AXXES
IT CONSULTANCY

ANGULAR TRAINEESHIP

AXXES
IT CONSULTANCY

WHO AM I?

Arno Chauveau

Frontend developer + Buddy

Ghent Ambassador

arno.chauveau@axxes.com

Part I

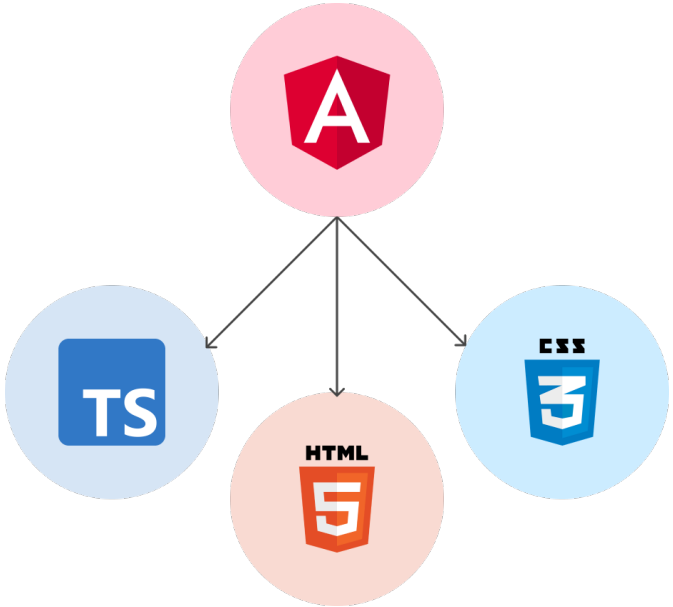
INTRODUCTION

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

WHAT IS ANGULAR?

- Frontend framework
- Opiniated structure & architecture
- Typescript + HTML + CSS/SASS
- Open-source
- Maintained by Google



AXXES
IT CONSULTANCY

ANGULAR 1/2/4/5/6/7/8/9/10 ???

Angular 1.x

=

AngularJS

A.K.A. the old framework
Run away when you see this

Angular > 2.x

=

Just Angular™

A.K.A. the new framework
A happy place for happy people

AXXES
IT CONSULTANCY

IMPORTANT ANGULAR CONCEPTS

- Components
- NgModules
- Client side routing
- Dependency Injection

Part II

Components

The visual building blocks

IT IS ABOUT PEOPLE

JE HEBT 1,37 EURO VINETIQ KREDIET | ALTIJD GRATIS VERZENDING | GRATIS RETOURNEREN

NEDERLANDS

VINETIQ

ZOEK ACCOUNT WINKELWAGEN

WIJN WIJNBOXEN GIFTS WIJNDOMEINEN WIJNWIJZER VALKE VLEUG EVENTS OVER ONS MEMBERSHIP

New Arrivals

Vin de Liège Odyssée 2019

BELGIË|WALLONIË

€14,20 ~~€14,95~~

IN WINKELWAGEN

4.1 Vivino

Vin de Liège Notes Blanches 2019

BELGIË|WALLONIË

€12,30 ~~€12,95~~

IN WINKELWAGEN

Vin de Liège Contre-point 2018

BELGIË|WALLONIË

€16,10 ~~€16,95~~

IN WINKELWAGEN

4.0 Vivino

Vin de Liège Les Éo-lides 2019

BELGIË|WALLONIË

€11,35 ~~€11,95~~

Niet meer in voorraad

Clos des Capucins 'Eminence Grise' 2016

FRANKRIJK|LOIRE|CHINON

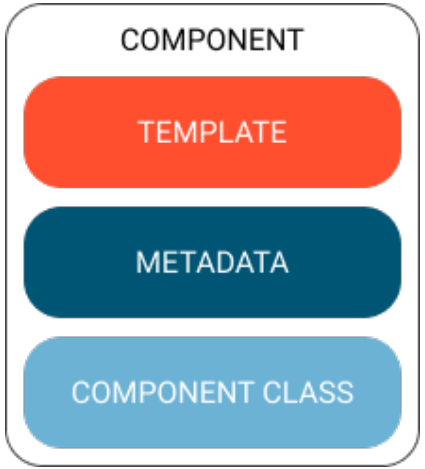
€34,68 ~~€36,50~~

IN WINKELWAGEN

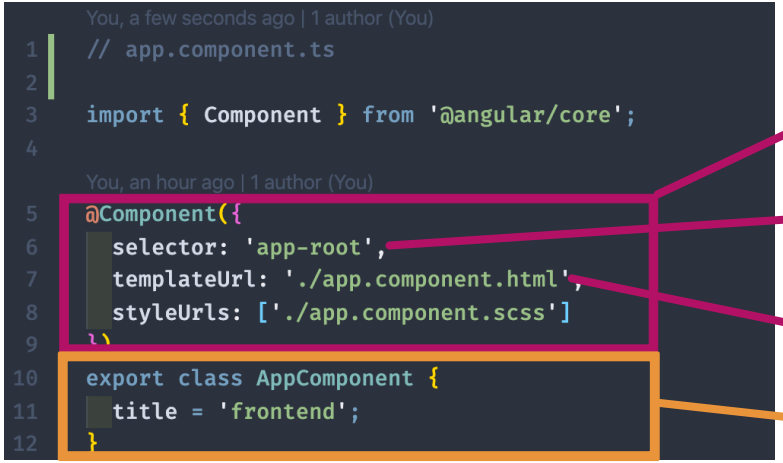
AXXES
IT CONSULTANCY

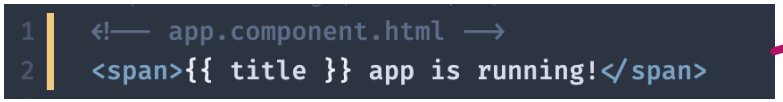
COMPONENTS

- Reusable building blocks of a web page
- Basically a custom DOM element
- Has a template (HTML + special Angular markup)
- Has metadata (using Typescript decorators)
- Has data + logic (Typescript class)
- Note: Additionaly components can contain custom stylesheets



COMPONENT ANATOMY





This annotation specifies that the class is a component class + is used to provide metadata

Can be used in other templates with <app-root></app-root>

Specifies the file for the template

The component class

The template

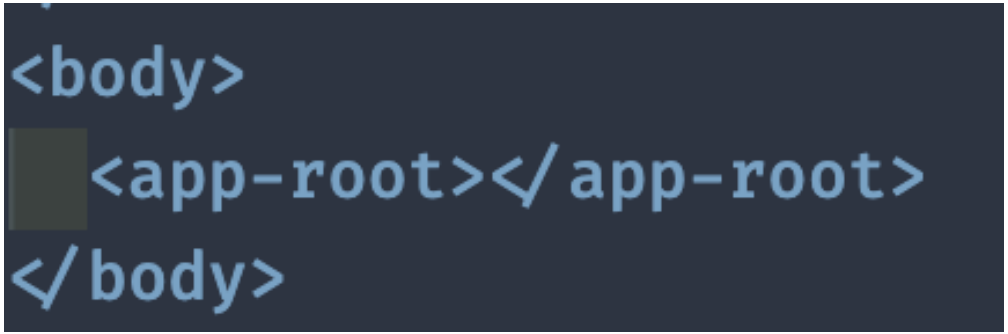
AXXES
IT CONSULTANCY

COMPONENT TEMPLATES

- Used to render content
- All valid HTML is allowed
- Can enrich regular HTML
- Used in component
  - Template: inline
  - TemplateUrl: different file

AXXES
IT CONSULTANCY

USING THE COMPONENT



Part III

NgModules

The backbone of your application

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

NGMODULES

- !== Ecmascript Modules or CommonJS modules
- Provides compilation context + distribution mechanism for components
- Minimum 1 per Angular app (ROOT module)
- Best practice to split up per feature (feature modules)
- Can export functionality
- Can import functionality from other NgModules
- Can be Lazy-loaded
- Put it in its own folder (!)

