approach

-Huge community and Google support

-Use of TypeScript

-Lot of ready components (material)

-Two-way binding

Cons:

-Slow because of using DOM, this is changed by IVY engine available since Angular 9

-Learning curve is relatively high compared to React.js

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Angular and Angular

Angular and Angular

Angular.js

First version of the framework

We won’t talk about it

Initially developed on getangular.com then taken by Google

It is not developed anymore

Angular 2+ (currently 11)

New version of the framework

No compatibility between 1.x and 2.+

Developed by Google

Google uses Angular on over 600+ own projects like Firebase for example (state at middle of 2018)

Angular 2+

Always remember to look for Angular 2+ solutions/examples for your problems.

Further you will clearly see if Stackoverflow answer is about Angular.js or Angular 2+

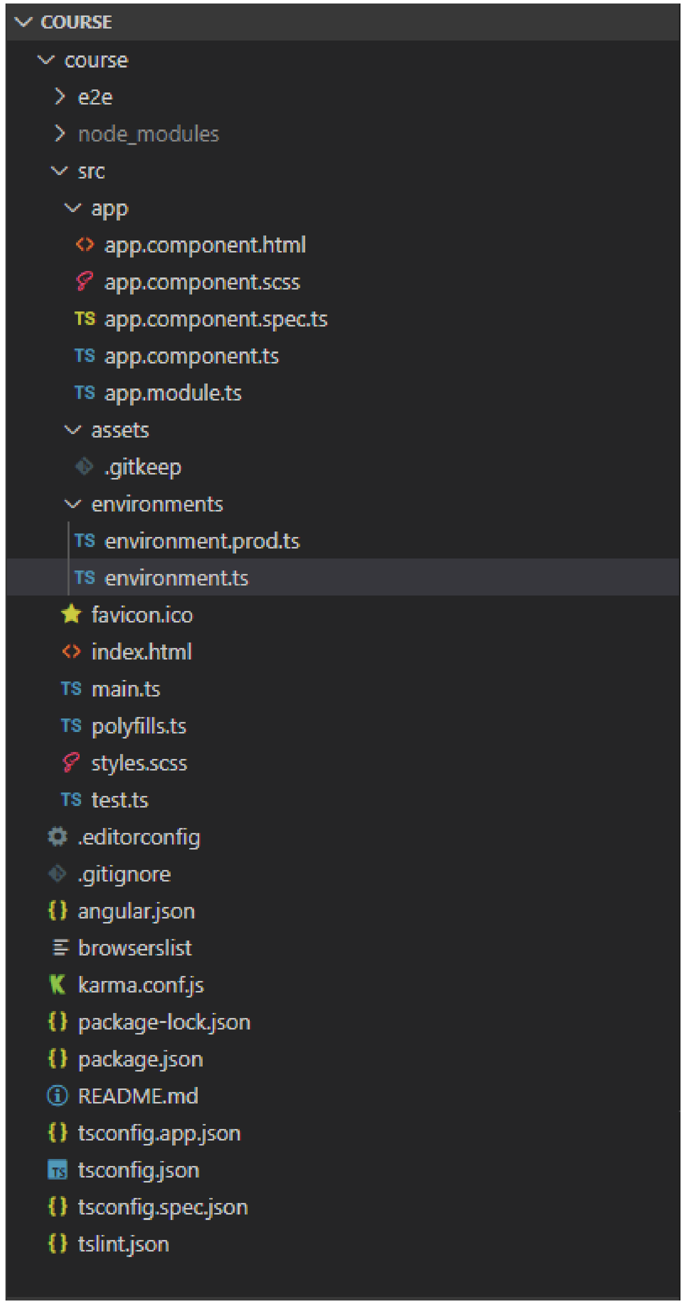
avanade

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Back to code

ng new command generates boilerplate of application which already works. How cool is that?

