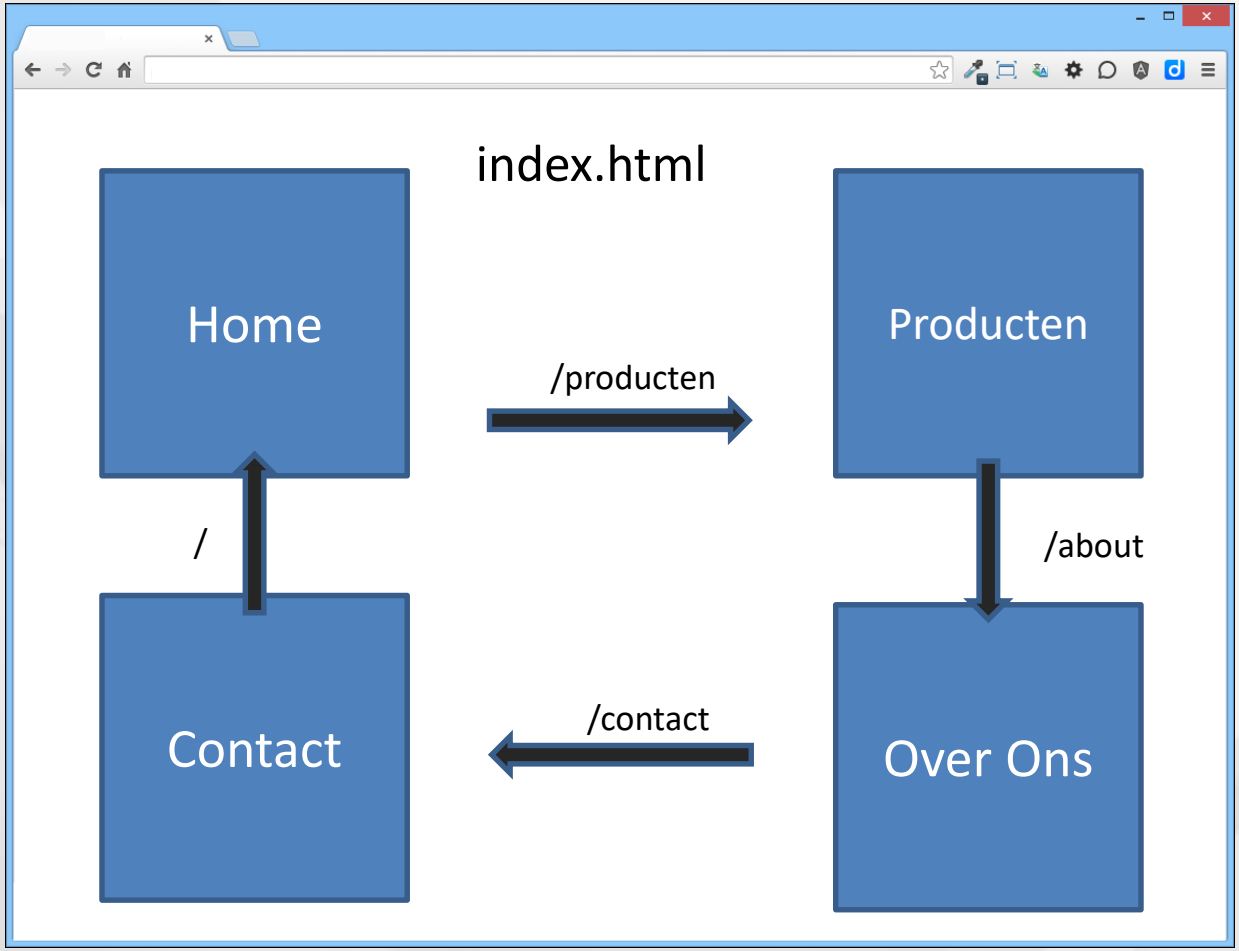
IK WIL

Angular – Module Routing

vijfhart
IT-OPLEIDINGEN

Routing architecture and goal



- Make use of SPA principle
- Making deep links possible

Angular 1: ng-route, of ui-router

```
1. <script src="js/vendor/angular/angular-route.min.js"></script>
```

```
2. <div ng-view></div>
```

```
3. var app = angular.module('myApp', ['ngRoute']);
```