AXXES
IT CONSULTANCY

MODULE ANATOMY

```
@NgModule({
  declarations: [ /*  ...  */ ],
  imports: [ /*  ...  */ ],
  providers: [ /*  ...  */ ],
  bootstrap: [ /*  ...  */ ],
  exports: [ /*  ...  */ ]
})
export class MyModule { }
```

- **DECLARATIONS:** specifies the components, directives and pipes that are part of the module
- **IMPORTS:** used to import other modules that export functionality eg. BrowserModule
- **PROVIDERS:** used to define injectables that are part of the module (services, guards, interceptors, values, ...)
- **BOOTSTRAP:** used to define the **root component,** should only be used in the root module
- **EXPORTS:** used to define the components that will be used in templates outside of this module

AXXES
IT CONSULTANCY

ROOT MODULE

- Called AppModule -> app.module.ts
- Module that gets bootstrapped when launching app
- Has a root component -> app.component.ts (specified with the bootstrap property)

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

Part IV

GETTING STARTED

Writing our first Angular App

IT IS ABOUT PEOPLE

IT IS ABOUT PEOPLE

REQUIREMENTS

1. NodeJS and NPM (of Yarn)
2. Angular CLI
3. IDE (VS Code, Atom, WebStorm, Sublime)

AXXES
IT CONSULTANCY

SETTING UP A NEW PROJECT

```
# install Angular CLI
npm i -g @angular/cli

# generate new project
ng new <project_name>

# run project
npm start
yarn start
```

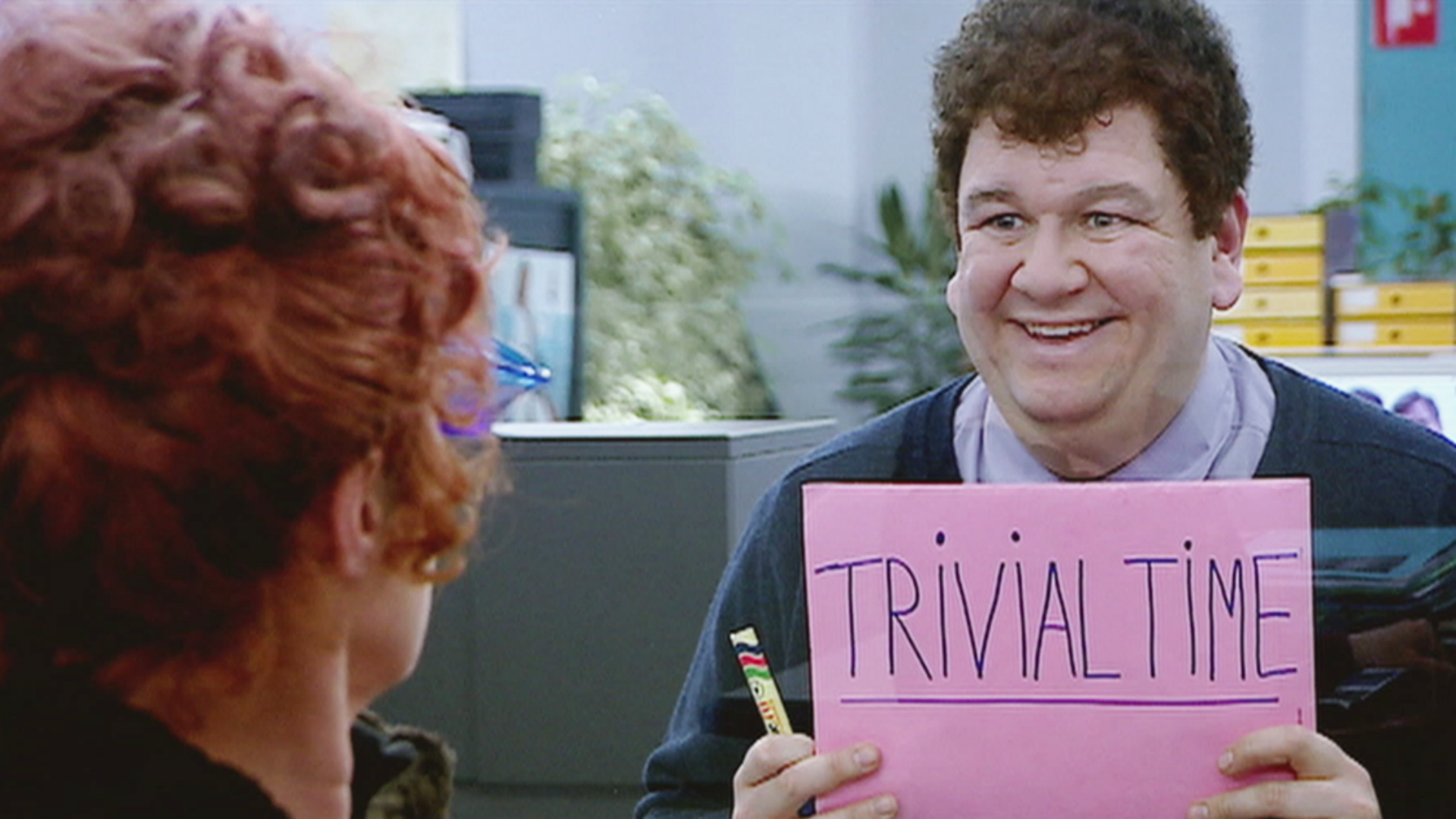
AXXES
IT CONSULTANCY

GENERATING COMPONENTS WITH THE CLI

EXERCISE 1

`git checkout 001`

IT IS ABOUT PEOPLE


TRIVIAL TIME

Part V

INTERPOLATION & EXPRESSIONS

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

INTERPOLATION

- Uses double curly braces {{ ... }}
- Can use (public) variables and methods from component class
- Can also be used in attributes

```
1 <h1>Welcome to {{siteName}}</h1>
2 <img width="300" src="{{image}}" />
```

AXXES
IT CONSULTANCY

EXPRESSIONS

- Also uses double curly braces {{ ... }}
- Most JS expressions are allowed with some exceptions:
  - Assignments (= , +=, -=, ...)
  - Operators like ‘new’, ‘typeof’ and ‘instanceOf’
  - Multiple expressions separated with ‘;’
  - Increment and decrement operators ++ and --

```
1 <p>{{value}} is half of {{ value * 2 }}</p>
```

AXXES
IT CONSULTANCY

SAFE NAVIGATION OPERATOR (?.)

- Prevents referencing null or undefined
- Can be used in interpolation and expressions
- Also available in Typescript > V3.7

```
1 {{personNoCompany ?. company ?. name}}
```

PART VI

PROPERTY BINDINGS (INPUTS)

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

PROPERTY BINDINGS (INPUTS)

- Uses square brackets [...]=""
- Pass data into components = INPUT
- Attribute is a name of a property of component
- Value is an expression -> should never have side effects.
- Angular also exposes this on most existing HTML tags

```
1 <img [src]="image" />
2 <my-component [data]="getData()"></my-component>
3 <my-component [data]="data"></my-component>
4
```

AXXES
IT CONSULTANCY

ONE-TIME STRING EVALUATION

- Square brackets can be omitted when:
  - Property on target component is a string
  - Fixed value
  - Initial value never changes

```
<error-state
    icon="icon-connection-failed"
    [text]="translate('Connection failed')">
</error-state>

```

Part VII

EVENT BINDING (OUTPUTS)

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

EVENT BINDING (OUTPUTS)

- Uses normal brackets (...)
- React to events from elements and components
- Should execute a statement
- Extra data can be passed with $event

```
1 <error-state
2     [text]="translate('Connection failed')"
3     (reload)="onReload()">
4 </error-state>
5 <button (click)="onClick($event)">Very Nice Button</button>
6
```

EXERCISE 2

`git checkout 002`

IT IS ABOUT PEOPLE

PART IIX

HIDING AND SHOWING

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

CONDITIONAL ELEMENTS WITH NGIF

- *NgIf
- * => structural directive => changes the DOM's structure
- Use responsibly: causes DOM redraw!

```
1 <loading-state *NgIf="loadingstate === 'loading'"
2                type="spinner">
3 </loading-state>
4 <data-list *NgIf="loadingstate ==='done'"
5            [data]="data">
6 </data-list>
7
```

Part IX

ITERATING

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

ITERATING

- Define a block of HTML that represents one item from a list
- Uses NgForOf directive
- Don’t forget * -> structural directive
- You can use var inside of your directive’s element
- Uses a microsyntax !== expression

```
1 <ul>
2   <li *ngFor="let todo of todos">{{todo.title}}</li>
3 </ul>
4
```

AXXES
IT CONSULTANCY

INDEX

- Zero based

```
1 <ul>
2   <li *ngFor="let todo of todos; let i = index">{{i}} - {{todo.title}}</li>
3 </ul>
4
```

AXXES
IT CONSULTANCY

TRACKBY

- Bad performance on big lists
- Angular sees list as array of objects
- Will re-render entire list on changes in those objects
- Best practice: always use ‘trackBy’ function!
- Return a unique identifier for the object.
- Angular now sees list as array of trackBy value

```
<ul>
  <li *ngFor="let todo of todos; trackBy:trackById">{{todo.title}}</li>
</ul>

```

```
trackById(todo: Todo): string{
  return todo.id;
}
```

Part X

DYNAMIC CLASSES

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

CLASSNAME BINDING

- Conditional class based on a Boolean expression

```
1 <div class="app_title" [class.app_title--bold]="active">
2     title
3 </div>
```

AXXES
IT CONSULTANCY

NGCLASS

- Multiple classes
- Object as property value
- Key = classname (string)
- Value = Boolean expression

```
1 <div class="app_title"
2   [ngClass]="{'app_title--bold': active, 'app_title--error': hasError}">
3     title
4 </div>
```

Part XI

DYNAMIC STYLES

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

STYLE BINDING

```
1 <div class="app_title"
2   [style.font-weight]="active ? 'bold' : 'normal'">
3     title
4 </div>
```