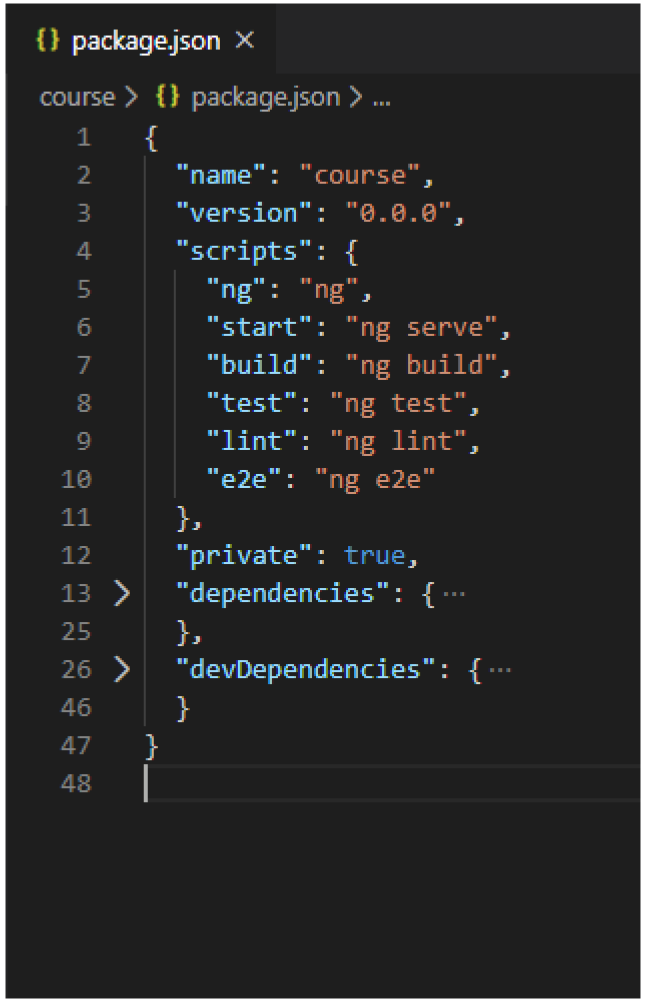
Let's take a look into some more important files to see what's happened.



```
COURSE
  course
    e2e
    node_modules
    src
      app
        app.component.html
        app.component.scss
        app.component.spec.ts
        app.component.ts
        app.module.ts
      assets
        .gitkeep
      environments
        environment.prod.ts
        environment.ts
      favicon.ico
      index.html
      main.ts
      polyfills.ts
      styles.scss
      test.ts
    .editorconfig
    .gitignore
    angular.json
    browserslist
    karma.conf.js
    package-lock.json
    package.json
    README.md
    tsconfig.app.json
    tsconfig.json
    tsconfig.spec.json
    tslint.json
```

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Configuration files



```
{
  "name": "course",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {···
  },
  "devDependencies": {···
  }
}
```

Package json is usual node package configuration file.

Usually inserts is added when using

npm install @angular/material

Where @angular/material is package name

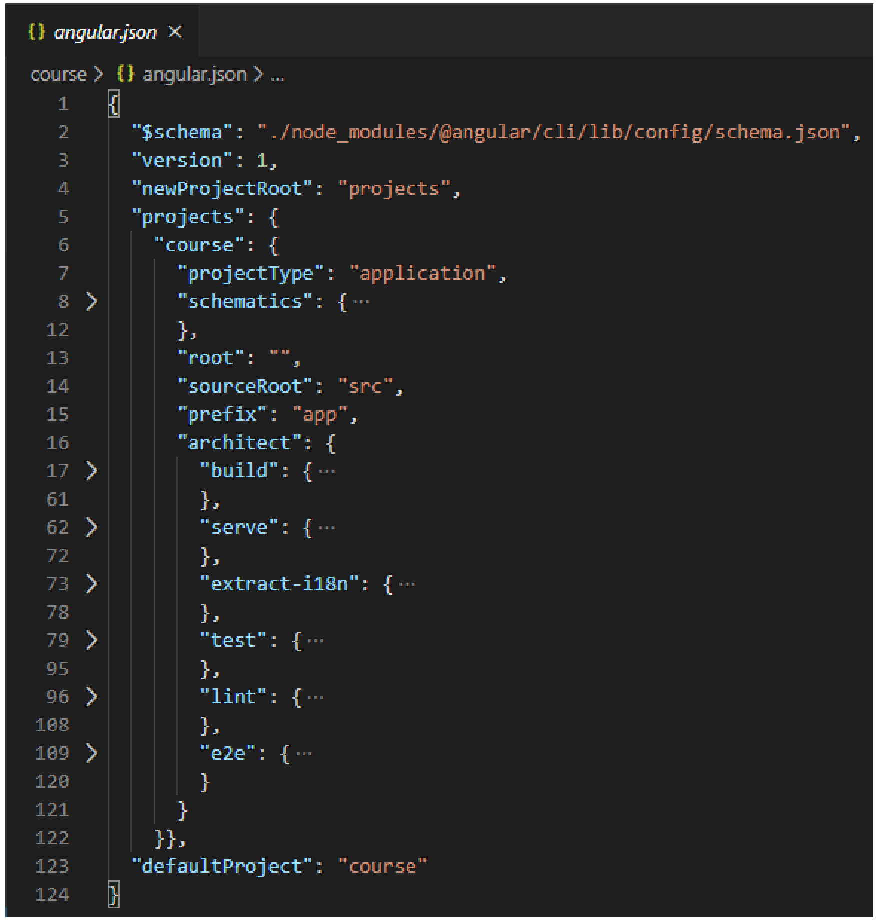
What is also interesting, we have scripts here.
You can use predefined scripts from here on console, for example:

