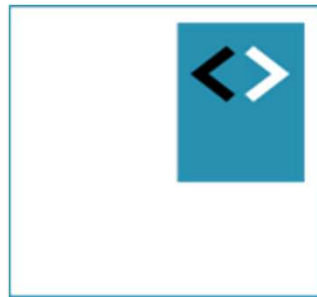




Angular Fundamentals

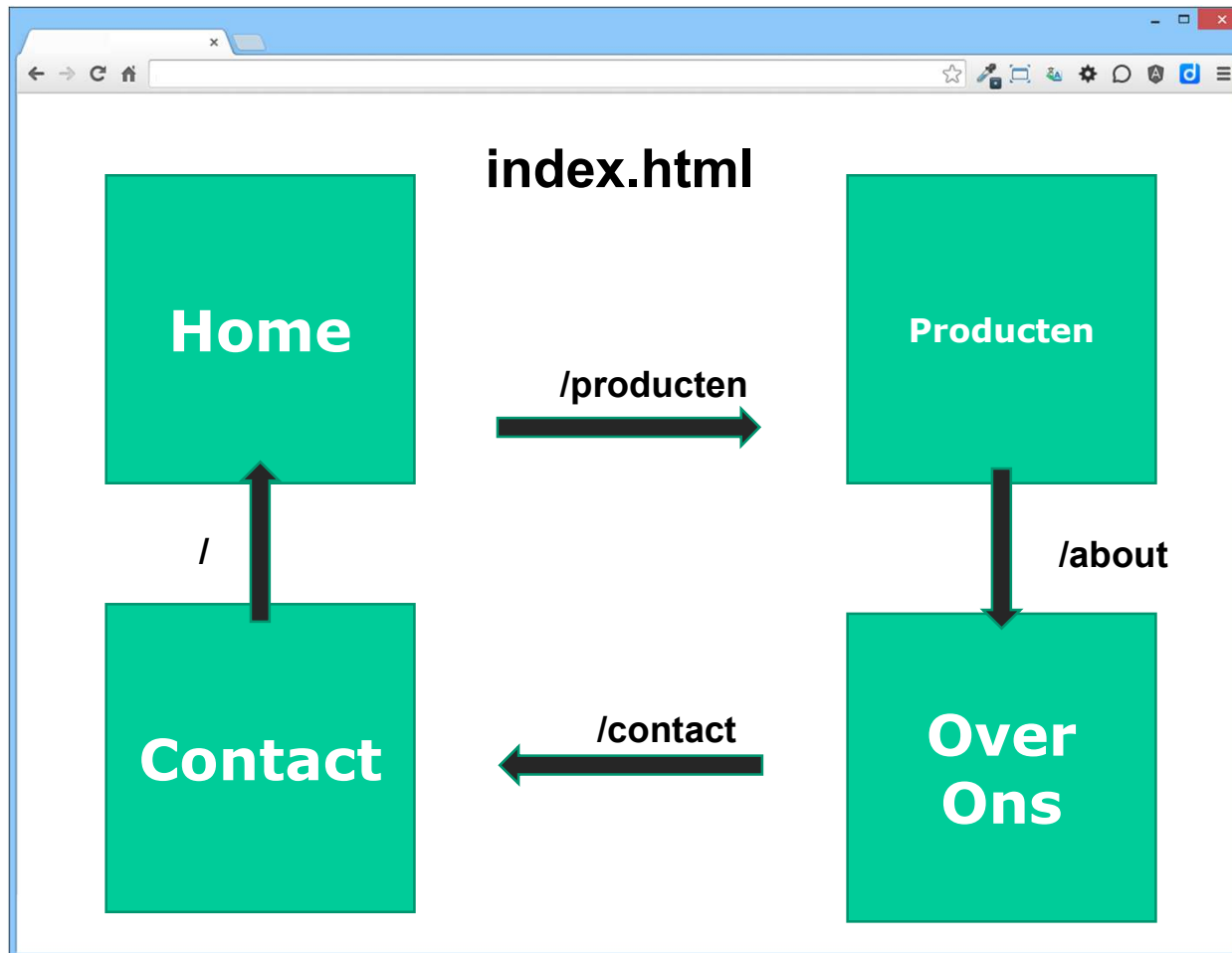
Module 6 – Routing

Lendex^{powered by} NIBC



Peter Kassenaar
info@kassenaar.com

Routing architecture and goal



- Make use of SPA principle
- Making deep links possible

Angular 1: ng-route, or 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']);`

