

Angular 2

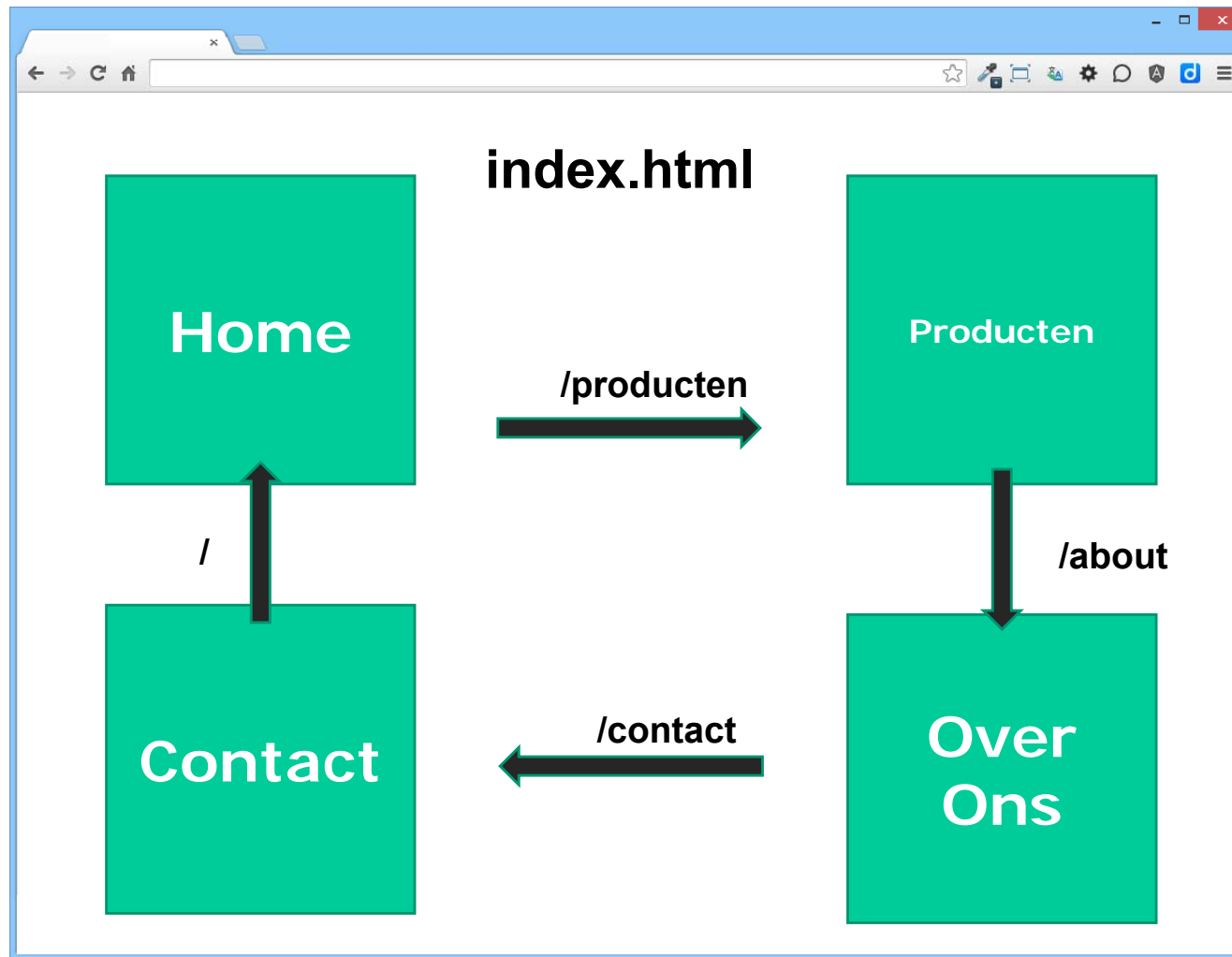
Module 5 - Routing



Peter Kassenaar –
info@kassenaar.com



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)

