IN4MATX 133: User Interface Software

Lecture:
Separation in Angular

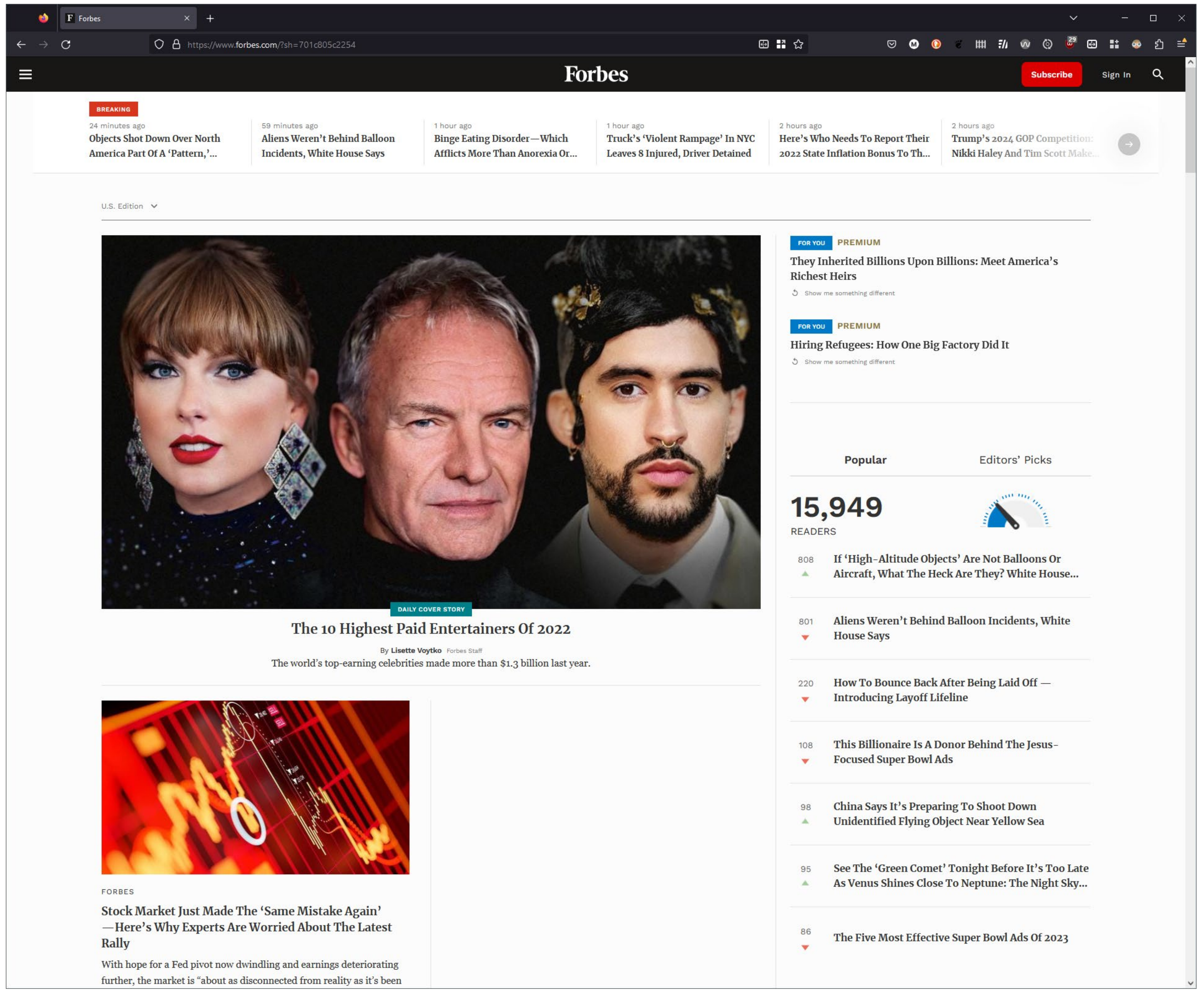
Today's goals

By the end of today, you should be able to…

- Differentiate and explain the roles of Angular components, modules, and services
- Implement a service in Angular
- Navigate Angular's file structure

A “large” client interface

- Hundreds of pages and ways to navigate between pages
- Repeated UI elements (status updates)
  - Angular implements these as *components*
- Different content, links, etc. displayed for each person



24 minutes ago
Objects Shot Down Over North America Part Of A ‘Pattern,’...

59 minutes ago
Aliens Weren’t Behind Balloon Incidents, White House Says

1 hour ago
Binge Eating Disorder—Which Afflicts More Than Anorexia Or...

1 hour ago
Truck’s ‘Violent Rampage’ In NYC Leaves 8 Injured, Driver Detained

2 hours ago
Here’s Who Needs To Report Their 2022 State Inflation Bonus To Th...