AXXES
IT CONSULTANCY

NGSTYLE

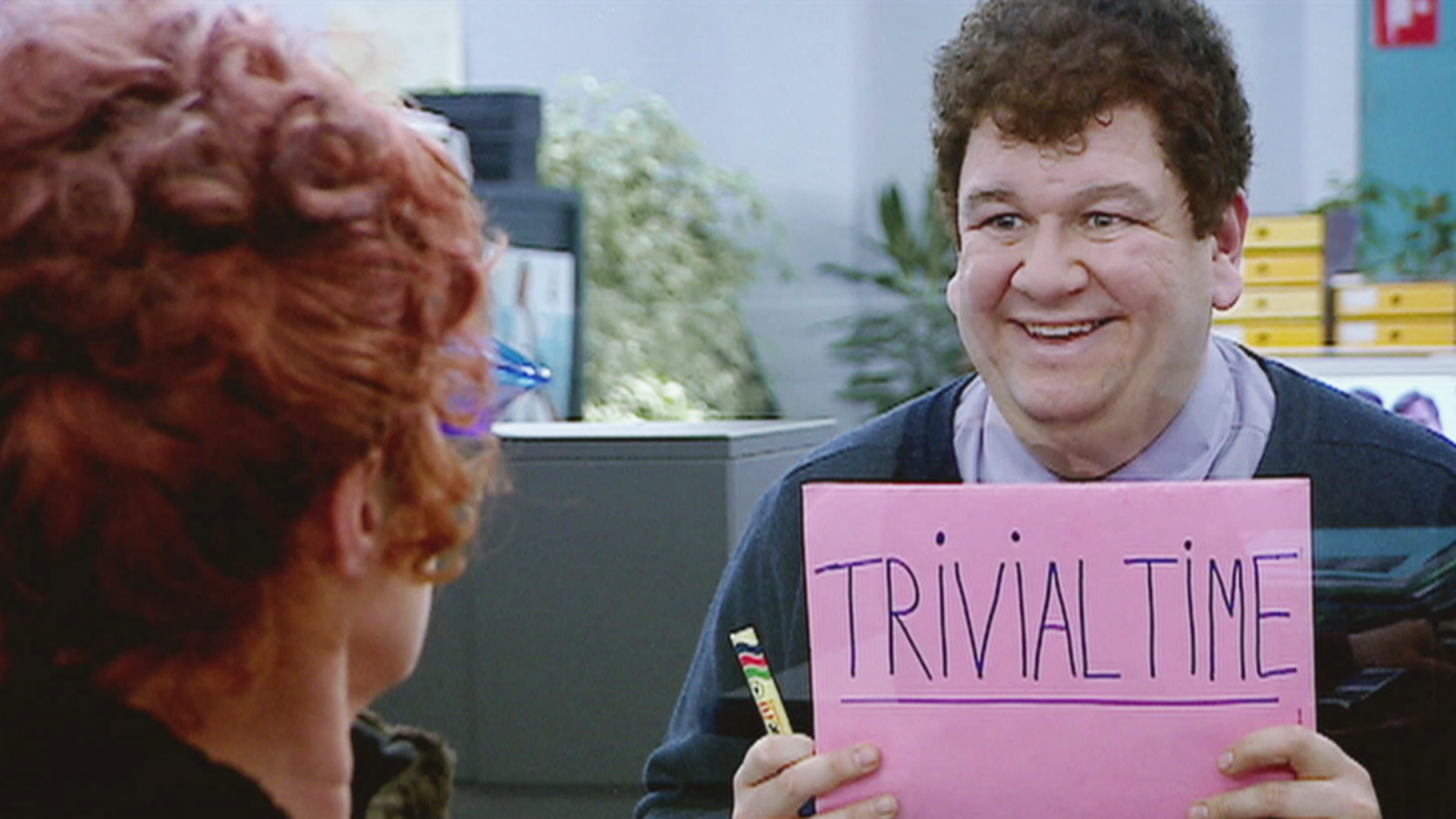
```
1 <div class="app_title"
2   [ngStyle]="{'font-weight': active ? 'bold' : 'normal'}">
3     title
4 </div>
```


You Must Unlearn
What You Have Learned

EXERCISE 3

`git checkout 003`

IT IS ABOUT PEOPLE


TRIVIAL TIME

Part X

PIPES

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

PIPES

- Transforms expression result
- Use for small transforms
- Can have parameters
- Chainable with other pipes

```
1 <p>My birthday is {{ birthday | date:"MM/dd/yy" }}</p>
```

AXXES
IT CONSULTANCY

BUILT-IN PIPES

- Uppercase
- Lowercase
- Date
- Json -> useful for viewing object contents while debugging
- Async -> for observables, auto (un)subscribe
- Many more

AXXES
IT CONSULTANCY

CUSTOM PIPES

- Export class, Pipe suffix, implement PipeTransform interface
- Use @Pipe decorator, add name in config object
- Add a transform function
  - First param = input
  - Rest = Pipe parameters
- **Don’t forget to add to your module declarations!**
- Or `ng g pipe <path>/<pipeName>`

```
@Pipe({name: power})
export class PowerPipe implements PipeTransform{

  transform(value: number, power: number): number{
    return Math.pow(value, power);
  }

}
```

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

```
<div>{{someNumber | power:2}}</div>
```

EXERCISE 4

`git checkout 004`

IT IS ABOUT PEOPLE

Part XIV

COMPONENTS RECAP

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

COMPONENTS RECAP

- Building blocks
- Contains HTML, styling and functionality
- Has a custom HTML tag
- Can use other components
- App is a tree of components

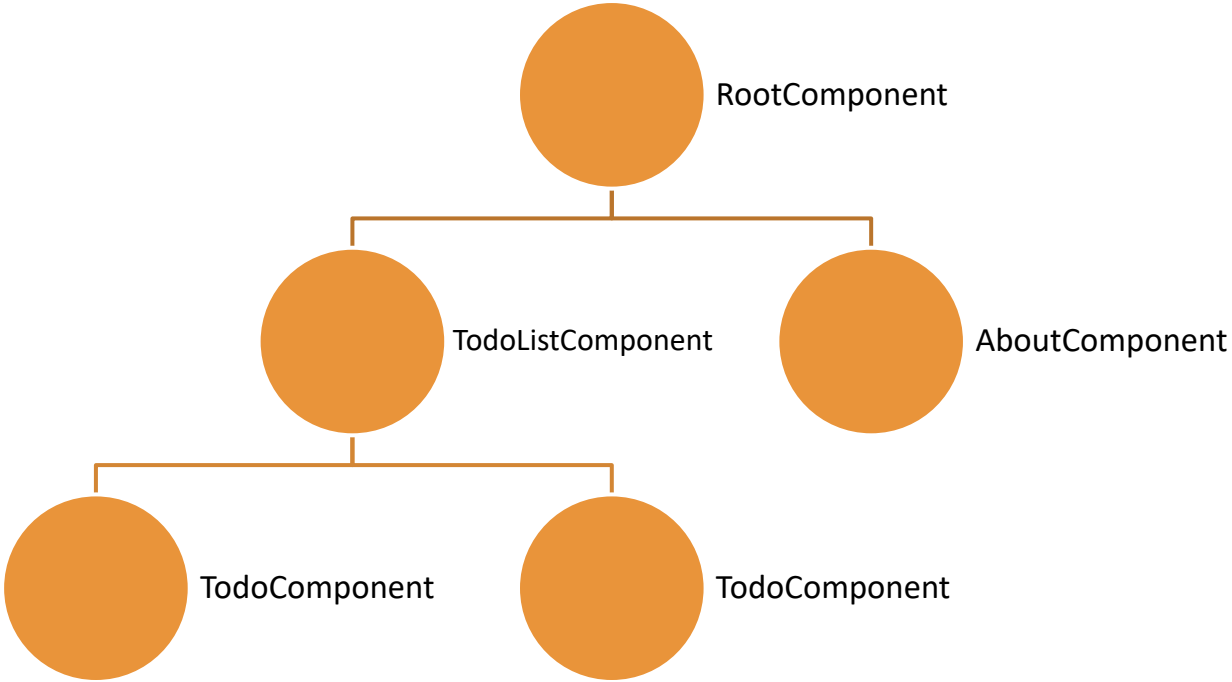
AXXES
IT CONSULTANCY

BENEFITS

- Hierarchical app structure
- Easy to test
- Easy to refactor
- SRP: Single responsibility principle
- Reusable

AXXES
IT CONSULTANCY

COMPONENT TREE EXAMPLE



Part XV

COMPONENT INTERACTION

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

PASS DATA FROM PARENT TO CHILD

- (public) property with @Input decorator on the child component
- Property binding [...]="..." on parent component

```
// child
@Component({
    selector: 'todo',
    template: `<div>{{todo.title}}</div>`
})
export class TodoComponent {
    @Input() todo: Todo;
}

// parent
@Component({
    selector: 'todo-list',
    template: `<todo *ngFor="let todo of todos" [todo]="todo"></todo>`
})
export class TodoListComponent{
    todos: Todo[] = [
      { id: 1, title: 'explain inputs'},
      { id: 2, title: 'explain outputs'}
    ];
}
```

AXXES
IT CONSULTANCY

INTERCEPTING INPUT CHANGES - SETTER

- Add @Input decorator to setter
- Useful to cleanup input values

```
 6 export class TodoComponent {
 7
 8     private _todo: Todo;
 9
10     @Input()
11     set todo (todo: Todo){
12         this._todo = todo.trim();
13     }
14
15     get todo(): Todo{
16         return this._todo;
17     }
18 }
```