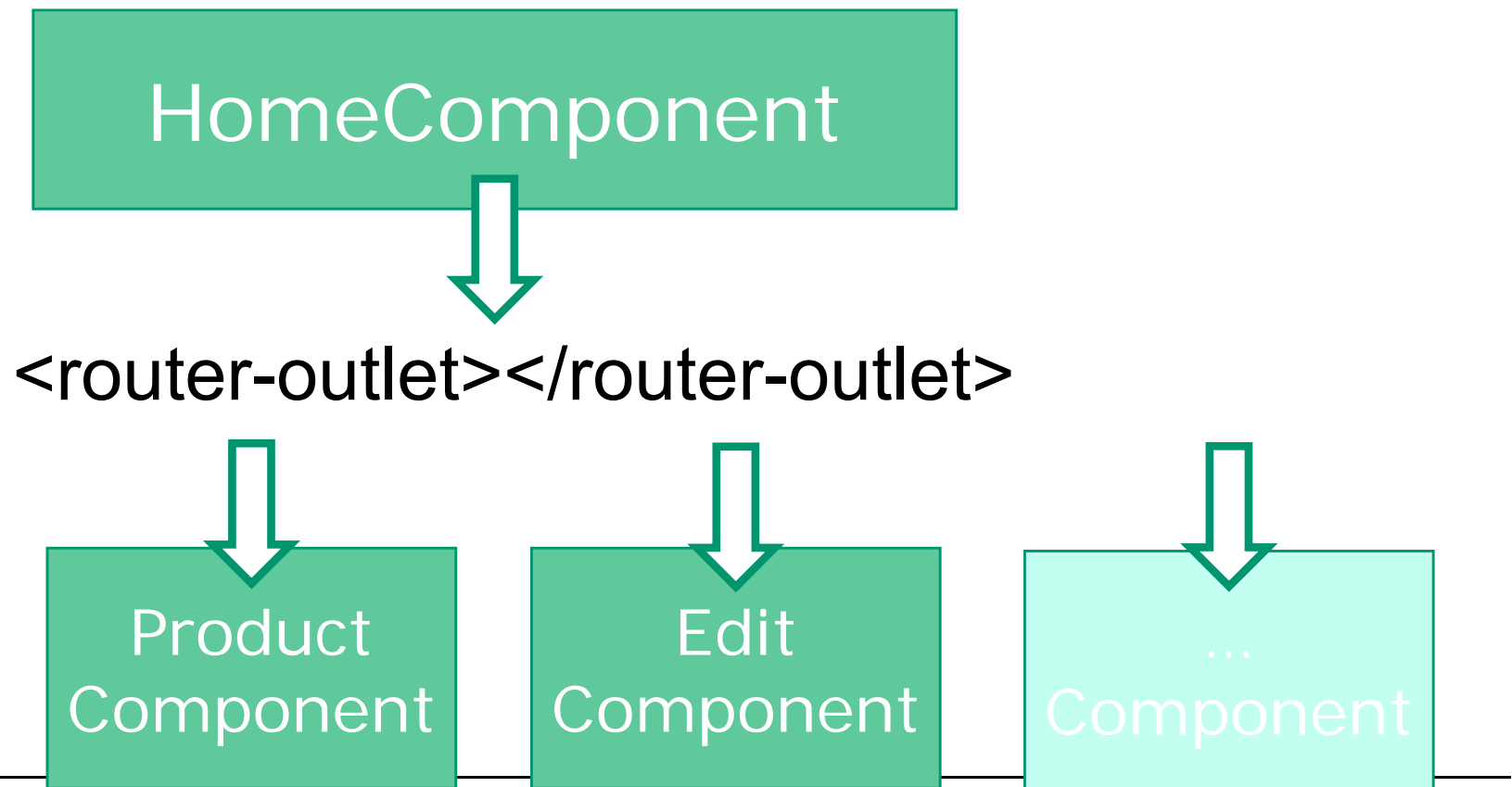
Angular 2: Component Router

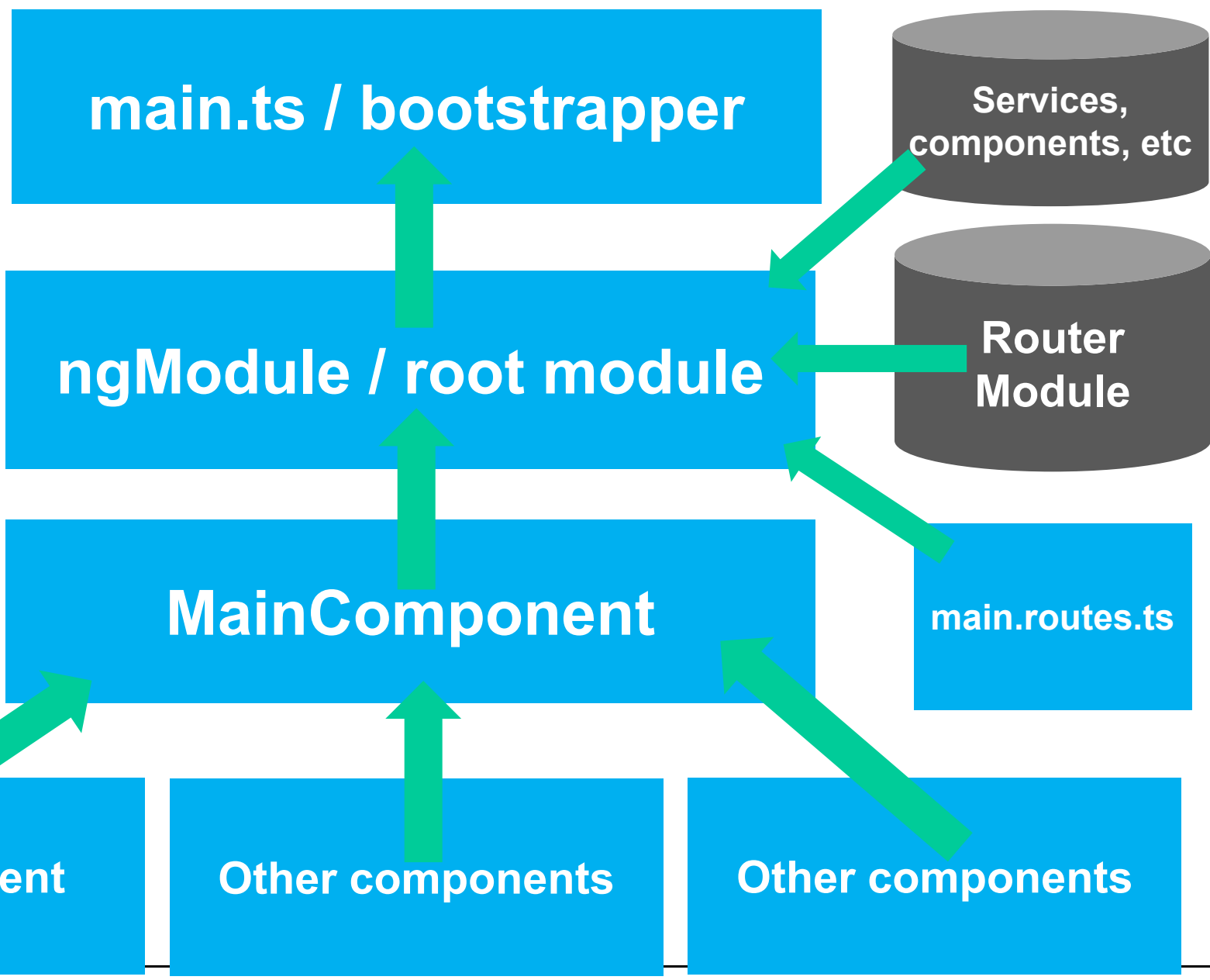
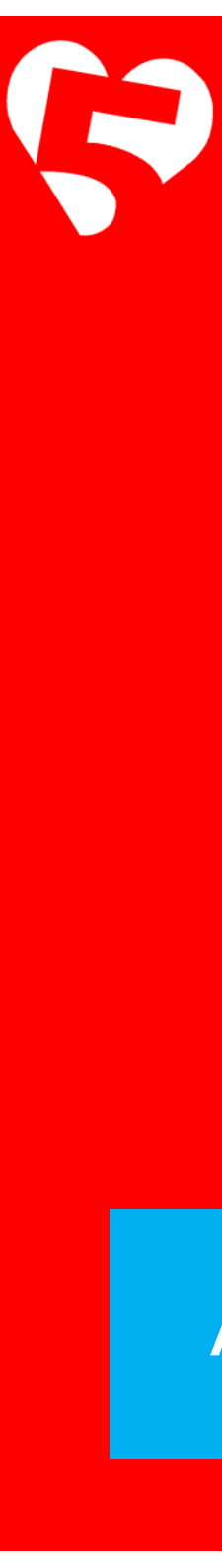
- Niet beschikbaar voor AngularJS 1.4+
- Niet beschikbaar: ui-router



Routing – every route is a Component

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







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.