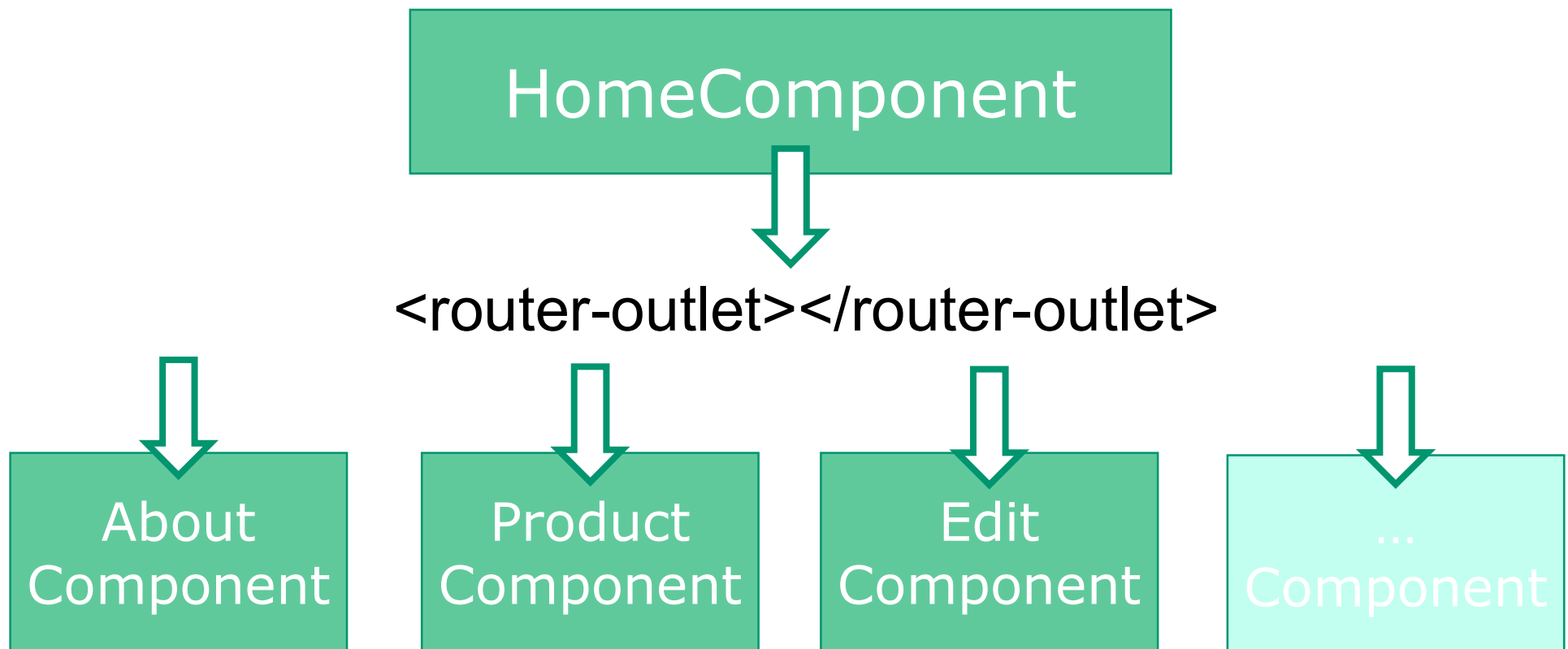
Then: configure `$routeProvider` (or `$stateProvider` with ui-router)