AXXES
IT CONSULTANCY

INTERCEPTING INPUT CHANGES – LIFECYCLE HOOK

- Implement OnChanges interface
- ngOnchanges function is triggered when changes are detected
- Param is key/value pair with simpleChange object

```
class SimpleChange {
  constructor(previousValue
  previousValue: any
  currentValue: any
  firstChange: boolean
```

```
 6 export class TodoComponent implements OnChanges{
 7
 8     @Input() todo: Todo;
 9
10     ngOnChanges( changes: SimpleChanges ){
11         if(changes.todo){
12             // do something.
13         }
14     }
15 }
```

AXXES
IT CONSULTANCY

COMMUNICATING FROM CHILD TO PARENT

- Child exposes EventEmitter with @Output decorator
- Parent binds reacting function to the event with (...)="doSomething()"
- Pass data with $event in event binding

```
1  // child
2  @Component({
3      selector: 'todo',
4      template: `
5          <div>
6              {{todo.title}}
7              <button (click)="delete.emit(todo)">delete</button>
8          </div>
9  `
10 })
11 export class TodoComponent{
12     @Input() todo: Todo;
13     @Output() delete = new EventEmitter();
14 }
```

```
//parent
@Component({
    selector: 'todo-list',
    template: `
        <todo
            *ngFor="let todo of todos"
            [todo]="todo"
            (delete)="deleteTodo($event)"
            ></todo>
    `
})
export class TodoListComponent{
    todos: Todo[] = [ ... ]
    deleteTodo(todo: Todo): void{
        //todo: delete todo
    }
}
```

EXERCISE 5

```
git checkout 005
```

IT IS ABOUT PEOPLE

Part XVI

COMPONENT TYPES

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

PRESENTATIONAL COMPONENTS

- How things look
- No dependencies
- Receive data and emit events (input and output) but only with direct parents or descendants
- Stateless
- Easily reusable
- AKA ‘dumb components’ or just ‘components’

AXXES
IT CONSULTANCY

CONTAINER COMPONENTS

- How things work
- Less markup
- Contains data and behavior
- Stateful
- Aka Smart components
- Can navigate between pages and will be navigated to

AXXES
IT CONSULTANCY

READING TIP:

https://blog.strongbrew.io/components-demystified/

Part XVII

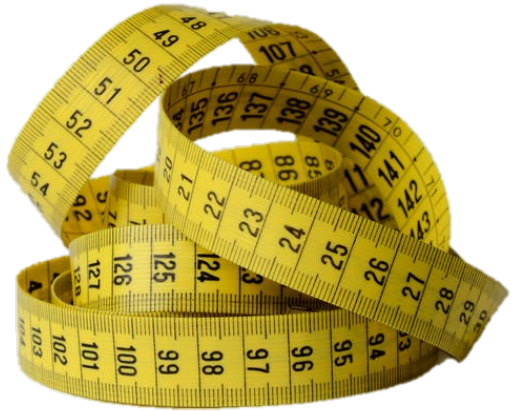
COMPONENT PROTIPS

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

SIZE DOESN'T MATTER

- Except in software development, where smaller is better
- Keep components as small as possible.
- Components should do one thing only!
- Some people put template in component to force this



AXXES IT CONSULTANCY

COMMUNICATION RULES

- Only communicate with direct parents or children!
- No keeping track of who is talking to who: Strict structure
- Also applies to Smart components, however they also have access to an API to communicate with the rest of the application.

AXXES
IT CONSULTANCY

KEEP YOUR COMPONENTS DUMB WHEN POSSIBLE

- Dumb components are much easier to reason about
- Dumb components makes the application less complex, since dumb components do not modify state/data
- It's easier to give dumb components a clear responsibility
- Dumb components are easier to test (less dependencies)

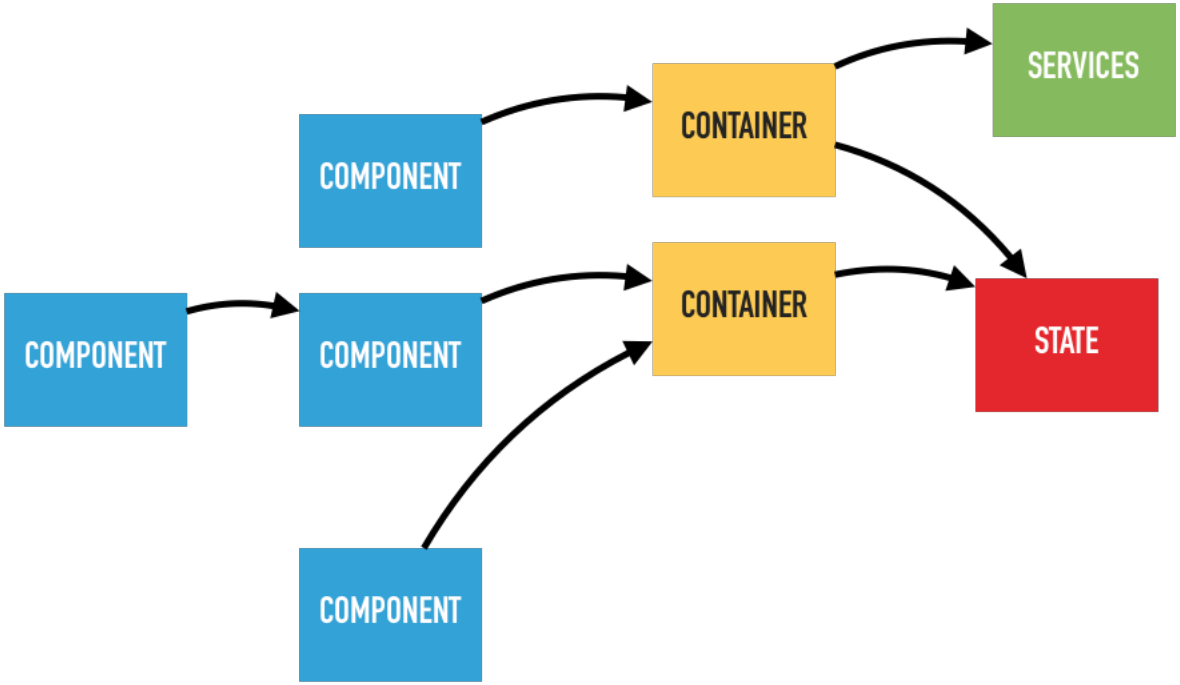
AXXES
IT CONSULTANCY

NAMING CONVENTIONS

- Use separate folders for components and containers
- ExampleContainer -> example.container.ts
- ExampleComponent -> example.component.ts
- You have to edit tslint file:
  - `"component-class-suffix": [true, "Component", ”Container"]`
- Generate with `ng g component <name> --type=“container”`

AXXES
IT CONSULTANCY

OVERVIEW



Part XVIII

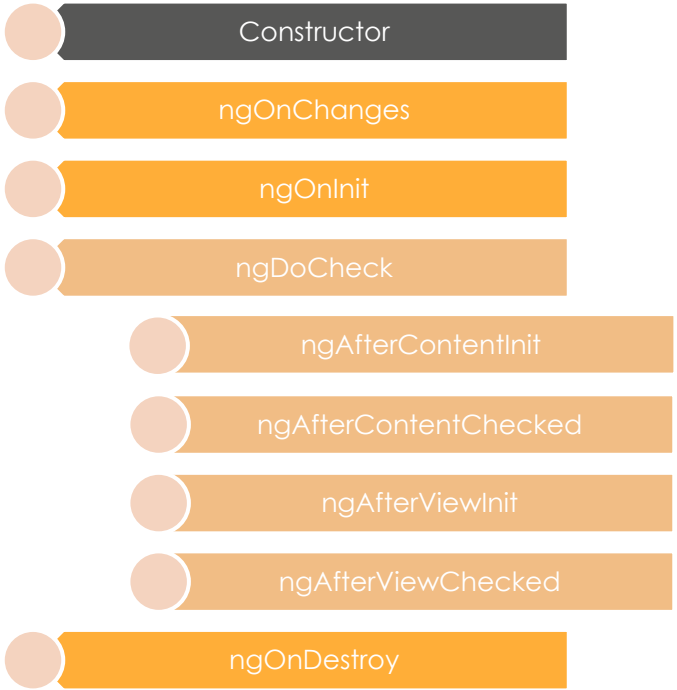
COMPONENT LIFECYCLE HOOKS

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

LIFECYCLE HOOKS

- Each component has a lifecycle
- Hooks enable to subscribe to events from this lifecycle
- Don't forget to implement interface



AXXES
IT CONSULTANCY

NGONCHANGES

- Triggers one time before onInit
- And each time an input changes

AXXES
IT CONSULTANCY

NGONINIT

- Triggered one time, after the first onChanges
- For initialization of component
- VS CTOR:
  - Inputs available
  - Props binding is available
  - Best practice to use this instead of CTOR