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';
```

Import Router-
onderdelen

```
...  
// Components  
import {MainComponent} from './MainComponent';
```

Nieuw!
MainComponent
gaan we nog maken

```
...  
@NgModule({  
  imports      : [  
    BrowserModule, HttpClientModule,  
    RouterModule.forRoot(AppRoutes)  
  ],  
  declarations: [  
    MainComponent,  
    AppComponent,  
    CityAddComponent  
  ],  
  bootstrap    : [MainComponent]  
})
```

Configure
RouterModule.forRoot()

MainComponent wordt nu
gebootstrapt

```
export class AppModule {  
}
```




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 'angular2/core';
```

```
@Component({
  selector: 'add-city',
  template: `<h1>Add City</h1>`
})
export class CityAddComponent {
  ...
}
```

```
// city.edit.component.ts
import { Component } from 'angular2/core';

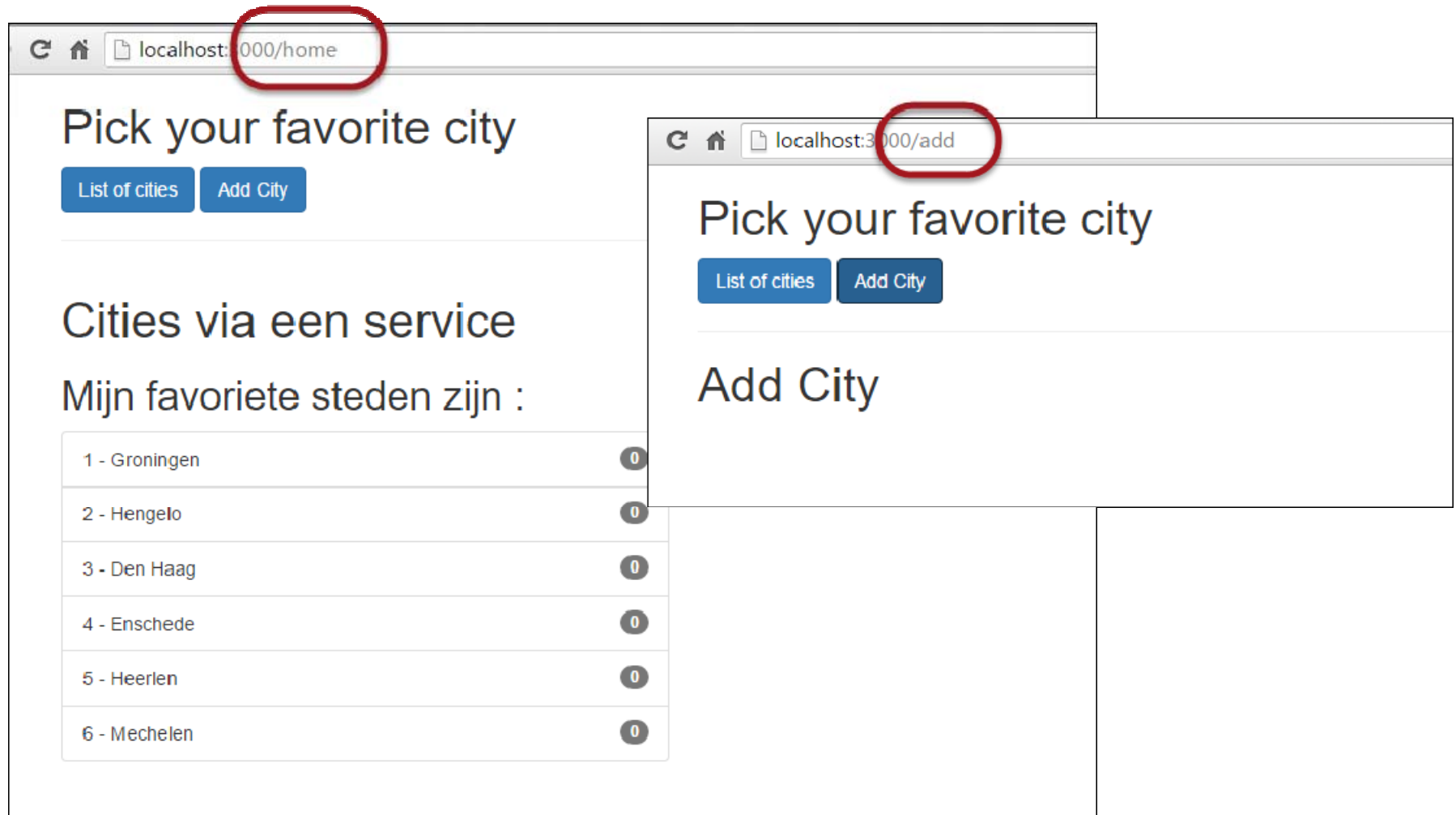
@Component({
  selector: 'edit-city',
  template: `<h1>Edit City</h1>`
})
export class CityEditComponent {
  ...
}
```

```
// city.detail.component.ts
import { Component } from 'angular2/core';

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



8. Testen

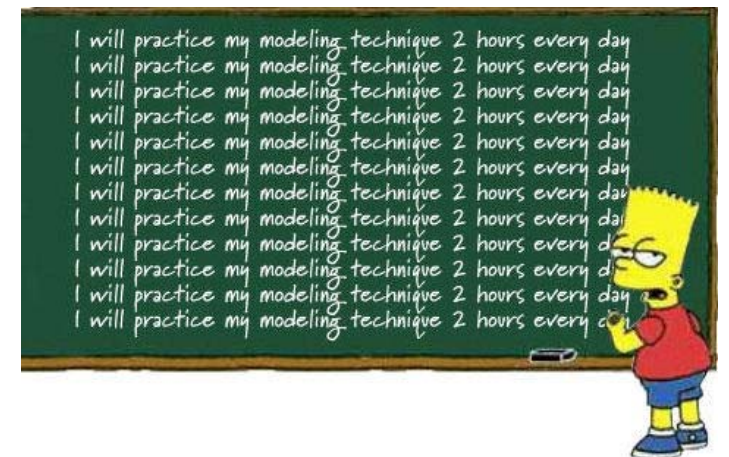




Checkpoint

- Routes worden op component-niveau ingesteld (Angular 1: module-niveau).
- Volg het stappenplan. Denk aan injecteren van RouterModule, `app.routes.ts` en `<base href="/">` in de HTML

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((id: any) => {
        this.id = id;
      });
  }
}
```