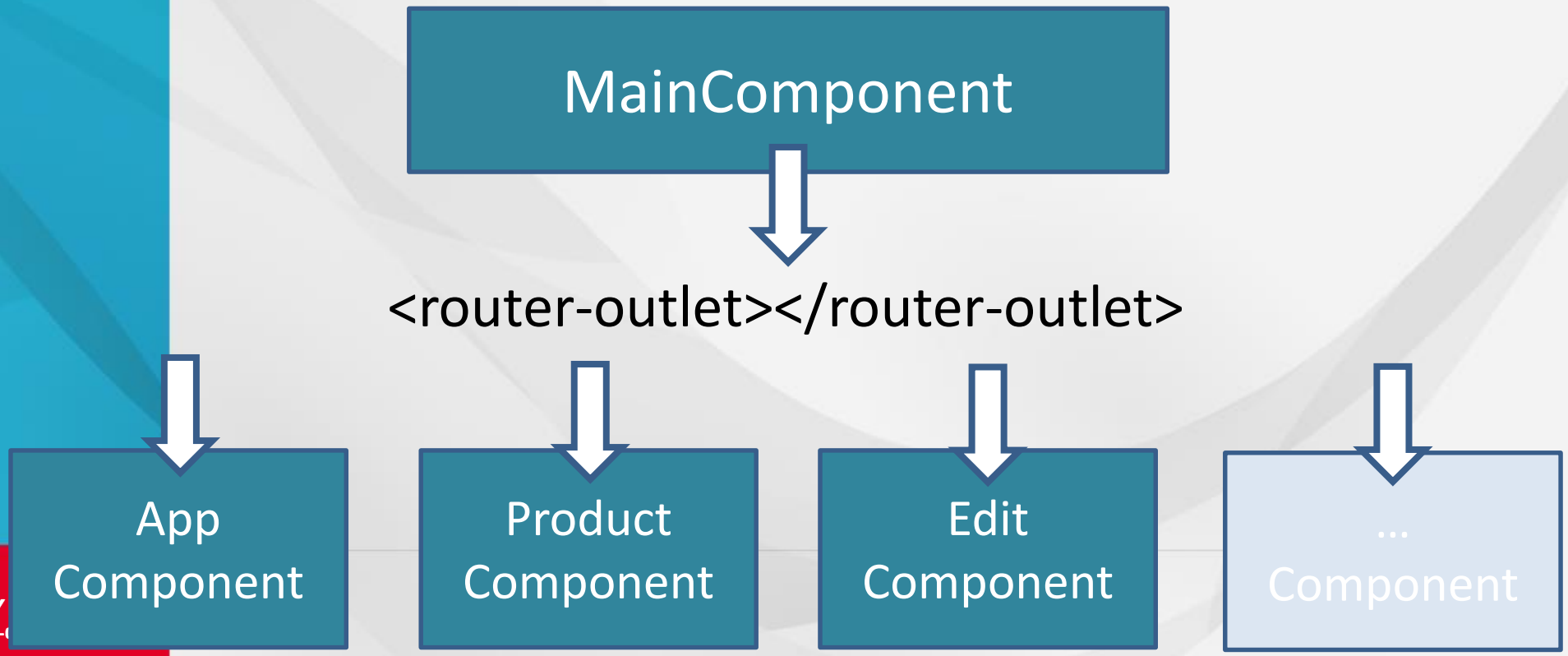
Daarna `$routeProvider` configureren (of `$stateProvider` bij ui-router)