npm run start will fire ng serve command

You can add your own command here for example.

avanade

Configuration files



```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "course": {
      "projectType": "application",
      "schematics": {⋯
      },
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {⋯
        },
        "serve": {⋯
        },
        "extract-i18n": {⋯
        },
        "test": {⋯
        },
        "lint": {⋯
        },
        "e2e": {⋯
        }
      }
    }},
  "defaultProject": "course"
}
```

angular.json file is responsible for our Angular project configuration.

Schema tells how it should look like but it's around 2k lines so we will skip most of the options for now.

Projects – here we have list of projects – what is interesting we have share code between those.

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Tip time

You can add following entry in tsconfig.json

```
 "typeRoots": [
      "node_modules/@types",
      "src/types"
    ],
```

which allow you to avoid importing interfaces from src/types when you create file with .d.ts extension. It is helpful when maintaining many imports in your .ts files

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Other configs and other files

Karma.conf.js – Karma test framework configuration file – will skip it for now.

Type script configuration files is also not very interesting at the begining.

Polyfills.ts you can configure targetting versions of browsers, and add you own polyfills here.

e2e – by default Angular uses Protractor to end to end testing.

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Summary

Angular set up working application with unit tests, e2e tests, basic components and default configuration (and routing, if you pick that option).

Time to dig up into Angular itself.