AXXES
IT CONSULTANCY

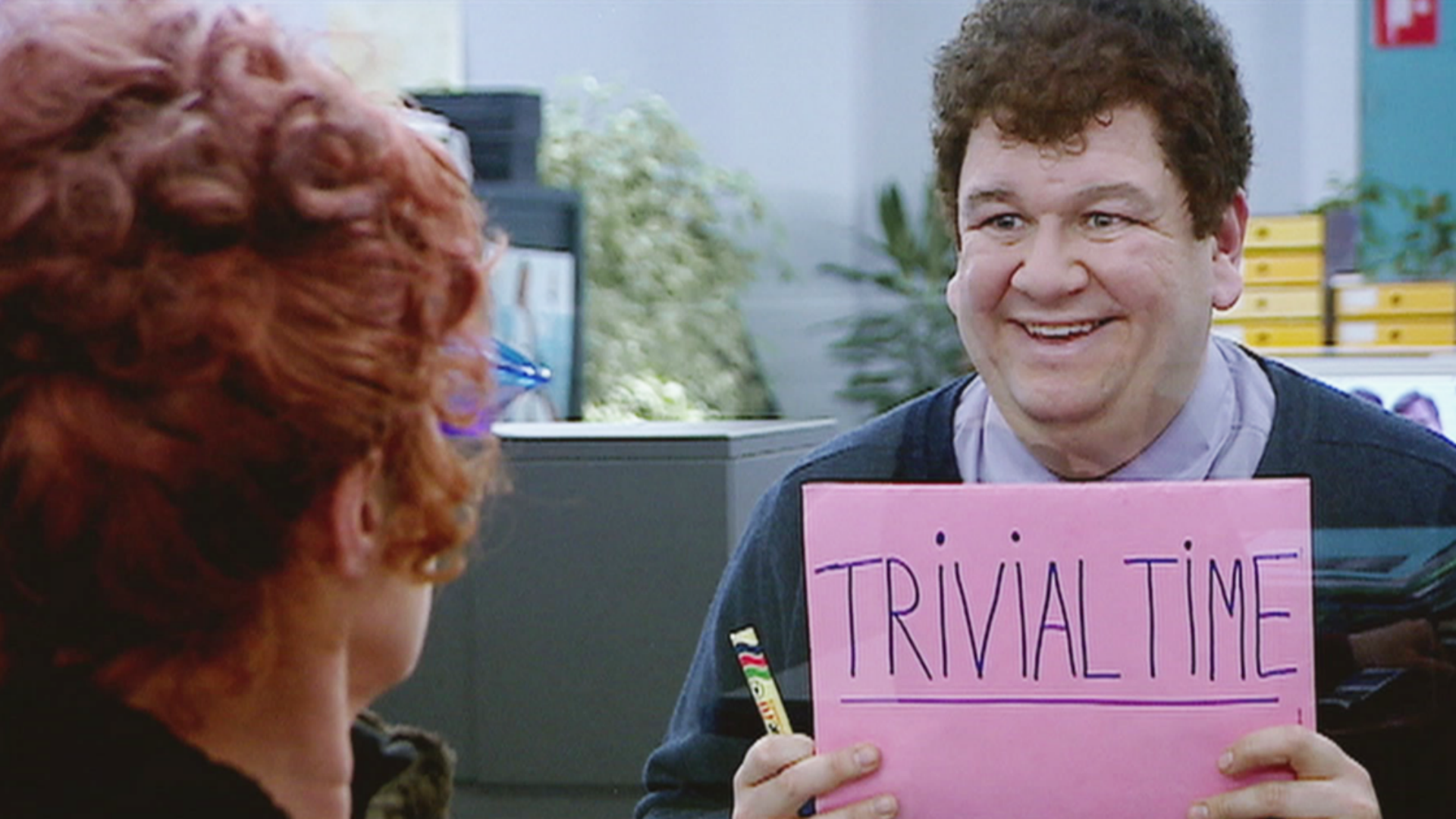
NGONDESTROY

- Triggered just before component is destroyed by Angular
- For component cleanup

EXERCISE 6

```
git checkout 006
```

IT IS ABOUT PEOPLE


TRIVIAL TIME

Part XIX

DEPENDENCY INJECTION

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

INJECTABLE CLASSES

- Normal class
- @Injectable decorator

```
1 @Injectable()
2 export class ExampleService {
3
4 }
```

AXXES
IT CONSULTANCY

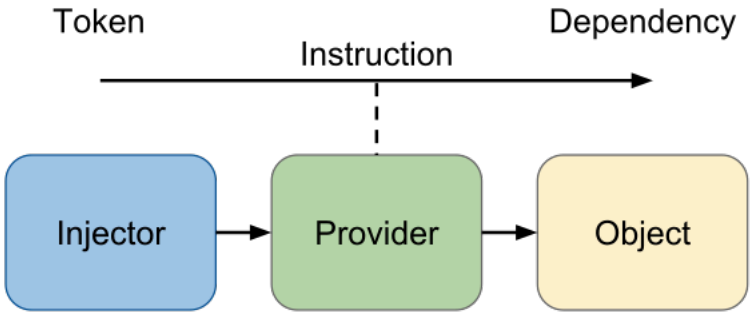
INJECTORS

- Injects injectables in classes
- RootInjector
- Or made on the fly by Angular on module and component scope

AXXES
IT CONSULTANCY

PROVIDERS

- **How** the injector injects injectables
- 3 ways to setup a provider



AXXES
IT CONSULTANCY

PROVIDER SETUP 1/3

- In @Injectable decorator with providedIn
  - Root (best practice!)
  - Module
  - component

```
1 @Injectable({ providedIn: 'root' })
2 export class ExampleService {
3
4 }
```

AXXES
IT CONSULTANCY

PROVIDER SETUP 2-3/3

- In @NgModule or @Component decorator with providers

```
1 @NgModule({
2     providers: [ExampleService]
3 })
4 export class ExampleModule {}
```

AXXES
IT CONSULTANCY

Provider types

```
@NgModule({
    providers: [
      // class provider
      {provide: ExampleService, useClass: ExampleService},

      // shorthand class provider
      ExampleService,

      // value provider
      {provide: 'TEST', useValue: 'test'},

      // factory provider
      {provide: ExampleService, useFactory: () => new ExampleService()}
    ]
})
export class ExampleModule {}
```

AXXES
IT CONSULTANCY

Usage

```
1 @Component( ... )
2 export ExampleComponent {
3     constructor(
4         exampleService: ExampleService
5     ){}
6
7     doSomething():void {
8         this.exampleService.doSomething();
9     }
10 }
```

Part XX

HTTP

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

HTTPCLIENT

- API:
  - get(<url>, <options>)
  - delete(<url>, <options>)
  - post(<url>, <body>, <options>)
  - put(<url>, <body>, <options>)
- Don't forget:
  - Import **HttpClientModule** in module
  - Inject HttpClient via ctor

```
1 @Injectable()
2 export class FooService {
3   constructor(
4     private http: HttpClient
5   ) {}
6
7   getTodo() {
8     this.http.get<Taco>('http://some-url.com/todo')
9       .subscribe(taco => console.log(todo));
10  }
11 }
```

AXXES
IT CONSULTANCY

OBSERVABLES

- **NO SUBSCRIPTION = NO HTTP CALL**
- Catch errors by:
  - pipe() with catchError()
  - Error callback subscription

```
@Injectable()
export class FooService {
  constructor(
    private http: HttpClient
  ) {}

  getTodo() {
    this.http.get<Taco>('http://some-url.com/todo')
        .pipe(
            catchError(err => console.error(err))
        )
        .subscribe(taco => console.log(todo));
  }
}
```

AXXES
IT CONSULTANCY

HTTP INTERCEPTORS

- Middleware for HTTP calls
- Add provider(s) in AppModule:
  - Token: HTTP_INTERCEPTORS
  - useClass: implement HttpInterceptor interfac
  - Multi: true

```
@ngModule({
    providers:[
      {
        provide: HTTP_INTERCEPTORS,
        useClass: logRequestInterceptor,
        multi: true
      }
    ]
})
export class AppModule {}
```