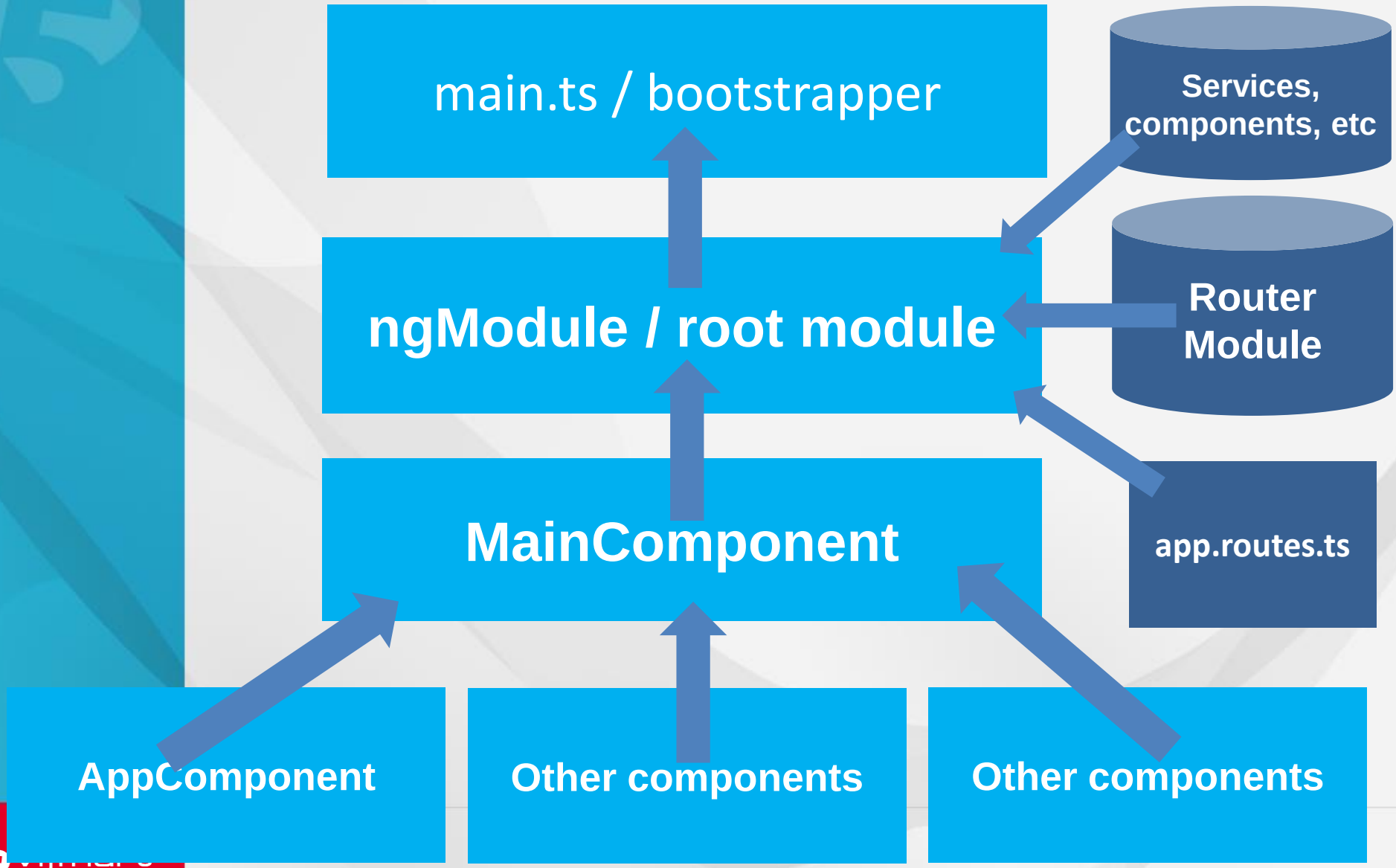
Angular 2: Component Router

- Niet beschikbaar voor AngularJS 1.4+
- Niet beschikbaar: ui-router

Routing – every route is a Component

- MainComponent (or: RootComponent, whatever) with main menu
- Components are injected in `<router-outlet></router-outlet>`





New project in Angular-CLI?

- Default: no routing added
  - `ng new <myProject> --routing`

Stappenplan routing

1. Base Href toevoegen in header van index.html (!)

```
<base href="/">
```

- Er *kunnen* meerdere routes per module zijn. Elke component kan zijn eigen `ChildRoutes` definiëren.
- De Angular-CLI doet dit automatisch voor je.

2. Routes toevoegen

Convention: `app.routes.ts`.

```
// app.routes.ts
import {Routes} from '@angular/router';
import {AppComponent} from "./app.component";
import {CityAddComponent} from "./city.add.component";

export const AppRoutes: Routes = [
   {path: '', component: AppComponent},
   {path: 'home', component: AppComponent},
   {path: 'add', component: CityAddComponent}
];
```

Er zijn meerdere opties en notatiewijzen om routes te declareren

3. Routes beschikbaar maken in Module

- Import `RouterModule` in applicatie
- Import `./app.routes` in applicatie

```
…
// Router
import {RouterModule} from '@angular/router';
import {AppRoutes} from './app.routes';

// Components
import {MainComponent}  from './MainComponent';
…
@NgModule({
   imports      : [
      BrowserModule, HttpModule,
      RouterModule.forRoot(AppRoutes)
   ],
   declarations: [
      MainComponent,
      AppComponent,
      CityAddComponent
   ],
   bootstrap    : [MainComponent]
})
export class AppModule {
}
```

Import Router-onderdelen

Nieuw! MainComponent gaan we nog maken

Configure RouterModule.forRoot()

MainComponent wordt nu gebootstrapt

4. MainComponent met Routing maken

→ Nieuwe component met hoofdmenu en `<router-outlet>`

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

@Component({
   selector: 'main-component',
   template: `
      <h1>Pick your favorite city</h1>
      <!-- Static 'main menu'. Always visible-->
      <!-- Add routerLink directive. Angular replaces this with correct <a href="..."> -->
      <a routerLink="home" class="btn btn-primary">List of cities</a>
      <a routerLink="add" class="btn btn-primary">Add City</a>
      <hr>
      <!-- Dynamically inject views here -->
      <router-outlet></router-outlet>
      <!-- Static footer here. Always visible-->
      `
})
export class MainComponent implements OnInit {
   constructor() {    }
   ngOnInit() { }
}
```

“Hoofdmenu”. Let op routerLink

<router-outlet>

Lege Component

5. Eventueel: index.html aanpassen

- Eventueel selector in index.html aanpassen
- Als `MainComponent` een andere selector heeft