avanade

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

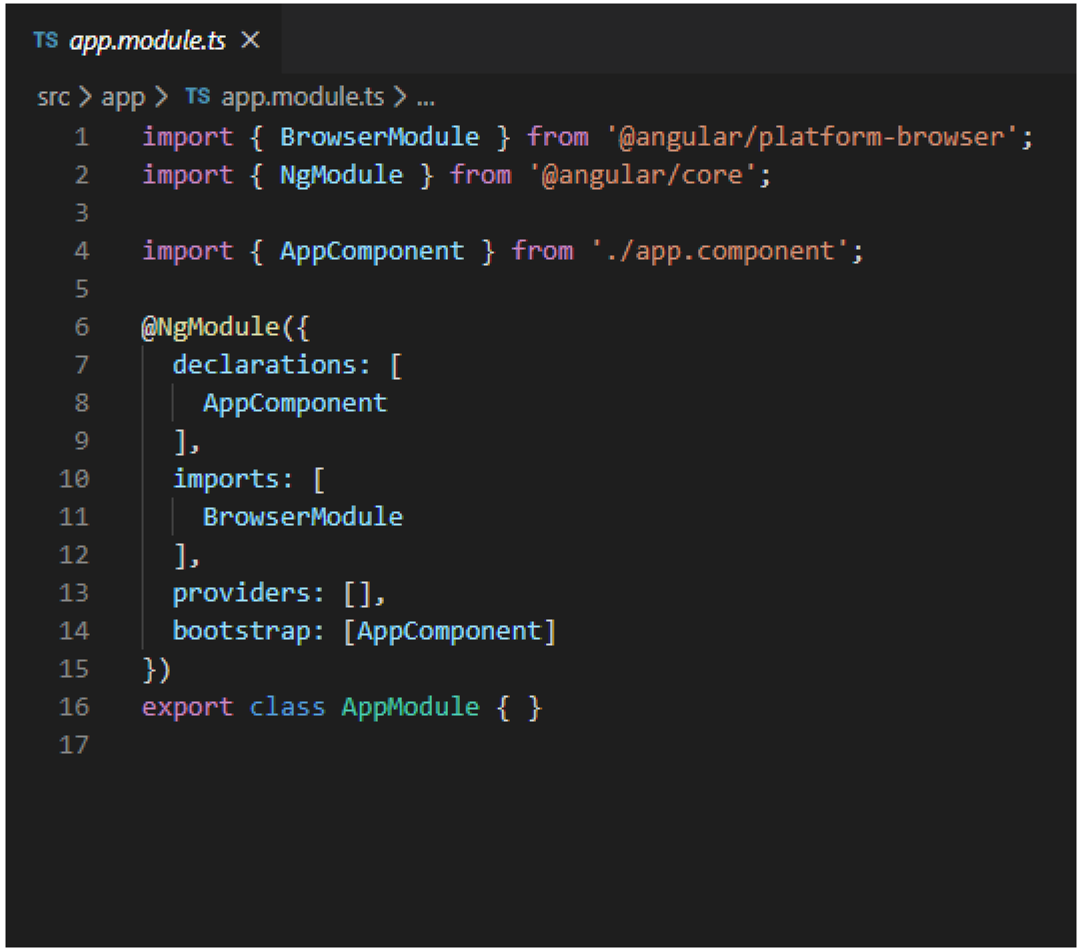
App.module.ts

Angular application is composed from modules which are then composed from components.

By default app.module is created and all created components are included in it.

Module in Angular is composed from several things:

- list of necessary imports
- @NgModule decorator
- Exported class

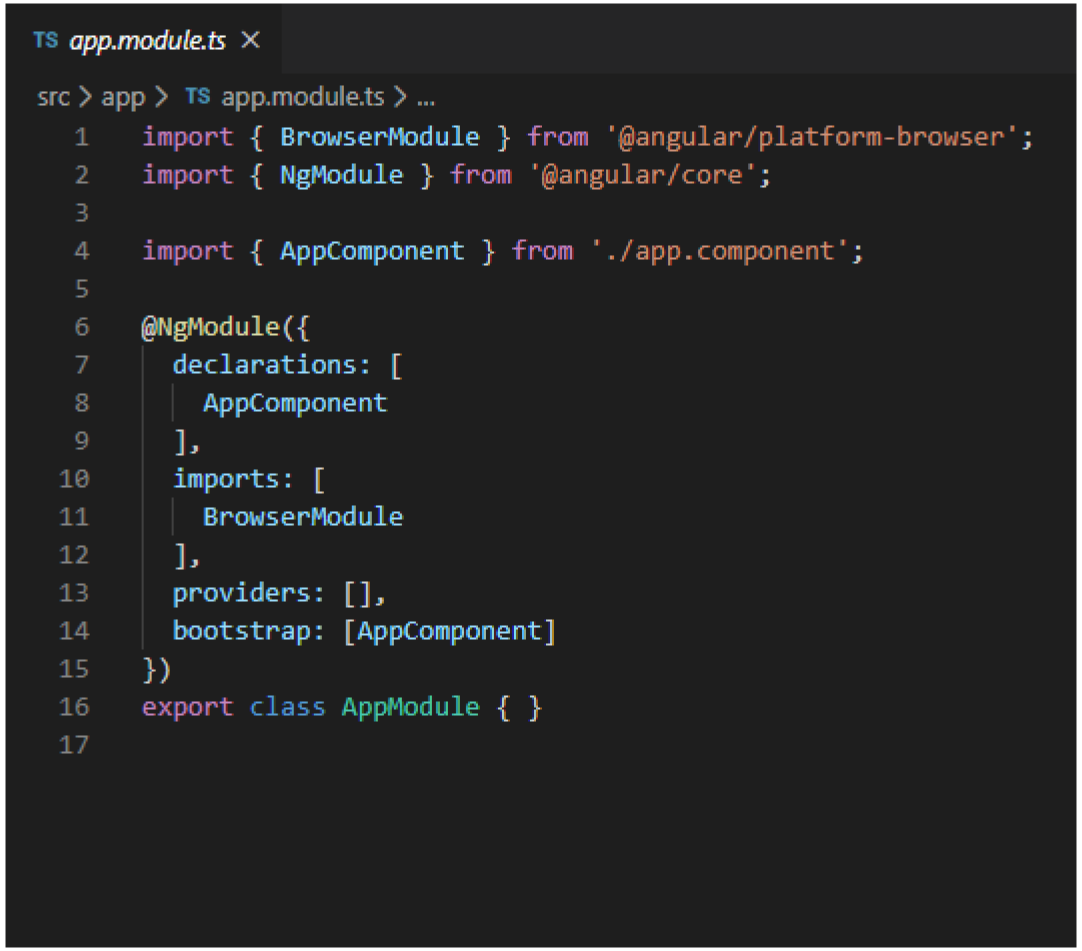


```typescript
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

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