AXXES
IT CONSULTANCY

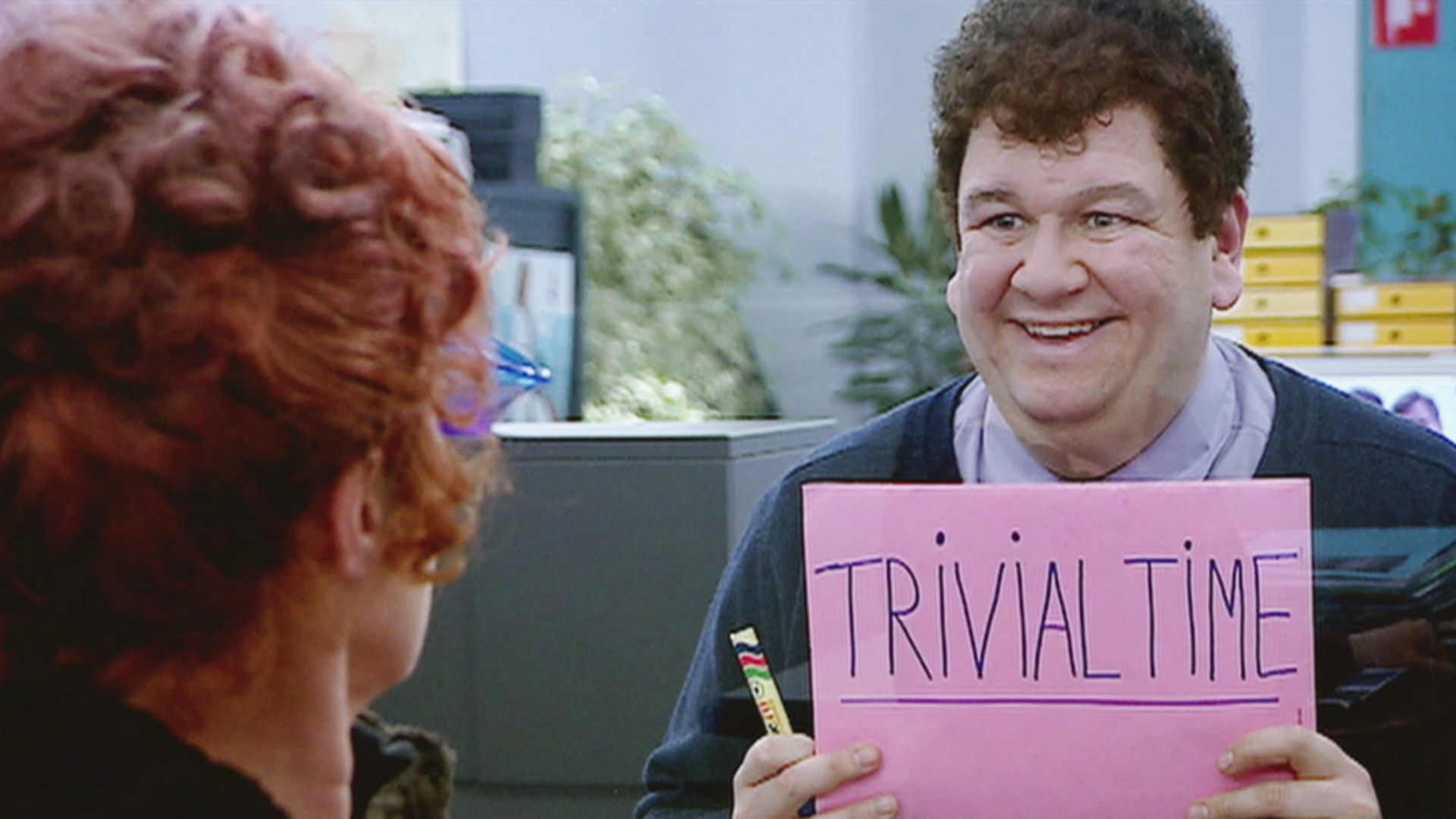
INTERCEPTOR EXAMPLE

```
@Injectable()
export class LogRequestInterceptor implements HttpInterceptor {

  intercept(req: HttpRequest<any>, next: HttpHandler)
    :Observable<HttpEvent<any>> {

    console.log({req});

    // pass request without doing anything
    return next.handle(req);
  }
}
```

EXERCISE 7

`git checkout 007`

IT IS ABOUT PEOPLE


TRIVIAL TIME

Part XXI

ROUTING

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

Setup

- Add <base> in html head of index.html
- Ng new does this for you

```
<head>
  <meta charset="utf-8">
  <title>TraineeshipTest</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
```

AXXES
IT CONSULTANCY

CONFIGURATION 1/2

- Best practice: create routing modules
- Create array of routes:
  - Path (no leading /)
  - Component (container)
- Pass array in import of RouterModule with forRoot(...)
- Also export RouterModule again

```
1 const routes: Routes = [
2   {path: 'hello', component: HelloContainer}
3 ];
4
5 @NgModule({
6   exports: [RouterModule],
7   imports: [RouterModule.forRoot(routes)]
8 })
9 export class AppRoutingModule {
10 }
```

AXXES
IT CONSULTANCY

CONFIGURATION 2/2

- Define where the routing component should appear
- <router-outlet></router-outlet>

```
@Component({
  selector: 'app-root',
  template: `
    <h1>{{title}}</h1>
    <router-outlet></router-outlet>
  `,
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'traineeship';
}
```

AXXES
IT CONSULTANCY

NAVIGATION 1/2

```
1 <a routerLink="/java/dylan">Java Coach</a>
2 <a [routerlink]="['/','dot-net', 'hannes']">.NET Coach</a>
```

AXXES
IT CONSULTANCY

NAVIGATION 2/2

```
@Component( ... )
export class CoachComponent{
    constructor(
        private router: Router
    )

    navigate(cc: string, name: string): void {
        this.router.navigate(['/', cc, name]);
    }

    navigateByUrl(cc: string, name: string): void{
        this.router.navigateByUrl(`/${cc}/${name}`);
    }
}
```

AXXES
IT CONSULTANCY

WILDCARD ROUTES

- Matches every route
- Path='**'
- **ALWAYS LAST**

```
1 const routes: Routes = [
2   {path: 'hello', component: HelloContainer},
3   {path: '**', component: PageNotFoundContainer}
4 ];
```

AXXES
IT CONSULTANCY

REDIRECT ROUTES

- Redirects to another route.
- Useful for default routes
- Don’t forget pathmatch=‘full’

```
1 const routes: Routes = [
2   {path: '', redirectTo: '/hello', pathMatch: 'full'},
3   {path: 'hello', component: HelloContainer},
4   {path: '**', component: PageNotFoundContainer}
5 ];
```

AXXES
IT CONSULTANCY

ROUTE PARAMETERS

- Add a token in route path with :
- Get param via injection of ActivatedRoute
- Subscribe to paramMap

```
const routes: Routes = [
  {path: '', redirectTo: '/consultants', pathMatch: 'full'},
  {path: 'consultants', component: ConsultantListContainer},
  {path: 'consultant/:id', component: ConsultantContainer},
  {path: '**', component: PageNotFoundContainer}
];

@Component({
    template:`
        <h1>id is {{$id | async}}</h1>
    `
})
export class ConsultantContainer implements OnInit{

    $id: Observable<string>;

    constructor(
        private route: ActivatedRoute
    ){}

    ngOnInit(): void{
        $id = this.route.paramMap
            .pipe(
                map(params => params.get('id'))
            )
    }
```

AXXES
IT CONSULTANCY

COMPONENT REUSE

- Param route components are not destroyed when param changes
- Oninit is only called once -> Observable
- Alternative: this.route.snapshot

```
13 export class ConsultantContainer implements OnInit{
14
15     id: string;
16
17     constructor(
18         private route: ActivatedRoute
19     ){}
20
21     ngOnInit(): void{
22         id = this.route.snapshot.paramMap.get('id');
23 }
```

AXXES
IT CONSULTANCY

GUARDS 1/2

- Protect routes
- Injectable class
- Implement interface
  - CanActivate
  - CanActivateChild
  - CanDeactivate
- Ng g guard guards/guardname

```
@Injectable()
export class AuthGuard implements CanActivate {

  constructor(private authService: AuthService) {

  }

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot):
Observable<boolean> | Promise<boolean> | boolean {
    return this.authService.isLoggedIn;
  }

}
```

AXXES
IT CONSULTANCY

GUARD 2/2

- Add to the routes that need to be guarded

```
1 const routes = [{
2   path: 'protected',
3   component: ProtectedContainer,
4   canActivate: [AuthGuard]
5 }];
```

AXXES
IT CONSULTANCY

ROUTES FOR FEATURE MODULES 1/3

- General routes in AppRoutingModule with forRoot(...)
  - Login
  - Default
  - wildcard

```
1 const routes: Routes = [
2   {path: '', redirectTo: '/hello', pathMatch: 'full'},
3   {path: 'hello', component: HelloContainer},
4   {path: '**', component: PageNotFoundContainer}
5 ];
6
7 @NgModule({
8   exports: [RouterModule],
9   imports: [RouterModule.forRoot(routes)]
10 })
11 export class AppRoutingModule {
12 }
13
```

AXXES
IT CONSULTANCY

ROUTES FOR FEATURE MODULES 2/3

- Feature module routes with forChild(...)

```
const routes: Routes = [
  {path: 'consultant', children: [
      {path: 'new', component: NewConsultantContainer},
      {path: ':id', component: ConsultantContainer}
    ]}
];

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

AXXES
IT CONSULTANCY

ROUTES FOR FEATURE MODULES 3/3

- Import in appModule
- Watch out for the order!

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

EXERCISE 8

```
git checkout 008
```

IT IS ABOUT PEOPLE

Part XXII

TEMPLATE DRIVEN FORMS

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

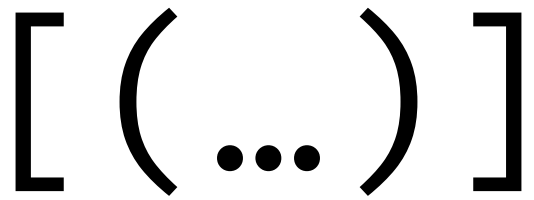
TEMPLATE DRIVEN FORMS

- Forms and validation in template (html)
- Don’t forget to import FormsModule
- AKA plain HTML forms

AXXES
IT CONSULTANCY

BINDING DATA

- ngModel
  - Property + event binding
  - AKA 2 way binding
  - Banana in the box syntax