2 hours ago
Trump’s 2024 GOP Competition: Nikki Haley And Tim Scott Make...

They Inherited Billions Upon Billions: Meet America’s Richest Heirs

Show me something different

Hiring Refugees: How One Big Factory Did It

Show me something different

The 10 Highest Paid Entertainers Of 2022

By Lisette Voytko Forbes Staff

The world’s top-earning celebrities made more than $1.3 billion last year.

15,949
READERS

808	If ‘High-Altitude Objects’ Are Not Balloons Or Aircraft, What The Heck Are They? White House...
801	Aliens Weren’t Behind Balloon Incidents, White House Says
220	How To Bounce Back After Being Laid Off — Introducing Layoff Lifeline
108	This Billionaire Is A Donor Behind The Jesus-Focused Super Bowl Ads
98	China Says It’s Preparing To Shoot Down Unidentified Flying Object Near Yellow Sea
95	See The ‘Green Comet’ Tonight Before It’s Too Late As Venus Shines Close To Neptune: The Night Sky...
86	The Five Most Effective Super Bowl Ads Of 2023

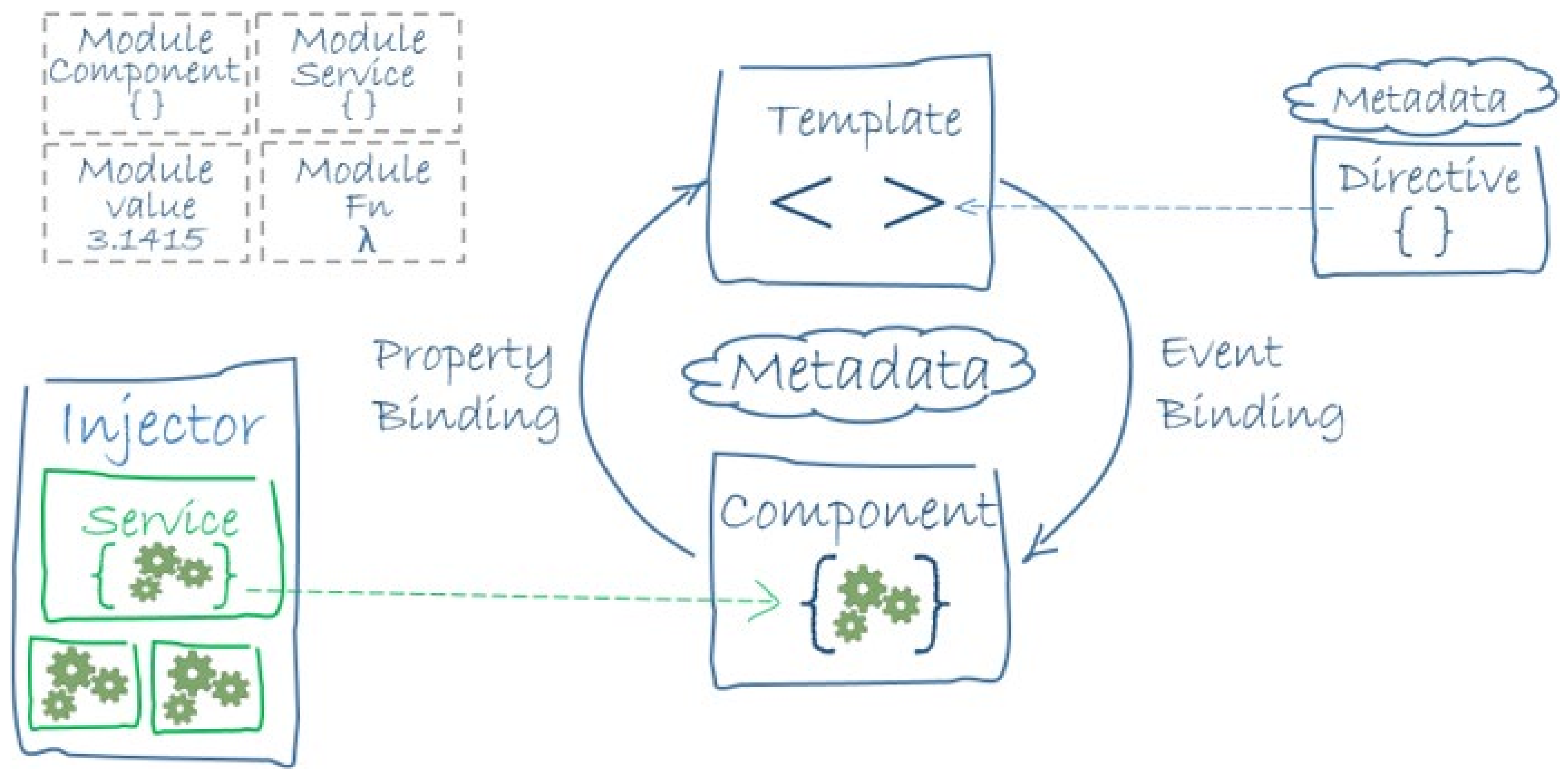
FORBES

Stock Market Just Made The ‘Same Mistake Again’—Here’s Why Experts Are Worried About The Latest Rally