Angular 2+: Component Router

- All-in-one solution
- NOT available for AngularJS 1.4+

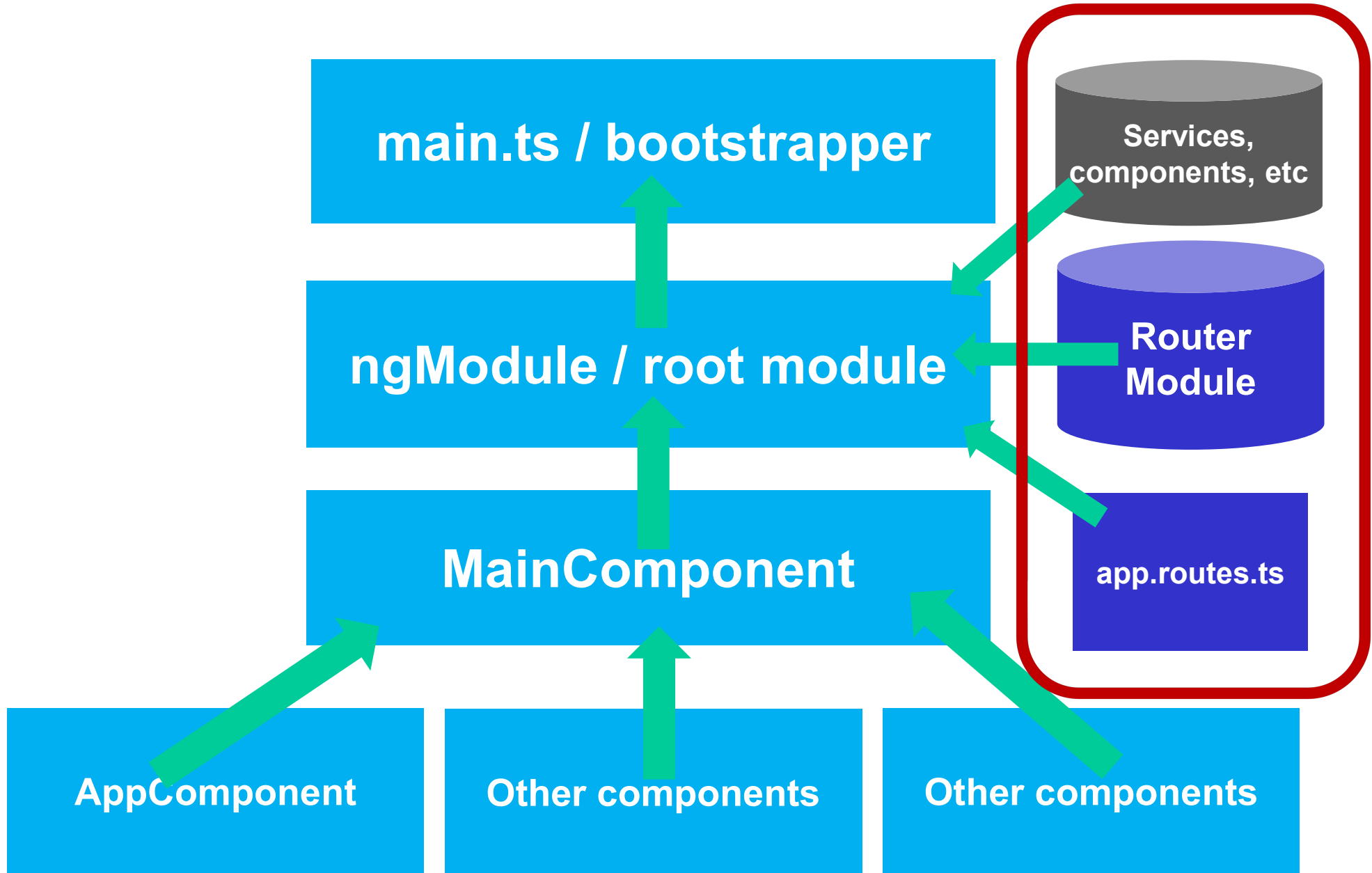
Routing – every route is a Component

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



Routing with Angular CLI

- Default: no routing in CLI-projects
- Add routing from the start?
 - `ng new myProject --routing`
 - OR – pick from the CLI options menu on `ng new`
- This creates `app-routing.module.ts` in project
- (a little) different than the approach in this module
 - We add routing later on – so you'll learn what components are used



Routing – Step 1

1. Check base href in header of index.html (!)

<base href="/">

- There *can* be multiple routes per module. Each component can configure its own `ChildRoutes` – to be discussed.
- Angular-CLI adds this automatically for you