```

app.module.ts

Inside decorator we have some arrays:

- declarations – here are listed Angular components/services/classes declared (be part of) this module
- Imports – another modules imports (Angular, material, application modules)
- Providers – for declaring injectables in specific module
- EntryComponents – for components that will be created in flight (no directives)
- Others like Bootstrap or id (enabled for ivy)



```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

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

```

app.component.ts

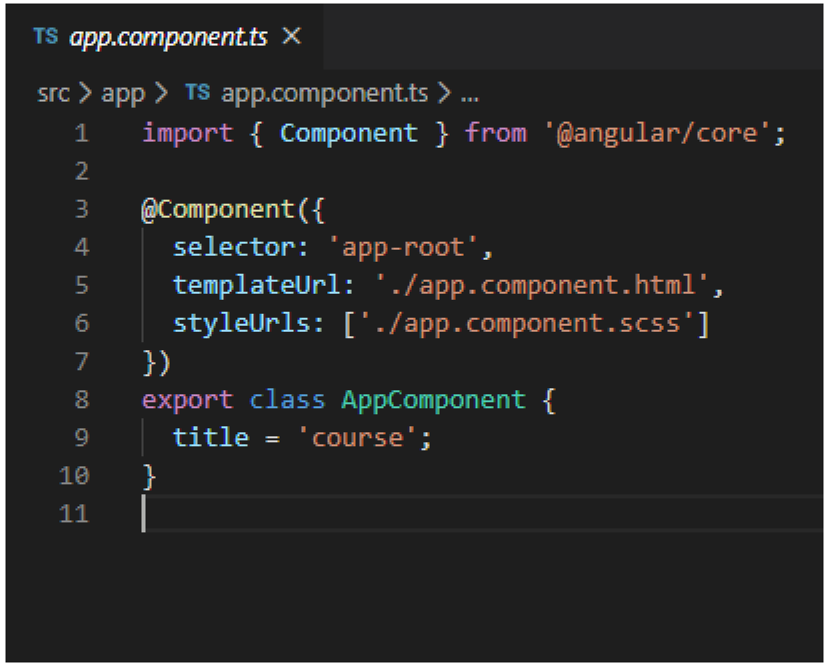
Again imports ;)

@Component decorator:
- selector – most important this is how we use component in different places

- templateUrl – html which is associated with component

- styleUrls: array of styles used to „style" our component

- animations: array of animations declared in specific format – allows entry and exit animations