With hope for a Fed pivot now dwindling and earnings deteriorating further, the market is "about as disconnected from reality as it's been

A “large” client interface

- Loading lots of libraries can be slow and expensive
- So Angular supports sectioning parts of projects into distinct modules



https://angular.io/guide/architecture

Angular modules

- Segment code into a library, similar to a JavaScript library
- A component only imports the modules it needs

Angular modules

- Segment code into a library, similar to a JavaScript library
- A component only imports the modules it needs

Angular modules

- By default, each Angular app has one module, `app.module.ts`
- But an app can create multiple modules to section off code
- `ng generate module [name]`
- Modules can *import* other modules
- Modules also *declare* which components they use
    - When you create a new component (`ng generate component [name]`), it automatically gets added to the declarations for the root module

Angular modules

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HelloComponent } from './hello/hello.component';
import { DayComponent } from './day/day.component';

@NgModule({
  declarations: [  ⬅Components used
    AppComponent,
    HelloComponent,
    DayComponent
  ],
  imports: [       ⬅Modules to import
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent] ⬅The “root” component of the module
})
export class AppModule { }
```

Angular modules

- `BrowserModule` is included by default
  - Required to run any app in the browser
- When creating an Angular project, can specify whether a *Routing* module should be created
  - Routing: defines what URIs to send to what **endpoints**
  - For Angular, defines what URIs to send to what **components**

Angular routing

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { ArtistPageComponent } from './pages/artist-page/artist-page.component';
import { TrackPageComponent } from './pages/track-page/track-page.component';
import { AlbumPageComponent } from './pages/album-page/album-page.component';
import { HomePageComponent } from './pages/home-page/home-page.component';

const routes: Routes = [
 { path: 'artist/:id', component: ArtistPageComponent},
 { path: 'track/:id', component: TrackPageComponent},
 { path: 'album/:id', component: AlbumPageComponent},
 { path: '', component: HomePageComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Listens for any endpoint `artist/:id`

`id` can be retrieved in `album-page.component.ts`

Retrieving route in a component

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-album-page',
  templateUrl: './album-page.component.html',
  styleUrls: ['./album-page.component.css']
})
export class AlbumPageComponent implements OnInit {

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
   var albumId = this.route.snapshot.paramMap.get('id');
  }

}
```

“Injecting a service”

Retrieve the id from the URI

Angular services

- Anything not associated with a specific view should be turned into a *service*
  - e.g., getting data from an API, parsing URIs for routing information
- Helps keep components lightweight
- Services can then be *injected* into a component (importing)
- To inject, import the service and retrieve it as a parameter in the constructor
- `ng generate service [name]`

Angular services

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
```
Importing a service

```
@Component({
  selector: 'app-album-page',
  templateUrl: './album-page.component.html',
  styleUrls: ['./album-page.component.css']
})
export class AlbumPageComponent implements OnInit {

  constructor(private route: ActivatedRoute) { }
```
Injecting it

```
  ngOnInit() {
   var albumId = this.route.snapshot.paramMap.get('id');
  }
}
```
Service can be referenced later

Angular services

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

@Injectable({
  providedIn: 'root'
})
export class SpotifyService {
 baseUrl:string = 'http://localhost:8888';

  constructor(private http:HttpClient) { }

  private sendRequestToExpress(endpoint:string) {
  }
}
```

Defined as injectable

Services can inject other services!

What module(s) can use this service

HttpClient injected

Import a custom service

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { SpotifyService } from '../../services/spotify.service';
```

Import service via file structure

```
@Component({
  selector: 'app-album-page',
  templateUrl: './album-page.component.html',
  styleUrls: ['./album-page.component.css']
})
export class AlbumPageComponent implements OnInit {

  constructor(private route: ActivatedRoute,
private spotifyService:SpotifyService) { }
```

Inject it like any other service

Angular classes

- Plain-old classes can also be made in Angular
  - Any processing or munging you need to do, for example
- `ng generate class [name]`

```
export class Dataparser {
 public constructor() {
   console.log('Hello, world!');
 }
}
```

Import a class

```
import { Component, OnInit, Input } from '@angular/core';
import { Dataparser } from '../dataparser';