Step 2

2. Add routes. Convention: `app.routes.ts` or `app.routing-module.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}
];
```

*Note: Some people or tools use **different notation** on declaring routes*

3. Make routes available in Module

Import RouterModule in application

Import ./app.routes in application

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

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

Import Router
stuff

New!
MainComponent.
To be created

Configure
RouterModule.forRoot()

MainComponent is now
bootstrapped

4. Create MainComponent with Routing

- New component with main menu and `<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() { }
}
```

5. Edit index.html

- IF `MainComponent` has a different selector, update `index.html`

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

6. Create new components and import

Every component is a route

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

```
@Component({
  selector: 'city-add',
  template: `...`
})
export class CityAddComponent {
  ...
}
```

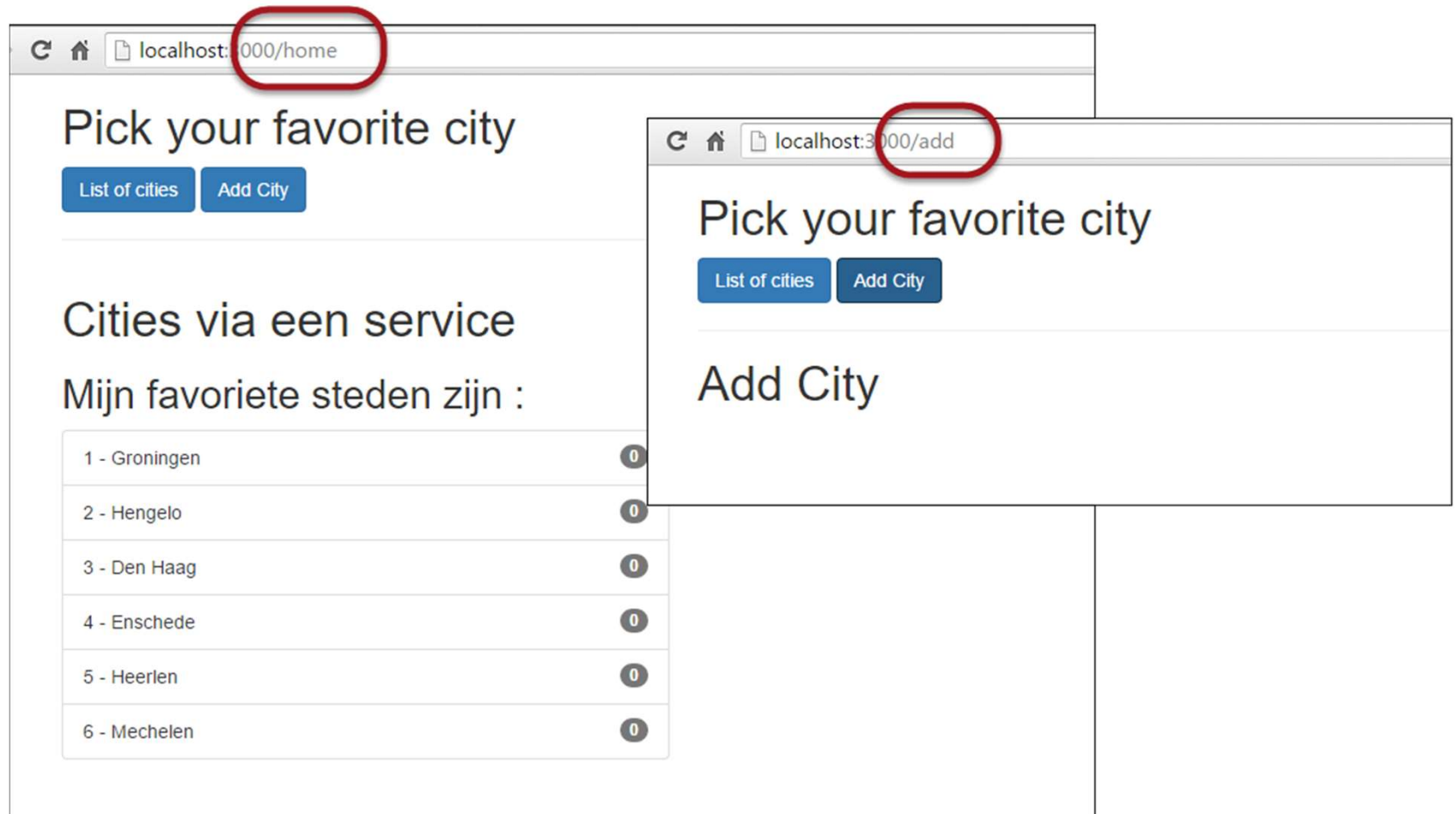
```
// city.edit.component.ts
import { Component } from '@angular/core';
```