```
<div class="container">
    <main-component>
        Loading...
    </main-component>
</div>
```

6. Nieuwe component(en) maken en importeren

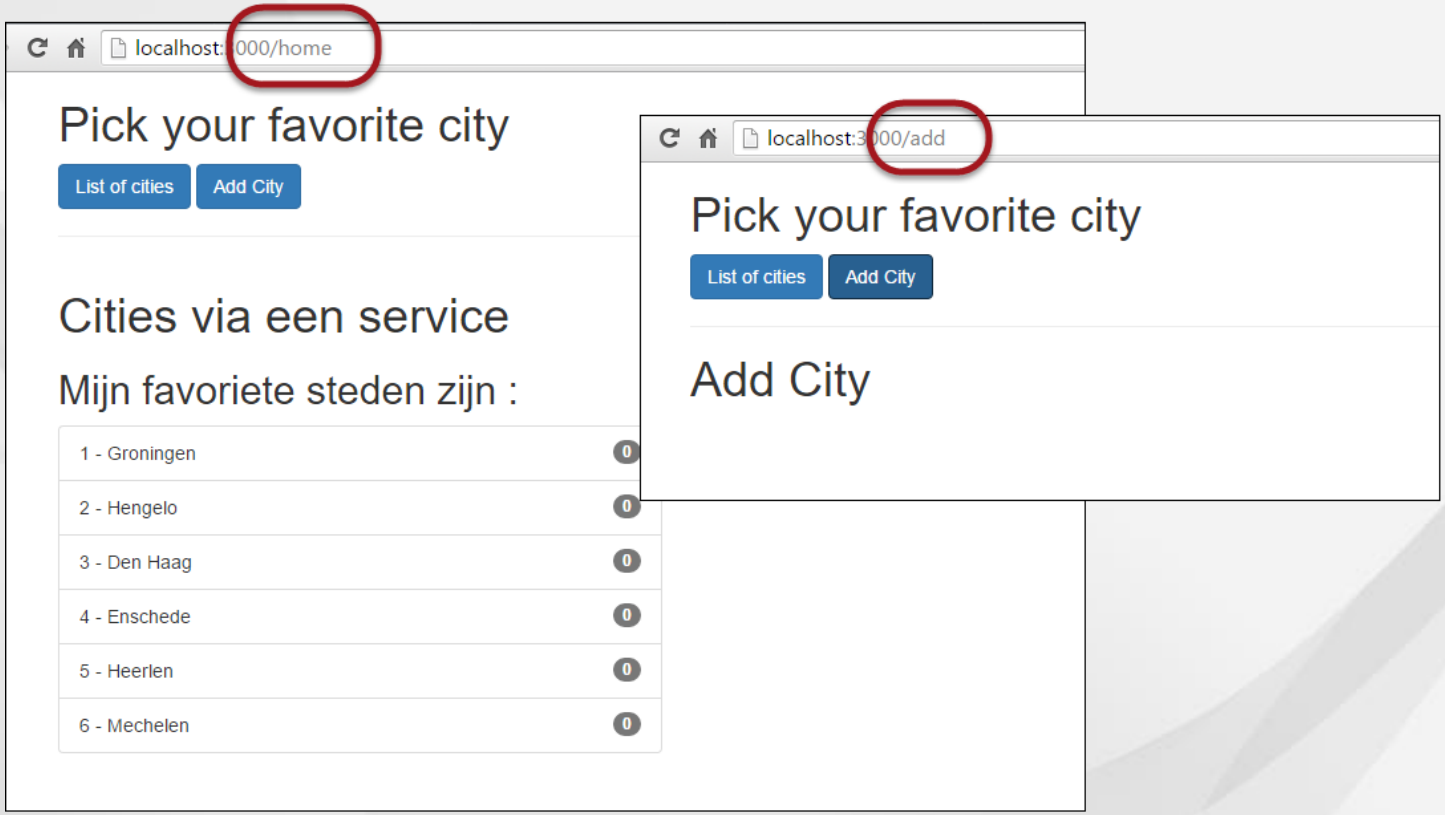
Elke component is een route

```
// city.add.component.ts
import { Component } from '@angular/core';

@Component({
   selector: 'add-c
   template: `<h1>A
})

export class CityAd
   …
}
```

```
// city.edit.component.ts
import { Component } from '@angular/core';

@Component({
   selector: 'edit-city',
   template: `<h1>Edit Cit
})

export class CityEditCompo
   …
}
```

```
// city.detail.component.ts
import { Component } from '@angular/core'

@Component({
   selector: 'detail-city',
   template: `<h1>Detail City</h1> …`
})