@Component({
  selector: 'app-day',
  templateUrl: './day.component.html',
  styleUrls: ['./day.component.css']
})
export class DayComponent implements OnInit {
  @Input() today:string;

  days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];

  constructor() {
   var data = new Dataparser();
  }

  ngOnInit() {
  }
}
```

Import class via file structure

Instantiate it like any other class

Import a library

- Since Angular is in TypeScript, it can use any JavaScript or TypeScript library
- Install as normal with npm: `npm install [packagename]`
  - If you want TypeScript typings, don't forget to install `@types/[packagename]`

Import a library

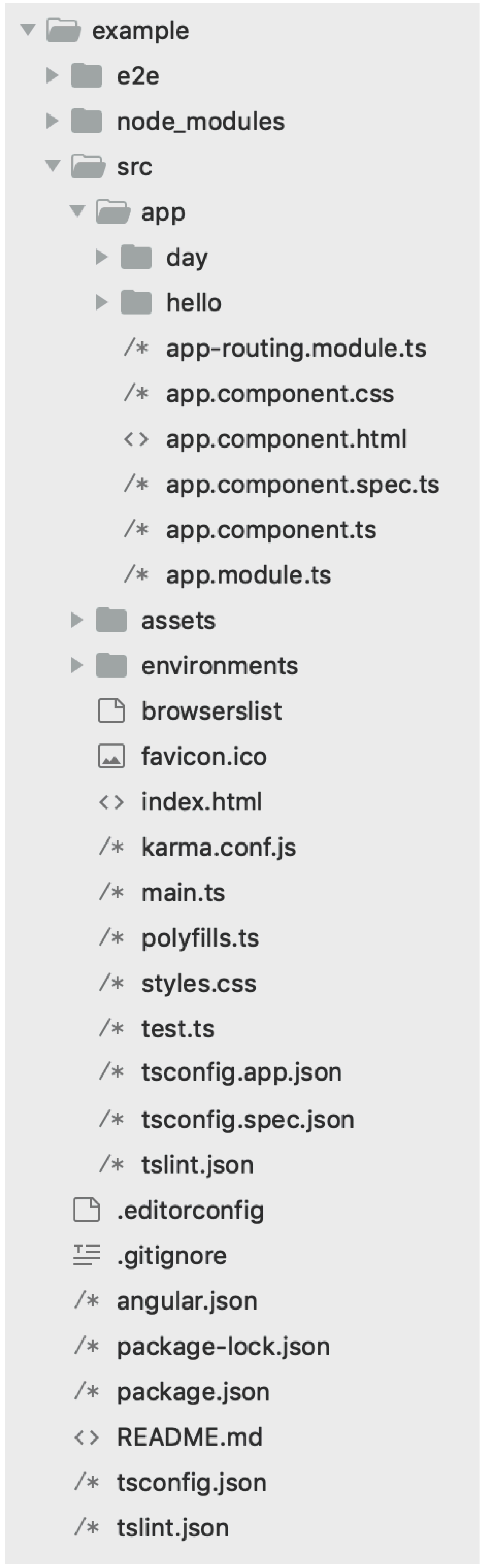
```
import * as chroma from 'chroma-js';
```
Note: different syntax

```
export class Dataparser {

  constructor() {
    console.log(chroma('royalblue')); //'#4169e1'
  }
}
```
Can now be referenced

Angular’s file structure

- Angular projects generate a *lot* of files
    - There are about 75 in the starter code for A3
- Most are boilerplate



```
example
  e2e
  node_modules
  src
    app
      day
      hello
      app-routing.module.ts
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
    assets
    environments
    browserslist
    favicon.ico
    index.html
    karma.conf.js
    main.ts
    polyfills.ts
    styles.css
    test.ts
    tsconfig.app.json
    tsconfig.spec.json
    tslint.json
  .editorconfig
  .gitignore
  angular.json
  package-lock.json
  package.json
  README.md
  tsconfig.json
  tslint.json
```

Socrative Quiz!

Enter your UCI Email when prompted f
name!!!
e.g.,

xxxxx@uci.edu

https://api.socrative.com/rc/52QwBu

Today’s goals

By the end of today, you should be able to…

- Differentiate and explain the roles of Angular components, modules, and services
- Implement a service in Angular
- Navigate Angular’s file structure

Deprecated…

Let’s talk about ngModel

To refresh…

Two-way binding [( )]

- “You often want to both display a data property and update that property when the user makes changes”
- Most common use: binding to user-generated input
- ngModel directive enables two-way binding to input fields

```
<!--enteredText variable contains inputted text-->
<!--textChanged() is called after every keystroke-->
<input [(ngModel)]="enteredText" (change)="textChanged()">
```

https://angular.io/guide/template-syntax

Deprecated…

Let’s talk about ngModel

- ngModel has been deprecated for use in *reactive forms*
- ngModel has not been deprecated as it’s used in assignment 3
- Assignment 3 uses *template-driven* forms
- ngModel was deprecated in reactive forms b/c it behaved differently than the ngModel in template forms…which was confusing.