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

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

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

7. Run the application



Catch-all routes

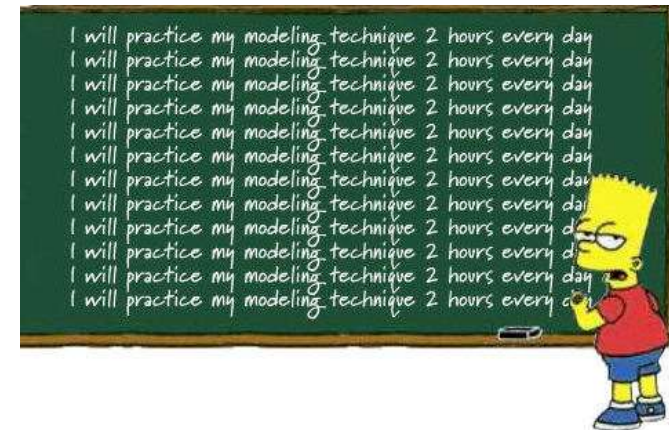
```
6 export const AppRoutes: Routes = [  
7   {path: '', component: AppComponent},  
8   {path: 'home', component: AppComponent},  
9   {path: 'add', component: CityAddComponent},  
10  {  
11    // catch all route  
12    path: '**',  
13    redirectTo: 'home'  
14  },  
15 ];
```

Use `**` as a catch-all route:

- `redirectTo`: route you want to show in address bar.
- The component is mentioned in the route that is pointed at.

Checkpoint

- Routes are created on module level (Angular 1: app level).
- Follow the steps. Remember to inject `RouterModule`, create `app.routes.ts` **en** `<base href="/">` and so on.
- Example: `/400-router` (`npm install` **and** `npm start`)
 - Add a new component to the routing example and make sure users can navigate to this component/route.
- Exercise: 7a). Optional: 7b) (=create app)
- Official docs:
<https://angular.io/guide/router>





Routeparameters

Master-Detail views and –applications


Dynamic routes

- Goal: Single detail page for customers, products, services, etc.
- Readable routes like: `/cities/5`, or `products/apple/iphone`, and so on
- Method:
 1. Edit `app.routes.ts` and hyperlinks on the page.
 2. Use `route:ActivatedRoute` in detail component
 3. Write hyperlinks like `<a [routerLink]=[...]>` with parameter

1. Edit app.routes.ts

```
// 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. Create Detail Component

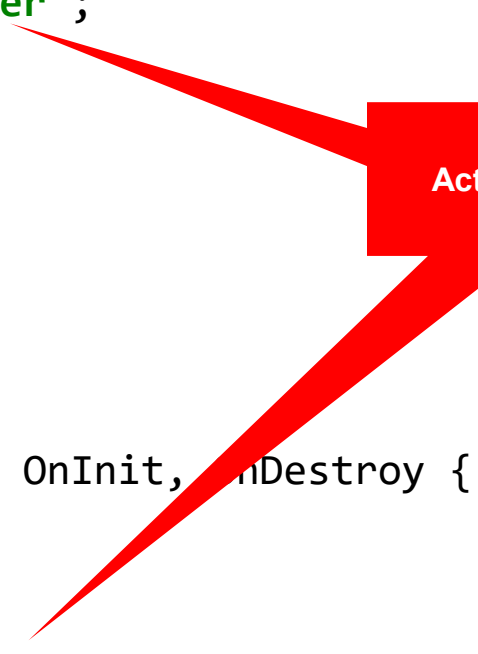
```
// city.detail.component.ts
...
import {ActivatedRoute} from '@angular/router';