export class CityDetailComponent{
   …
}
```

7. Testen



Catch-all routes

```
6   export const AppRoutes: Routes = [
7       {path: '', component: AppComponent},
8       {path: 'home', component: AppComponent},
9       {path: 'add', component: CityAddComponent},
10      {
11          // catch all route
12          path     : '**',
13          redirectTo: 'home'
14      },
15  ];
16
```

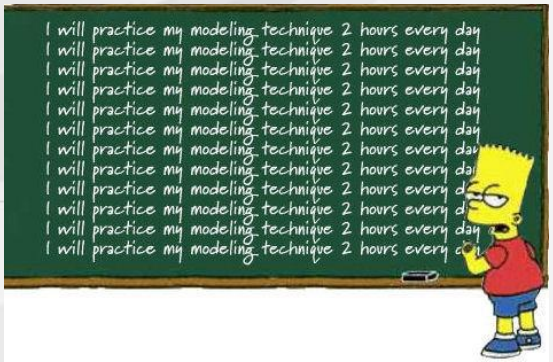
Gebruik `**` voor een catch-all route:

- `Component` opgeven (=route blijft zichtbaar in URL-balk)
- `redirectTo`: opgeven (=nieuwe route staat in URL-balk)

Checkpoint

- Routes worden op module-level ingesteld (Angular 1: app-level).
- Volg het stappenplan. Denk aan injecteren van RouterModule, `app.routes.ts` en `<base href="/">` in de HTML
- Voorbeeld: `400-router`
- Oefening `7a)`. Optioneel: `7b)`
- Officiële documentatie: https://angular.io/guide/router

Oefening….



Routeparameters

Master-Detail views en –applications

Dynamische routes maken

Doel: Enkele detailpagina voor klanten, producten, diensten, etc.

Leesbare routes als: `/cities/5`, of `products/philips/broodrooster`, enzovoort

Werkwijze:

1. Aanpassen `app.routes.ts` en hyperlinks in de pagina.
2. Gebruik `route:ActivatedRoute` in de detail component
3. Schrijf hyperlinks als `<a [routerLink]=...>` met parameter

1. app.routes.ts aanpassen

```
// app.routes.ts
import {Routes} from '@angular/router';
import {AppComponent} from "./app.component";
import {CityAddComponent} from "./city.add.component";
import {CityDetailComponent} from "./city.detail.component";

export const AppRoutes: Routes = [
   {path: '', component: AppComponent},
   {path: 'home', component: AppComponent},
   {path: 'add', component: CityAddComponent},
   {path: 'detail/:id', component: CityDetailComponent}
];
```

2. Detail Component maken

```
// city.detail.component.ts
…
// import {RouteParams} from "@angular/router"; // OLD way
import {ActivatedRoute} from '@angular/router';

@Component({
   selector: 'city-detail',
   template: `<h1>City Detail</h1>
   <h2>Details voor city: {{ id }}</h2>
   `
})

export class CityDetailComponent implements OnInit, OnDestroy {
   id: string;
   currentCity: City;

   constructor(private route: ActivatedRoute) {}

   ngOnInit() {
      this.route.params
         .subscribe((params: any) => {
            this.id = params.id;
         });
   }
}
```

ActivatedRoute

2a. DetailComponent - variants

Using router snapshots (meest eenvoudig)

```
// OR:
// Work via Router-snapshot:
// Sometimes we're not interested in future changes of a route parameter.
// All we need the id and once we have it, we can provide the data we want to provide.
// In this case, an Observable can bit a bit of an overkill.
// A *snapshot* is simply a snapshot representation of the activated route.
this.id = this.route.snapshot.params['id'];
this.name = this.route.snapshot.params['name'];
```

2b. DetailComponent - variants

```
ngOnInit() {
   // NEW:
   this.sub = this.route.params
      .subscribe((params: any) => {
         this.id = params['id'];
         this.name = params['name'];
      });
}

ngOnDestroy() {
   // If subscribed, we must unsubscribe before Angular destroys the component.
   // Failure to do so could create a memory leak.
   this.sub.unsubscribe();
}
```

.unsubscribe()

3. Detail component toevoegen aan Module

```
// app.module.ts
…
// Components
import {CityDetailComponent}  from './city.detail.component';