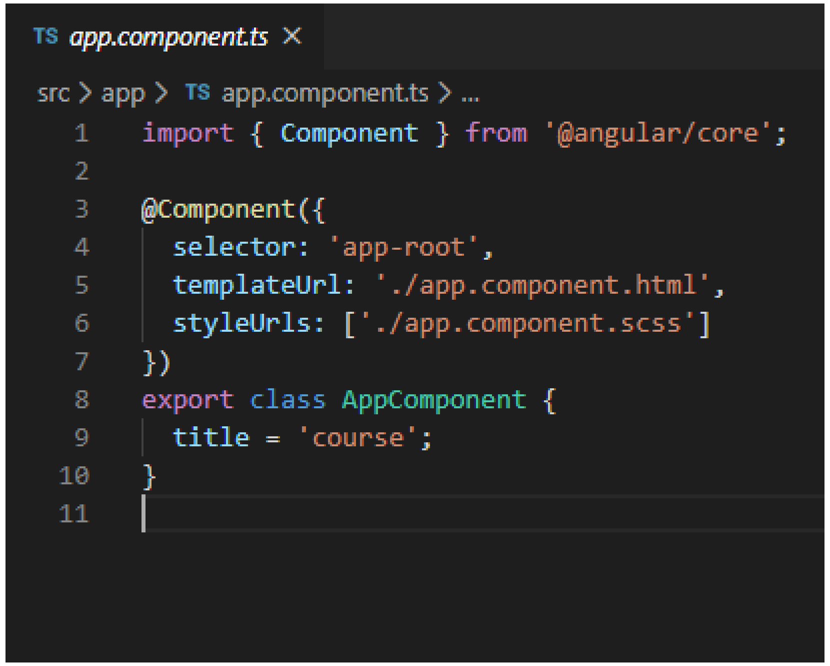
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'course';
}
```

app.component.ts

Class – our logic and data

Containing properties like title in this case and methods responsible for our logic

Constructor – injected services used by component – preferable logic should be here

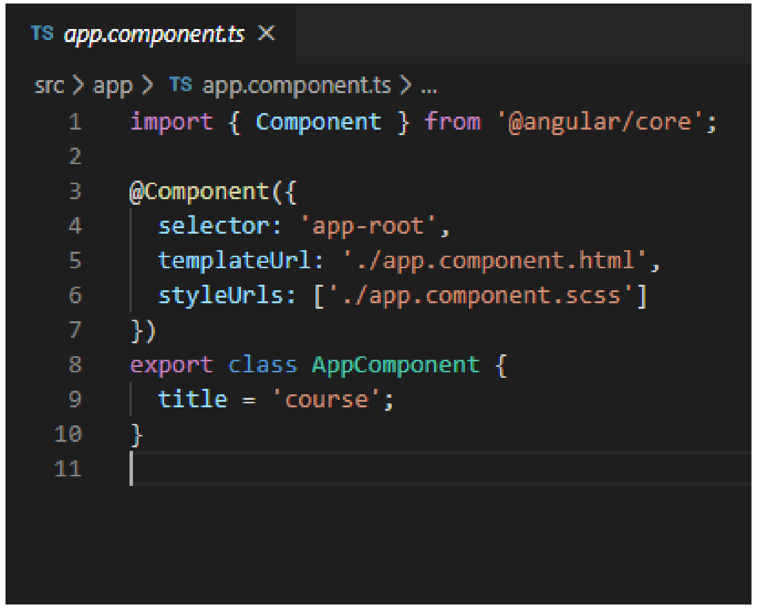


```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'course';
}
```

app.component.html

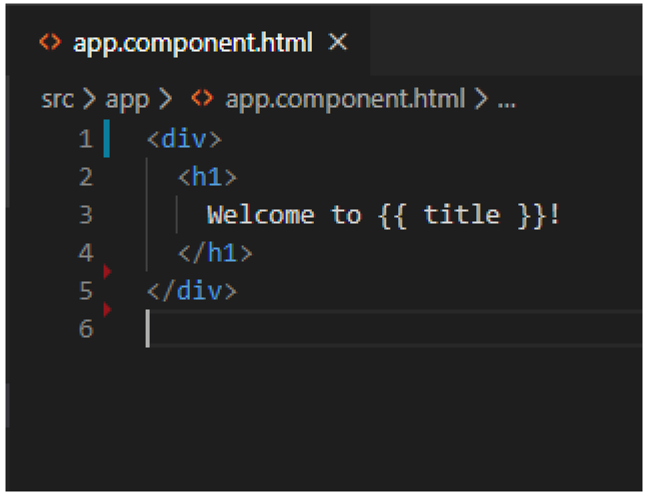
Binding – interpolation

Properties in component can be easily interpolated in HTML


TS app.component.ts ×

src > app > TS app.component.ts > ...

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'course';
}
```


app.component.html ×

src > app > app.component.html > ...

```
<div>
  <h1>
    Welcome to {{ title }}!
  </h1>
</div>
```

schematics

Provided by:

Angular

Material and others

Yourself?

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Components creation

Creating component is as easy as creating app thanks to schematics:

`ng generate component name` or simply `ng g c name`

This way you can create services, pipes, directives and others, we will see that later.

You can specify module to which you want to add created component by adding flag `--module` or `-m` and name of the module after. This is helpful when working with multiple modules in the app.