@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;
      });
  }
}
```



A red rectangular box labeled "ActivatedRoute" has two red arrows pointing to the code. One arrow points to the import statement `import {ActivatedRoute} from '@angular/router';` and the other points to the constructor parameter `private route: 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. Add Detail component to Module

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


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



Component

3. Edit App Component ('Master View')

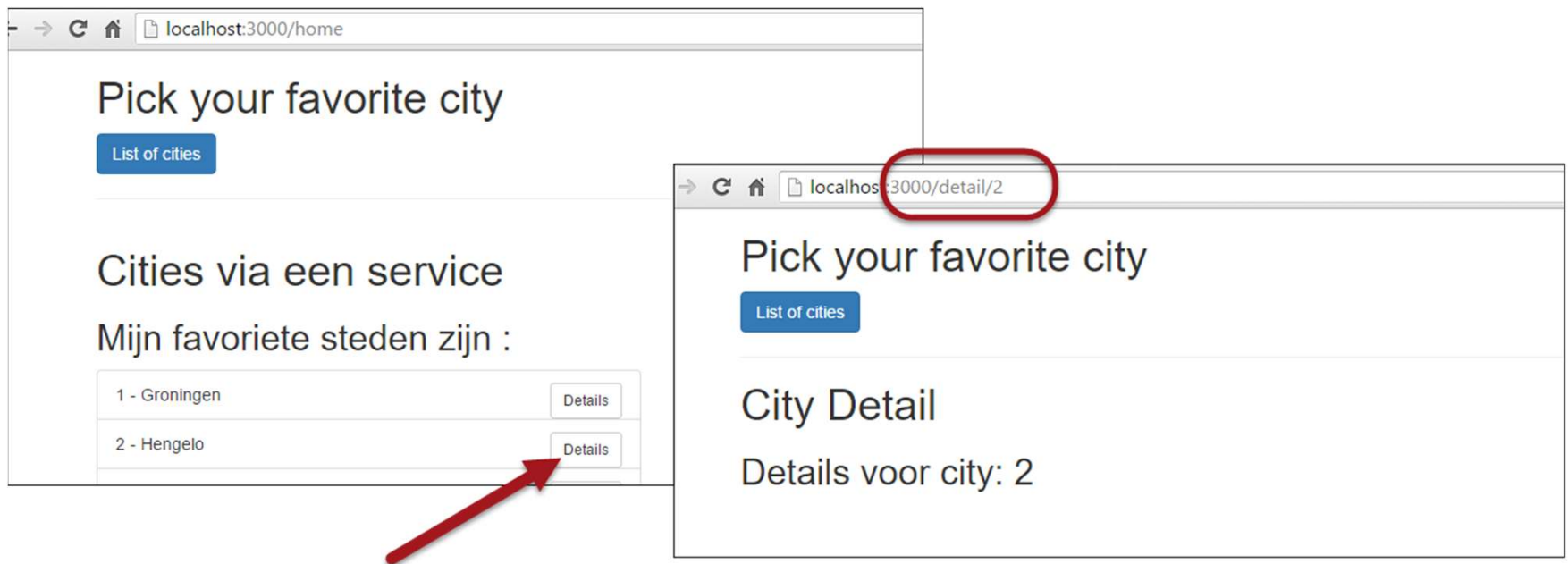
```
<li *ngFor="let city of cities" class="list-group-item">
  <a [routerLink]="['/detail', city.id]">
    {{ city.id }} - {{ city.name }}
  </a>
</li>
```



Remember that `[routerLink]` should now be calculated dynamically and thus should be written with `[...]` for attribute binding

Passing parameters

- You pass an *array of parameters* to `[routerLink]`
- Parameters are matched on position. Not on name.
- Optional : extend service to return specific product/item



Optional parameters : [queryParams]

HTML