@NgModule({
   imports      : [
      …
   ],
   declarations: [
      …
      CityDetailComponent
   ],
   providers    : [CityService],
   bootstrap    : [MainComponent]
})
export class AppModule {
}
```

Component

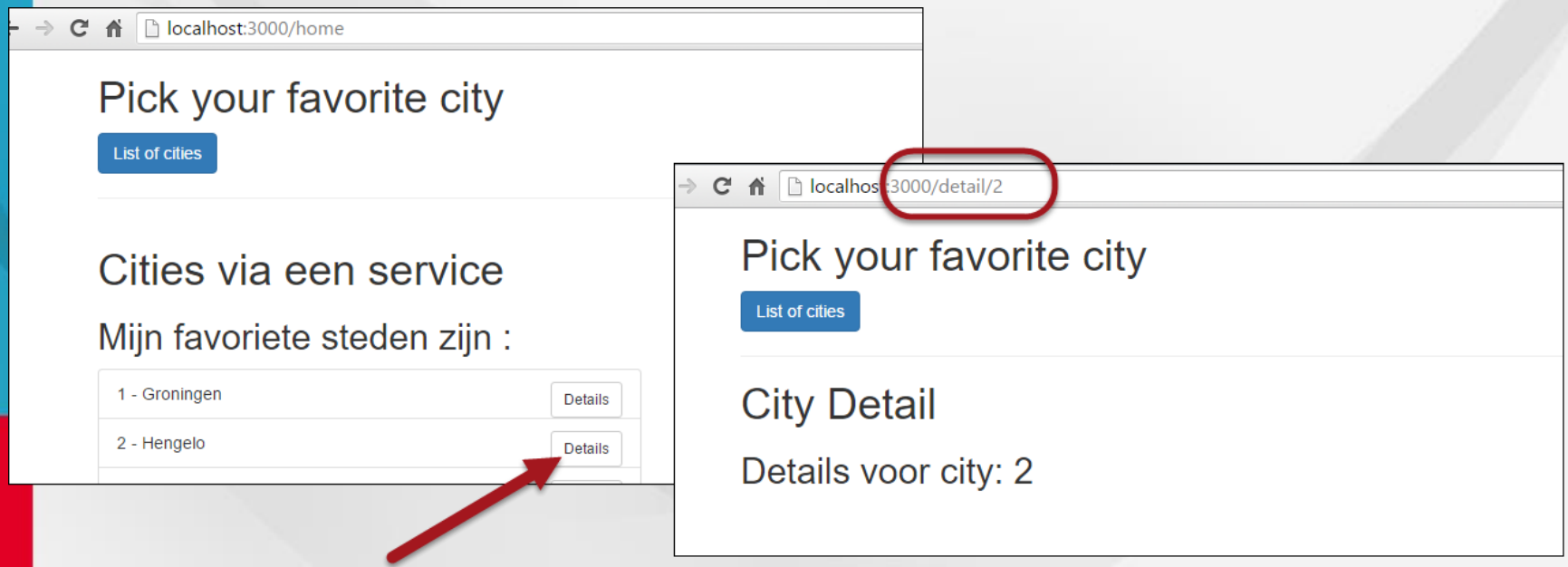
App Component ('Master View') aanpassen

```
<li *ngFor="let city of cities" class="list-group-item">
   <a [routerLink]="['/detail', city.id]">
      {{ city.id}} - {{ city.name }}
   </a>
</li>
```

Let er op dat `[routerLink]` nu dynamisch moet worden gevuld en dus binnen `[…]` moet staan voor attribute binding

Meegeven van parameters

- Let op meegeven van *array van parameters* aan `[routerLink]`
- Parameters worden gematched op positie. Niet op naam.
- Optioneel : service uitbreiden om specifiek product/item te retourneren



Vervolg, bijv. details via service: city.service.ts:

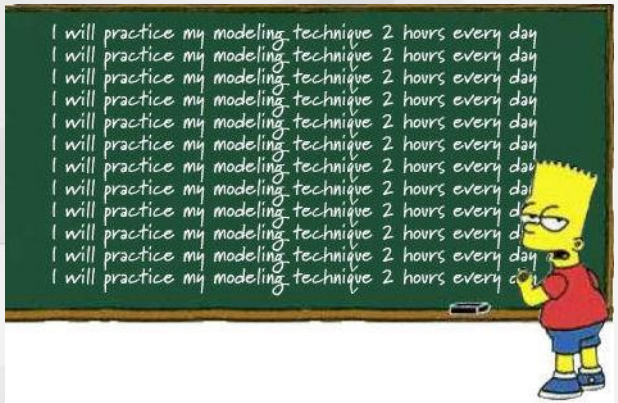
→ Bijvoorbeeld (kan beter, maar het werkt wel):

```
// retourneer een city, op basis van ID
getCity(id: string): City[] {
   return this._http.get('app/cities.json')
      .map(cities =>cities.json())
      .map(cities => cities.find((city: City) => {
         return city.id === parseInt(id);
      }))
}
```

Checkpoint

- RouteParameters worden met `:parameterName` ingesteld in `app.routes.ts`.
- Denk aan injection van `ActivatedRoute` in de component.
- Hierin is een property `.params` aanwezig met de meegegeven parameters.
- Voorbeeld: `\401-route-parameter`
- Oefening `7c)`

Oefening….



Meer over routing

- Router Guards – delen van je routes beveiligen
- Child Routes
- Named Router Outlets
  - http://onehungrymind.com/named-router-outlets-in-angular-2/
- Router resolvers
  - https://blog.thoughtram.io/angular/2016/10/10/resolving-route-data-in-angular-2.html
- Lazy Loading – Applicatie opdelen in Modules en laden *on demand*
  - https://angular.io/guide/router#lazy-loading-route-configuration
- Optionele parameters – [QueryParams]
  - *https://angular-2-training-book.rangle.io/handout/routing/query_params.html*

Bonus: Sheets over Route Guards

Delen van de applicatie beveiligen met Guards

Guard Types

- Four types of guards:
  - `CanActivate` – decides if a route can be activated
  - `CanActivateChild` – decides if children of a route can be activated
  - `CanDeactivate` – decides if a route can be deactivated
  - `CanLoad` – decides if a module can be loaded lazily

http://blog.thoughtram.io/angular/2016/07/18/guards-in-angular-2.html

Defining Guards

- Multiple ways (as functions or as classes)
- Regardless, it needs to return a
  - `Observable<boolean>`,
  - `Promise<boolean>` or
  - `boolean`.
- Defined in `@NgModule`, or as a separate class

1. .Guards as a function

→ Define a token and a guard function. For example in `app.module.ts`.