Create component named user-list in existing (or created in flight path)

```
ng g c components/user-list
```

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Tip time

You can easily toggle your visual studio code explorer and/or terminal.

Ctrl + b will toggle visibility of explorer

Ctrl + j or ctrl + ` will toggle terminal

Ctrl + + and ctrl + - will change font size.

This will help to see more code at a time when necessary.

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Why do we even create components?

To maximize fragmentation which lead into greater readability and easier maintainance.

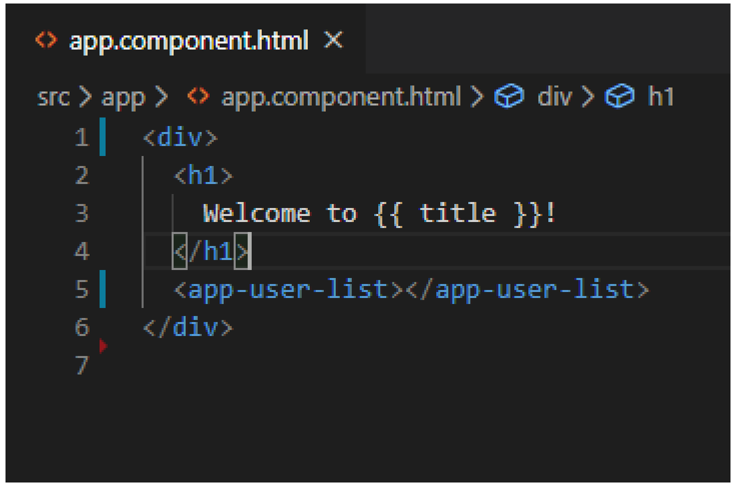
To reuse parts of code.

To allow developers work almost separately on the same application – without conflicts

Nest components

Nesting or using components is as easy as writing component name

Check it by opening your browser



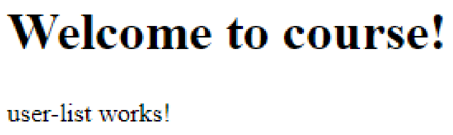
```
<div>
  <h1>
    Welcome to {{ title }}!
  </h1>
  <app-user-list></app-user-list>
</div>
```

It works!

On a side we can see that HTML from user-list components is nested inside app-root

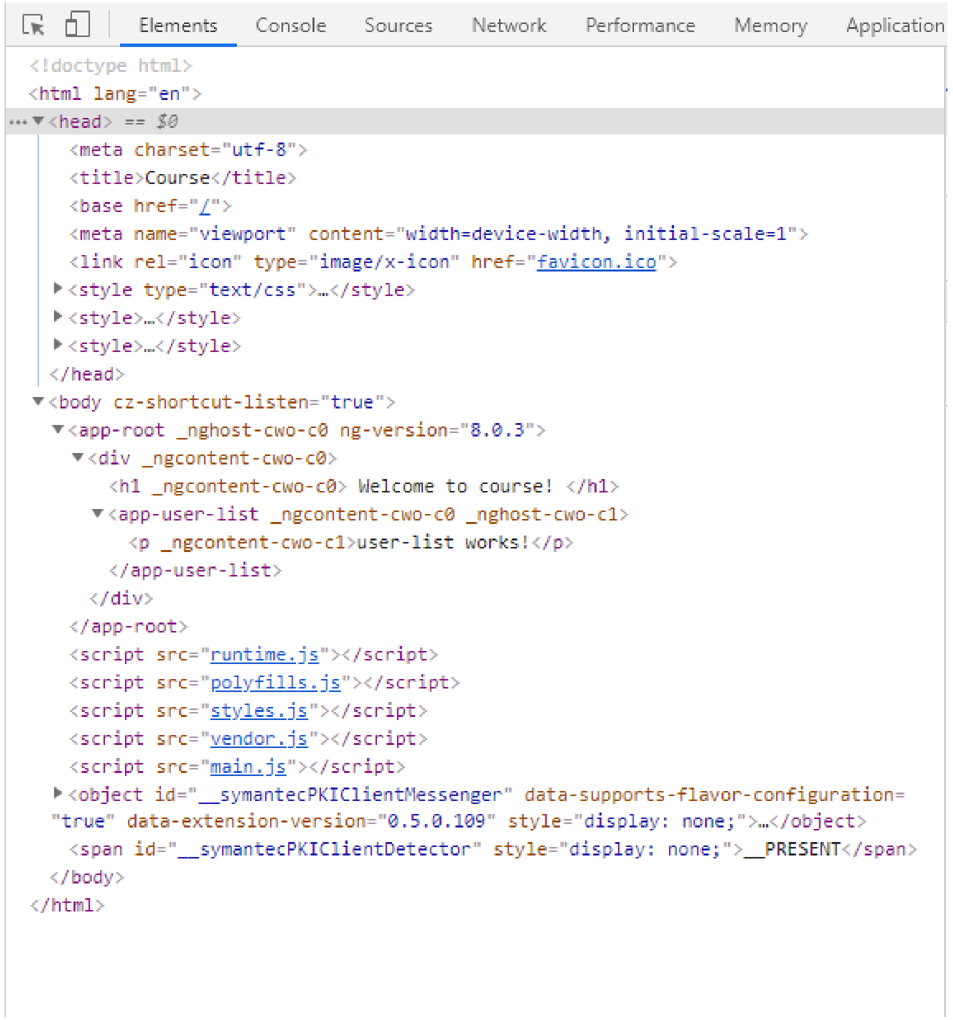
What is cool from dev perspective we can find out where HTML come from by simple finding parent in DOM. It allows styling components in an easier way.