```
@Component({
  selector: 'app-login',
  template: `
    <form>
      <input type='text'
             name='username'
             [(ngModel)]='credentials.username'
             required>
      <input type='password'
             name='password'
             [(ngModel)]='credentials.password'
             required>
      <button type='submit'>Login</button>
      <button type='button' (click)='reset()'>
        Reset
      </button>
    </form>

    <h1>Form data</h1>
    {{credentials | json}}
  `
})
export class LoginComponent {
  credentials = new Credentials('', '');

  reset() {
    this.credentials = new Credentials('', '');
  }
}
```

AXXES
IT CONSULTANCY

SUBMIT DATA

- Add button with type ‘submit’
- Add event binding to ngSubmit to form

```
<form (ngSubmit)='onSubmit()'>
  …
  <button type='submit'>Login</button>
  …
</form>
```

```
onSubmit() {
  console.log(this.credentials);
}
```

AXXES
IT CONSULTANCY

FORM VALIDATION

- Add HTML validation rules
- Use template var # with ngForm to check status:
  - Valid
  - Invalid
  - Pristine
  - Touched

```
<form (ngSubmit)='onSubmit()' #loginForm='ngForm'>

  <input    type='text'
            name='username'
            [(ngModel)]='credentials.username'
            required>

  <input    type='password'
            name='password'
            [(ngModel)]='credentials.password'
            required>

  <button type='submit' [disabled]='!loginForm.valid'>
    Login
  </button>
  <button type='button' (click)='reset()'>
    Reset
  </button>
</form>
```

AXXES
IT CONSULTANCY

ELEMENT VALIDATION

- Use template var (#) with ngModel to check status:
  - Valid
  - Invalid
  - Pristine
  - Touched

```
<input id='username'
       type='text'
       name='username'
       [(ngModel)]='credentials.username'
       #username='ngModel'
       required>
<span *ngIf='!username.valid'>Mandtory</span>
```

AXXES
IT CONSULTANCY

VALIDATION STYLING

- NgModel automatically adds CSS classes to elements:
  - ng-touched, ng-untouched
  - ng-valid, ng-invalid

```
1 input.ng-invalid {
2   border: 1px solid red;
3 }
```

Part XXIII

REACTIVE FORMS

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

REACTIVE FORMS

- Model driven approach
- Make form in component controller and map to template form
- Don’t forget to import FormsModule and ReactiveFormsModule

AXXES
IT CONSULTANCY

BINDING DATA

- Make a FormControl in component controller
- Use the FormControl directive to assign formcontrol to input

```
@Component({
  selector: 'app-login',
  styleUrls: ['./login.component.scss'],
  template: `
    <form>
      <input id='username'
             type='text'
             [formControl]='username'>

      <button type='submit'>Login</button>
      <button type='button' (click)='reset()'>Reset</button>
    </form>

    <h1>Form data</h1>
    {{username.value}}
  `
})
export class LoginComponent {

  username = new FormControl('');

  reset() {
    this.username.setValue('');
  }

}
```

AXXES
IT CONSULTANCY

GETTING AND SETTING DATA

```
1
2 export class LoginComponent {
3
4   username = new FormControl('');
5
6   reset() {
7     this.username.setValue('');
8   }
9
10  getUserName(): string {
11    return this.username.value;
12  }
13 }
14
```

AXXES
IT CONSULTANCY

FORMGROUPS 1/2

- Collection of FormControls
- Formgroup with ctor object:
  - Key
  - Formcontrol

```
1 export class LoginComponent {
2
3   loginForm = new FormGroup({
4     username: new FormControl(''),
5     password: new FormControl('')
6   });
7
8 }
9
```

AXXES
IT CONSULTANCY

FORMGROUPS 2/2

- In form add formGroup with [formGroup] directive
- Add formControls to input elements with formControlName directive

```
1 <form [formGroup]='loginForm'>
2
3   <input id='username'
4          type='text'
5          formControlName='username'>
6
7   <input type='password'
8          name='password'
9          formControlName='password'>
10
11  <button type='submit'>Login</button>
12 </form>
```

AXXES
IT CONSULTANCY

GETTING AND SETTING DATA

```
1 // get FormControl from FormGroup and set or get:
2 this.loginForm.get('username').setValue('');
3 this.loginForm.get('username').value;
4
5 // set values for the entire FormGroup:
6 this.loginForm.setValue({username: '', password:''});
7
8 // patch values for a part of the FormGroup:
9 this.loginForm.patchValue({password: ''});
```

AXXES
IT CONSULTANCY

VALIDATION

- Validation rules are added to formControl in component controller
- Built in from angular in Validators class
- Check entire formGroup validity

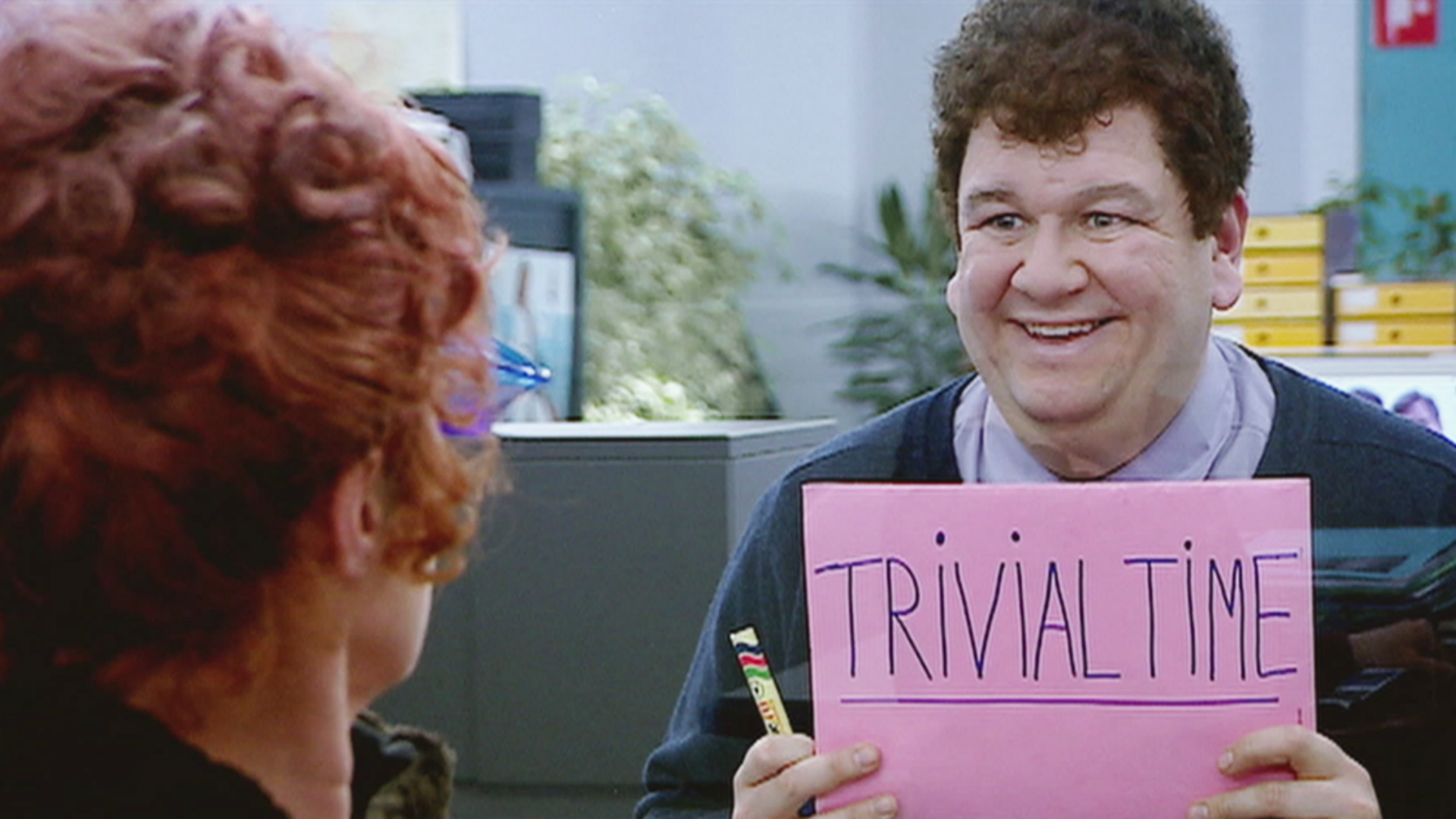
```
<button
    type='submit'
    [disabled]='loginForm.invalid'>
  Login
</button>
```

```
loginForm = new FormGroup({
    username: new FormControl('', Validators.required),
    password: new FormControl('', Validators.required)
  });
```

EXERCISE 9