```
// app.module.ts
…

@NgModule({
   …
   providers   : [
      CityService,
      {
         provide : 'CanAlwaysActivateGuard',
         useValue: () => {
            console.log("Route requested");
            return true; // do validation or other stuff here
         }
      }
   ],
   …
})
export class AppModule {}
```

Token

Function

Use the guard token in app.routes

```
// app.routes.ts
…
export const AppRoutes: Routes = [
   …
   {
      path: 'home',
      component: AppComponent,
      canActivate: ['CanAlwaysActivateGuard'] // Defined in app.module.ts
   },
   …
];
```

(re)use of string token

You *can* have multiple tokens/functions, guarding your route

Guards as a class

- Used: when the guard needs Dependency Injection
- Common use: with some kind of Authentication Service.

- All about Implementing interfaces!
  - `canActivate()`
  - `canActivateChild()`
  - `canDeactivate()`

canActivateViaAuthGuard.ts

```
// canActivateViaAuthGuard.ts
import { Injectable } from '@angular/core';
import { CanActivate } from '@angular/router';
import { AuthService } from './auth.service';

@Injectable()
export class CanActivateViaAuthGuard implements CanActivate {

   constructor(private authService: AuthService) {}

   canActivate() {
      return this.authService.isLoggedIn();
   }
}
```

Class/Guard name

Auth Service

Interface implementation

Register Guard class on module and routes

```
// app.module.ts
…
@NgModule({
   …
   providers    : [
      …,
      AuthService,
      CanActivateViaAuthGuard
   ],
   …
})
export class AppModule
}
```

```
// app.routes.ts
…
import {CanActivateViaAuthGuard} from "./canActivateViaAuthGuard";

export const AppRoutes: Routes = [
   …
   {
      path        : 'add',
      component   : CityAddComponent,
      canActivate: [CanActivateViaAuthGuard]
   },
   …
];
```

Deactivating routes

- Called when navigating *away* from a route
- Same approach as `CanActivate` route

```
// canDeactivateGuard.ts
import {Injectable} from '@angular/core';
import {CanDeactivate} from '@angular/router';
import {CanDeactivateComponent} from "./canDeactivate.component";

@Injectable()
export class CanDeactivateGuard implements CanDeactivate<CanDeactivateComponent> {

   canDeactivate(target:CanDeactivateComponent) {
      // Can the user deactivate the route? Test for changes here!
      // For now, return Yes|Nope from the browser confirm dialog.
      if (target.hasChanges()) {
         return window.confirm('Do you really want to cancel? There might be unsaved changes.');
      }
      return true;
   }
}
```

Add guard to routes

```
// app.routes.ts
…
import {CanDeactivateComponent} from "./canDeactivate.component";
import {CanDeactivateGuard} from "./canDeactivateGuard";

export const AppRoutes: Routes = [
   …
   {
      path          : 'deactivate',
      component     : CanDeactivateComponent,
      canDeactivate: [CanDeactivateGuard]
   },
   ….
];
```

Create DeactivateComponent

→ Add implementation of `.hasChanges()`!

```
// …
export class CanDeactivateComponent implements OnInit {
   // Properties voor de component/class
   myForm:FormGroup = new FormGroup({
      txtInput:new FormControl()
   });