```
<a [routerLink]="['/detail', city.id, city.name]"
  [queryParams]="{province:city.province, population:180000}">
  {{ city.id}} - {{ city.name }}
</a>
```

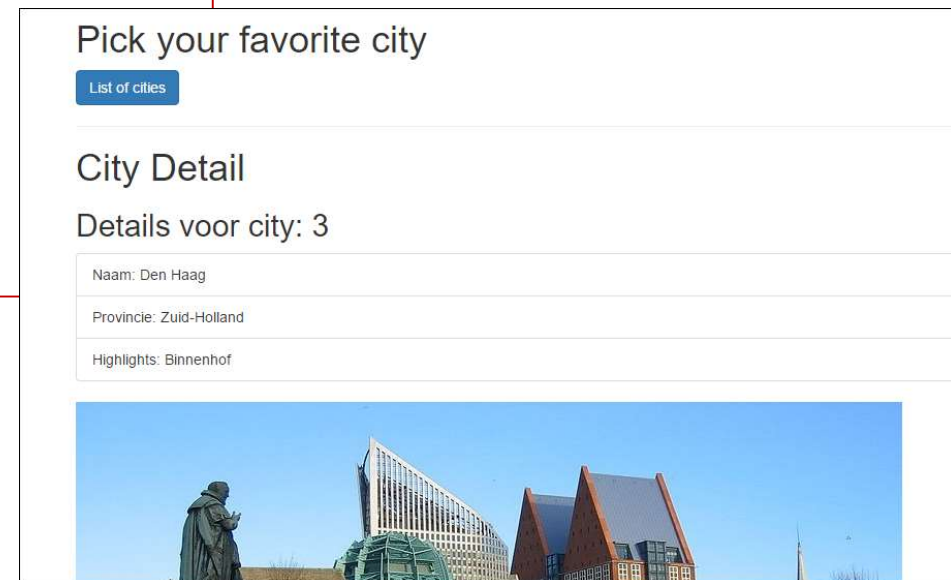
Class

```
this.route.queryParams.subscribe((params: any) => {
  this.province = params.province;
})
```

Next up – details via Service

- Make sure to add a method like `.getCity(id)` that returns a city, based on id.

```
// NEW, with fetching details via Service:  
this.sub = this.route.params.pipe(  
  map(params => params['id']),  
  switchMap(id => this.cityService.getCity(id))  
)  
.subscribe((city:City) => {  
  this.currentCity = city;  
});
```



In city.service.ts:

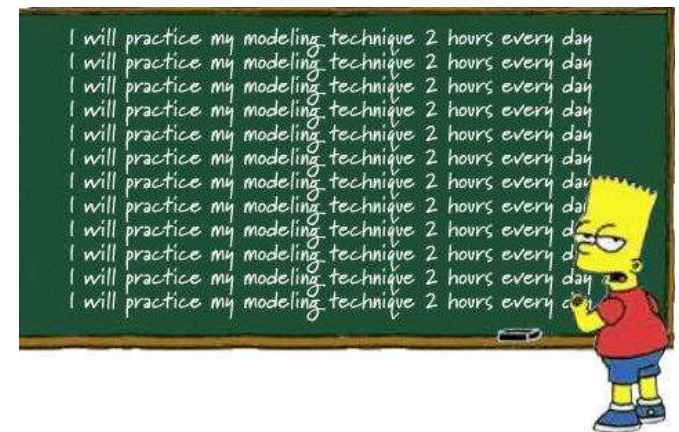
- Edit/add a method to return a specific city