ActivatedRoute



2a. DetailComponent - variants

Using router snapshots

```
// 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'];  
  });  
}
```

.unsubscribe()

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

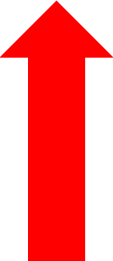


Component



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>
```

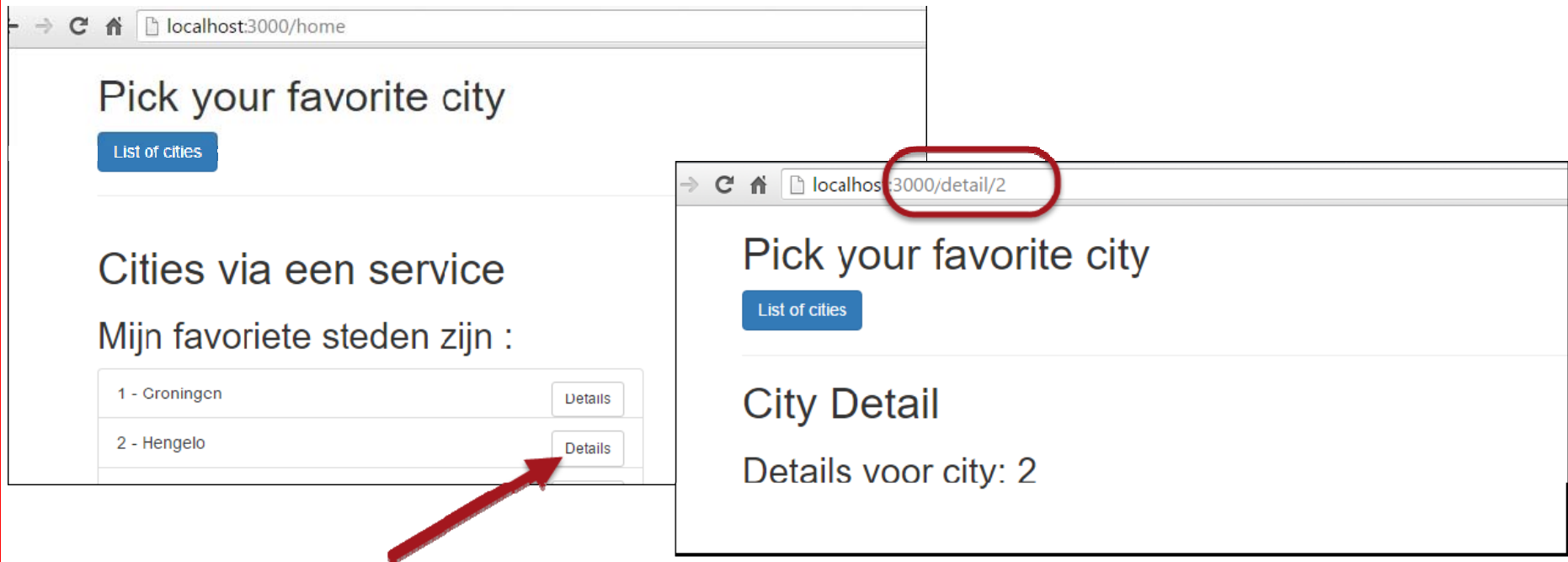
A large red arrow pointing upwards from the text below towards the `[routerLink]` attribute in the code above.

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 – details via Service

- Uncomment de regels die te maken hebben met cityService:

// NEW, with fetching details via Service:

```
this.sub = this.route.params
    .map(params => params['id'])
    .switchMap(id => this.cityService.getCity(id))
    .subscribe((city) => {
        this.currentCity = <City>city[0];
    });
```

Pick your favorite city

List of cities

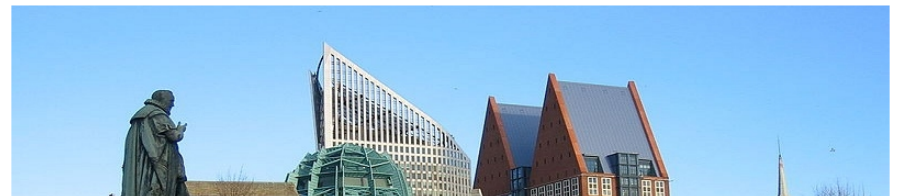
City Detail

Details voor city: 3

Naam: Den Haag

Provincie: Zuid-Holland

Highlights: Binnenhof





In city.service.ts:

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

// retourneer een city, op basis van ID

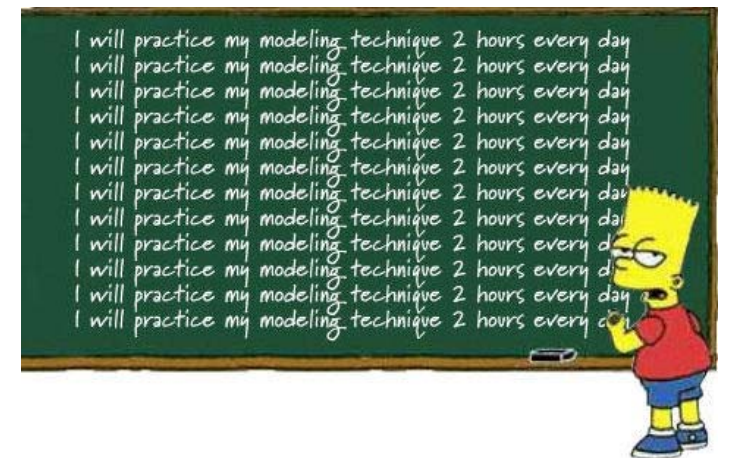
```
getCity(id: string): City[] {  
    return this._http.get('app/cities.json')  
        .map(cities => cities.json())  
        .map(cities => cities.filter((city: City) => {  
            return city.id === parseInt(id);  
        })))  
}
```




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.

Oefening....





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



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



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 ch
    }
    return true;
  }
}
```

A large red arrow points upwards from the bottom of the slide, specifically towards the 'return true;' line in the code block.