We see some _nghost and _ngcontent it allows Angular to encapsulate the styles (they won't leak) identify components.



Welcome to course!

user-list works!



```
<!doctype html>
<html lang="en">
... ▼<head> == $0
    <meta charset="utf-8">
    <title>Course</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
   ▶<style type="text/css">…</style>
   ▶<style>…</style>
   ▶<style>…</style>
  </head>
 ▼<body cz-shortcut-listen="true">
   ▼<app-root _nghost-cwo-c0 ng-version="8.0.3">
     ▼<div _ngcontent-cwo-c0>
        <h1 _ngcontent-cwo-c0> Welcome to course! </h1>
       ▼<app-user-list _ngcontent-cwo-c0 _nghost-cwo-c1>
          <p _ngcontent-cwo-c1>user-list works!</p>
        </app-user-list>
      </div>
    </app-root>
    <script src="runtime.js"></script>
    <script src="polyfills.js"></script>
    <script src="styles.js"></script>
    <script src="vendor.js"></script>
    <script src="main.js"></script>
   ▶<object id="__symantecPKIClientMessenger" data-supports-flavor-configuration=
  "true" data-extension-version="0.5.0.109" style="display: none;">…</object>
    <span id="__symantecPKIClientDetector" style="display: none;">__PRESENT</span>
  </body>
</html>
```

One-way binding

We saw one-way binding using interpolation. We can do the same by binding to HTML properties.

```html
<div>
  <h1 title="{{title}}">
    Welcome to {{ title }}!
  </h1>
  <app-user-list></app-user-list>
</div>
```

```html
<div>
  <h1 [title]="title">
    Welcome to {{ title }}!
  </h1>
  <app-user-list></app-user-list>
</div>
```

Take me to the other side

user-list.component.html

```
<p>user-list works!</p>
on-click -> click
<button (click)="warnUser()">
  click me
</button>
```

user-list.component.ts

```
warnUser() {
    alert('You clicked me you bastard!');
}
```

Excercise

Create user component with property id: number and assign some value to it.

Print value in paragraph using interpolation.

Nest this component in list component 3-times.

avanade

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Two-way binding

user-list.component.html

```
<input type="text"
[(ngModel)]="inputValue"
style="display: block;">

<p>Hello {{inputValue}}!</p>
```

app.module.ts

```
import { FormsModule } from '@angular/forms';
imports: [
    BrowserModule,
    FormsModule
  ],
```

user-list.component.ts

```
inputValue: string = 'initial string';
```

Some useful resources

https://angular.io/ there is literally everything (sometimes condensed)

https://material.angular.io/ great UI library working with hammer.js usefull when you dont feel comfortable with styles.

https://www.youtube.com/watch?v=jnp_ny4SOQE&feature=youtu.be&t=1320 ivy compiler which will be default in v9

avanade

©2019 Avanade Inc. All Rights Reserved. <Highly Confidential>

Thank you

avanade