   constructor(private route: Router) { }
   ngOnInit() {}

   moveAway() {
      this.route.navigate(['/home']);
   }
   hasChanges(){
      return this.myForm.dirty; // return state of the form
   }
}
```

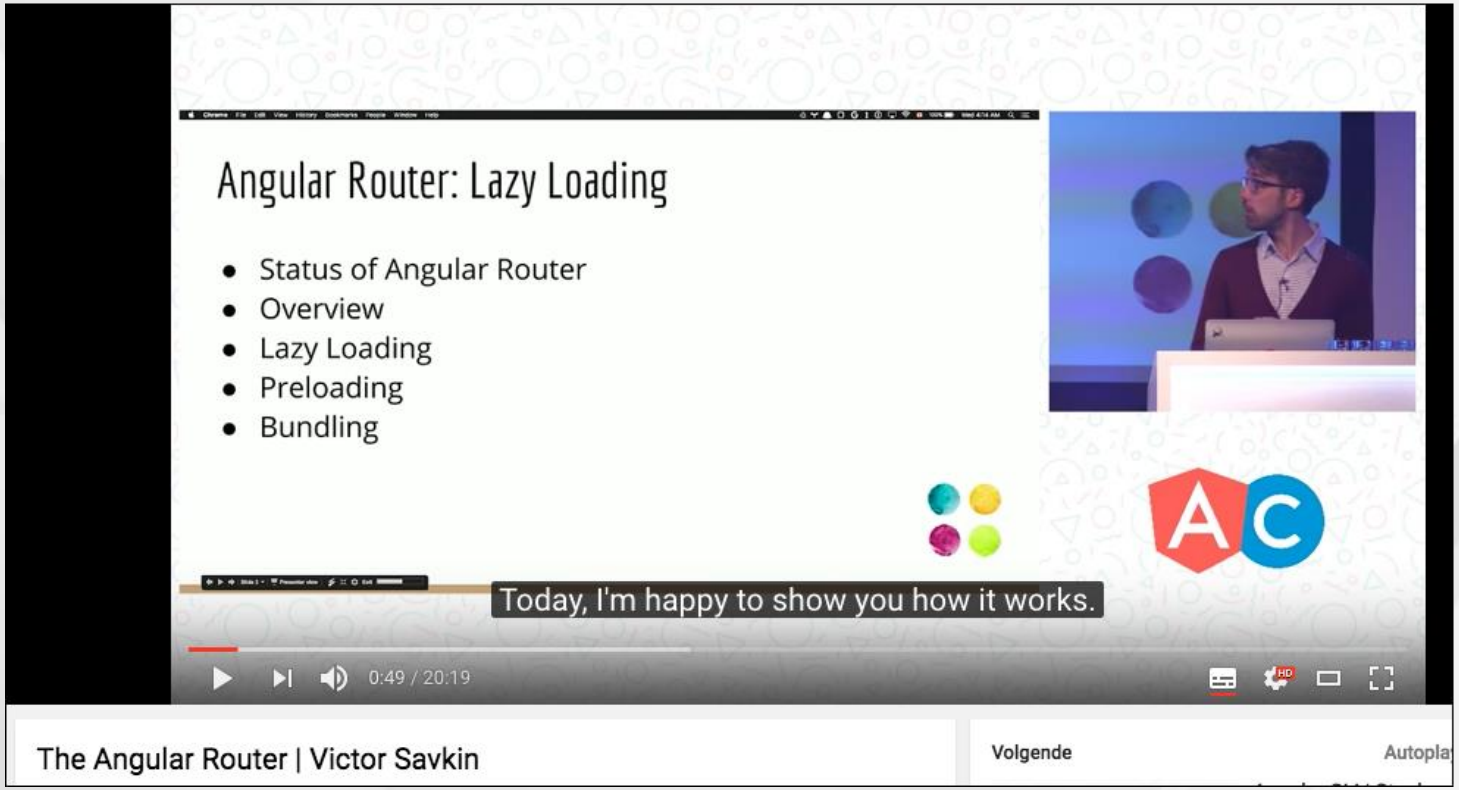
Meer over routing

- https://angular.io/docs/ts/latest/guide/router.html
- http://blog.thoughtram.io/angular/2016/06/14/routing-in-angular-2-revisited.html
- http://blog.thoughtram.io/angular/2016/07/18/guards-in-angular-2.html
- https://vsavkin.com/

Victor Savkin (=maker van de router)

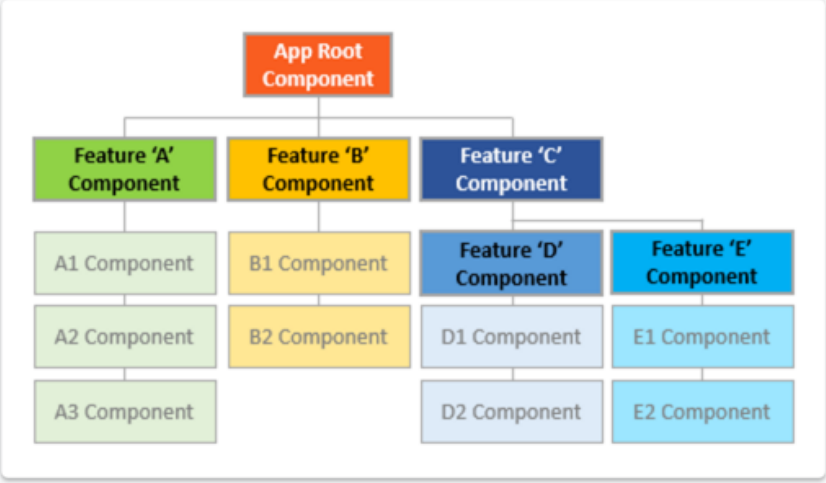
https://leanpub.com/router

https://www.youtube.com/watch?v=QLns6s02O48



Advanced routing

It's looking good as a general pattern for Angular applications.



- each feature area in its own module folder
- each area with its own root component
- each area root component with its own router-outlet and child routes
- area routes rarely (if ever) cross

https://angular.io/docs/ts/latest/guide/router.html

Victor Savkin on Routing

https://vsavkin.com/