```
// return a city, based on Id
getCity(id: string): City[] {
    return this.http.get<City>('app/cities.json').pipe(
        map(cities => cities.filter((city: City) => {
            return city.id === parseInt(id);
        }))
    )
}
```

Checkpoint

- RouteParameters are set with `:parameterName` in `app.routes.ts`.
- Remember to inject `ActivatedRoute` in component.
- Use the property `.params` to retrieve the passed in values.
- Example: `\401-route-parameter`
- Exercise 7c)

Exercise....



Additional routing techniques

- Router Guards – Secure parts of your application, based on Auth-logic
- Child Routes
- Named Router Outlets
 - <http://onehungrymind.com/named-router-outlets-in-angular-2/>
- Router resolvers
 - <https://blog.thoughttram.io/angular/2016/10/10/resolving-route-data-in-angular-2.html>
- Lazy Loading – Split app in Modules and load *on demand*
 - <https://angular.io/guide/router#lazy-loading-route-configuration>



More info

More background information on routing

Some hyperlinks

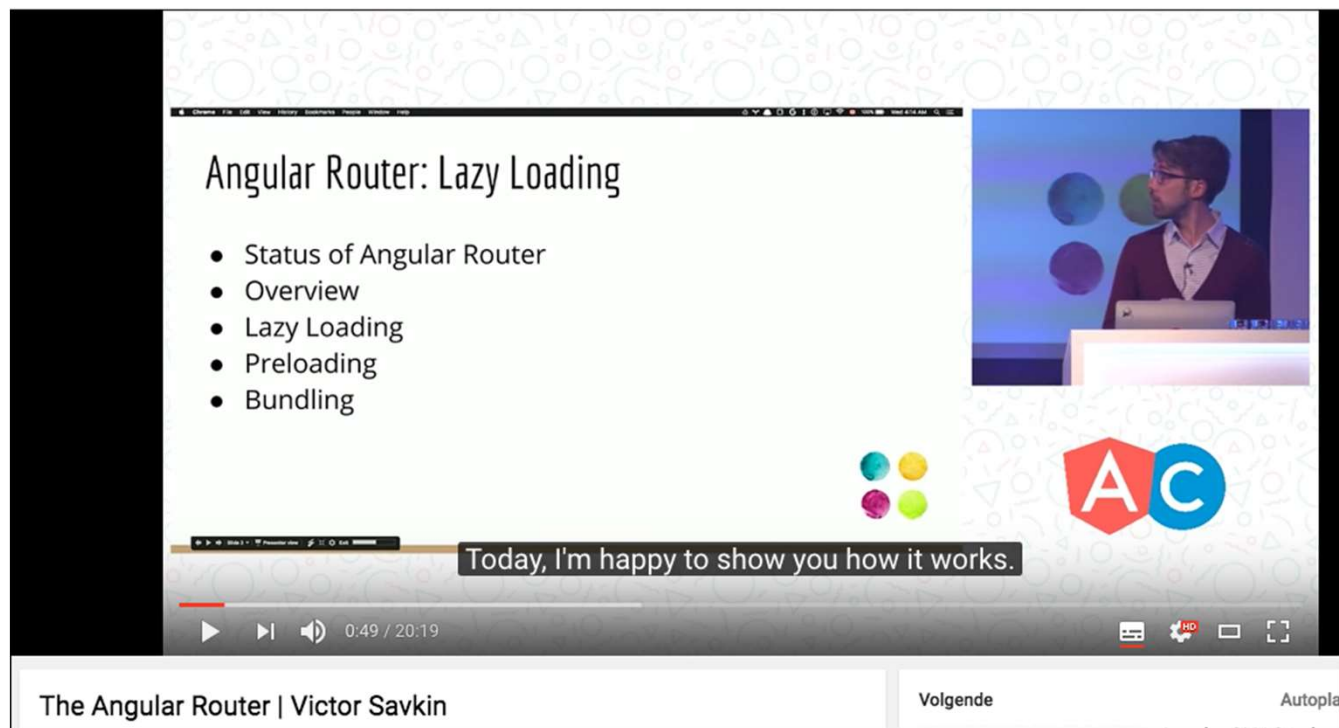
- <https://angular.io/docs/ts/latest/guide/router.html>
- <http://blog.thoughttram.io/angular/2016/06/14/routing-in-angular-2-revisited.html>
- <http://blog.thoughttram.io/angular/2016/07/18/guards-in-angular-2.html>
- <https://vsavkin.com/>
- https://angular-2-training-book.rangle.io/handout/routing/child_routes.html

Victor Savkin (=creator of the router)



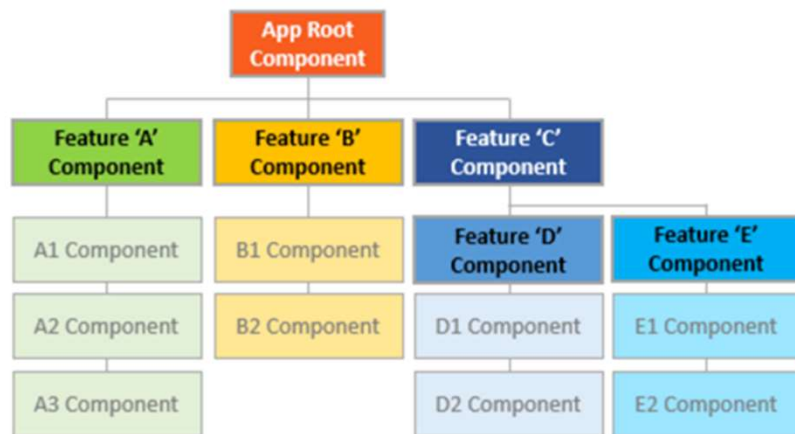
<https://leanpub.com/router>

<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/>