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]
  },
  ...
];
```

A thick red arrow pointing upwards, highlighting the `CanDeactivateGuard` in the `canDeactivate` array of the route configuration.



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
  }
}
```





Child Routes

Nested routing, routing per component



[TBS]

- Zie ook https://angular-2-training-book.rangle.io/handout/routing/child_routes.html



Meer over routing

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



New Component Router

⌵

Root Component

Outlet

Child Component

Leaf Component

Lazy Loading

NG

2:48 / 16:36

Routing - Misko Hevery

<https://www.youtube.com/watch?v=d8yAdeshpcw>



Victor Savkin (=maker van de router)





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

The video player shows a presentation slide titled "Angular Router: Lazy Loading". The slide lists the following topics:

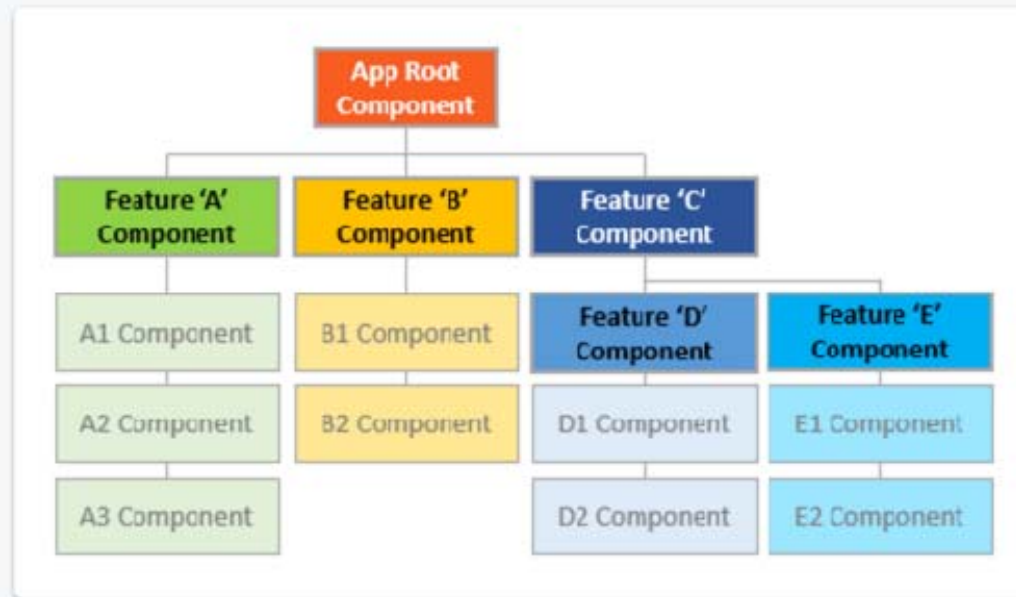
- Status of Angular Router
- Overview
- Lazy Loading
- Preloading
- Bundling

A small inset video in the top right corner shows the presenter, Victor Savkin, at a podium. The video player includes a progress bar at 0:49 / 20:19, a subtitle "Today, I'm happy to show you how it works.", and the Angular logo (A and C) in the bottom right corner of the slide area. The video title "The Angular Router | Victor Savkin" is displayed in the bottom left, and "Volgende" (Next) and "Autoplay" options are in the bottom right.



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

Victor Savkin – creator of the router

Victor Savkin on Angular 2
In-depth articles about Angular 2 by a core team member.

Angular Router: Understanding Router State

An Angular 2 application is a tree of components. Some of these components are reusable UI components (e.g., list, table), and some are...

Victor Savkin
Oct 30

ANGULAR 2 ROUTER

Angular 2 Core Concepts

- Components
- Dependency Injection
- Property Bindings
- Single API
- Hierarchical

Tackling State

- State
- Why?
- Application #1: View Boundary
- Application #2: View Boundary
- Application is `fu(a:Observable):Observable`

<https://vsavkin.com/>