```
git checkout 009
```

IT IS ABOUT PEOPLE


TRIVIAL TIME

Part XXIV

STATE MANAGEMENT

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

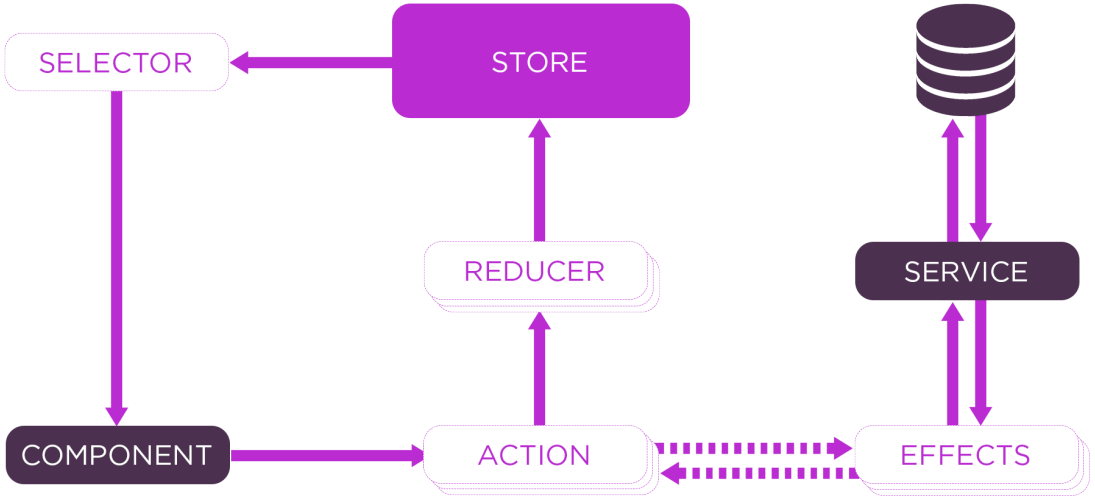
STATE MANAGEMENT

- State types
  - App State
  - Data state
  - Routing state
  - Component state
- Multiple components update state
- Multiple components represent state

AXXES
IT CONSULTANCY

A FIX: REDUX

- Central place for state -> single source of truth (STORE)
- Uni-directional data flow
- COMPONENTS notify store of change (ACTION)
- REDUCER changes STATE based on ACTION
- COMPONENTS react to STATE changes
- ACTIONS can also TRIGGER EFFECTS



AXXES
IT CONSULTANCY

ACTIONS

- Convention: Source of action between []
- Can have properties
- Grouped per module

```
export const homeScore = createAction('[Scoreboard Page] Home Score');
export const awayScore = createAction('[Scoreboard Page] Away Score');
export const resetScore = createAction('[Scoreboard Page] Score Reset');
export const setScores = createAction('[Scoreboard Page] Set Scores', props<{home:number, away:number}>());
```

AXXES
IT CONSULTANCY

REDUCERS

- Pure functions
- INPUT: Current state + action
- OUTPUT: new state
- Initial state set as default param.

```
export const initialState = {
  home: 0,
  away: 0,
};

const scoreboardReducer = createReducer(
  initialState,
  on(homeScore, state ⇒ ({ ...state, home: state.home + 1 })),
  on(awayScore, state ⇒ ({ ...state, away: state.away + 1 })),
  on(resetScore, state ⇒ ({ home: 0, away: 0 })),
);

export function reducer(state , action: Action) {
  return scoreboardReducer(state, action);
}
```

AXXES
IT CONSULTANCY

STATE

- As flat as possible
- Immutable: make it readonly!

```
export interface State {
  home: number;
  away: number;      You, a few seconds ag
}

export const initialState = {
  home: 0,
  away: 0,
};
```

AXXES
IT CONSULTANCY

FRAMEWORKS

AXXES
IT CONSULTANCY

NGRX SETUP

- Yarn add @ngrx/store
- Import StoreModule and setup with reducer

```
import { NgModule } from '@angular/core';
import { StoreModule } from '@ngrx/store';
import * as fromScoreboard from
'./reducers/scoreboard.reducer';

@NgModule({
  imports: [
    StoreModule.forRoot({ game: fromScoreboard.reducer
})
  ],
})
export class AppModule {}
```

AXXES
IT CONSULTANCY

SUBSCRIBE TO STATE CHANGES

- Works with observables
- Use async pipe in template

```
1 export class MyCounterComponent {
2   todos$: Observable<Todo[]>;
3
4   constructor(private store: Store<{todo: TodoState}>) {
5     this.todos$ = store.select(state ⇒ state.todo.data);
6   }
7 }
```

AXXES
IT CONSULTANCY

DISPATCHING ACTIONS

```
export class MyCounterComponent {
  todos$: Observable<Todo[]>;

  constructor(private store: Store<{todo: TodoState}>) {
    this.todos$ = this.store.select(state ⇒ state.todo.data);
  }

  addTodo(todo: Todo){
    this.store.dispatch(new AddTodo(todo));
  }
}
```

AXXES
IT CONSULTANCY

EFFECTS

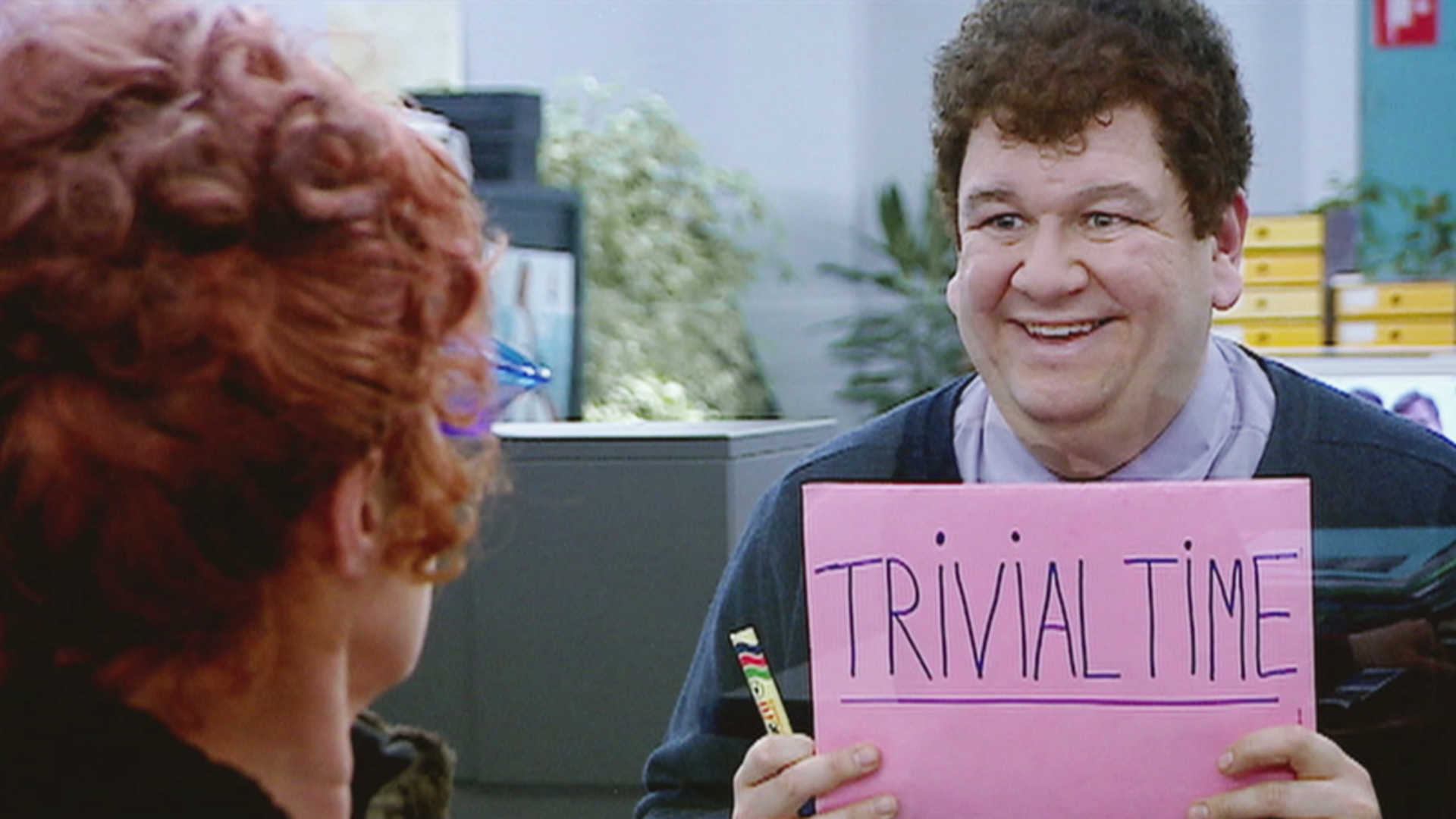
```
@Injectable()
export class MovieEffects {

  loadMovies$ = createEffect(() =>
    this.actions$.pipe(
      ofType('[Movies Page] Load Movies'),
      mergeMap(() => this.moviesService.getAll()
        .pipe(
          map(movies => ({ type: '[Movies API] Movies Loaded Success', payload: movies })),
          catchError(() => of({ type: '[Movies API] Movies Loaded Error' }))
        )
      )
    )
  );

  constructor(
    private actions$: Actions,
    private moviesService: MoviesService
  ) {}
}
```

```
import { EffectsModule } from '@ngrx/effects';
import { MovieEffects } from './effects/movie.effects';

@NgModule({
  imports: [
    EffectsModule.forRoot([MovieEffects])
  ],
})
export class AppModule {}
```

EXERCISE 10

`git checkout 010`

IT IS ABOUT PEOPLE


TRIVIAL TIME