



Global Knowledge®

Angular Fundamentals

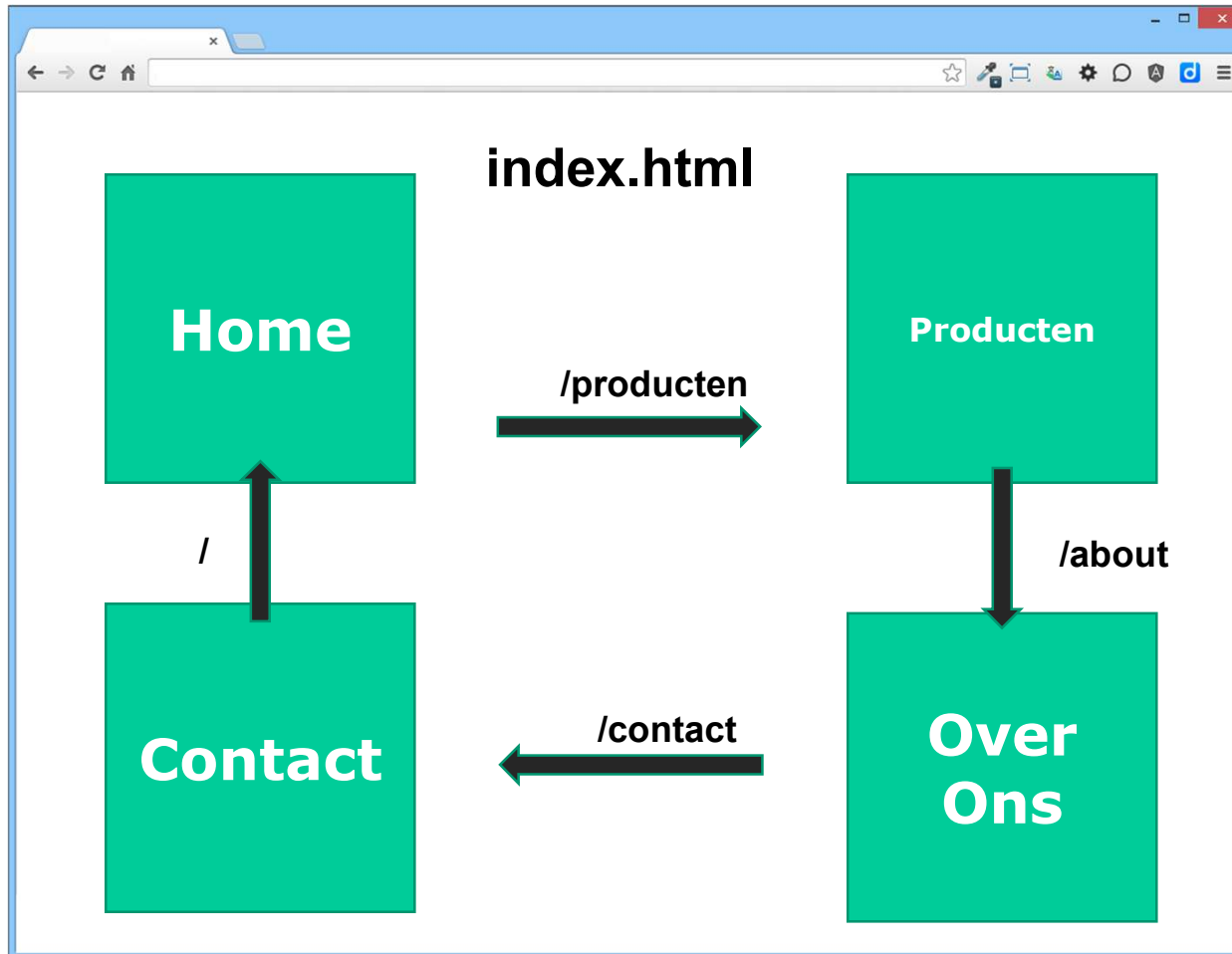
Routing

Peter Kassenaar –
info@kassenaar.com

WORLDWIDE LOCATIONS

BELGIUM CANADA COLOMBIA DENMARK EGYPT FRANCE IRELAND JAPAN KOREA MALAYSIA MEXICO NETHERLANDS NORWAY QATAR
SAUDI ARABIA SINGAPORE SPAIN SWEDEN UNITED ARAB EMIRATES UNITED KINGDOM UNITED STATES OF AMERICA

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']);`

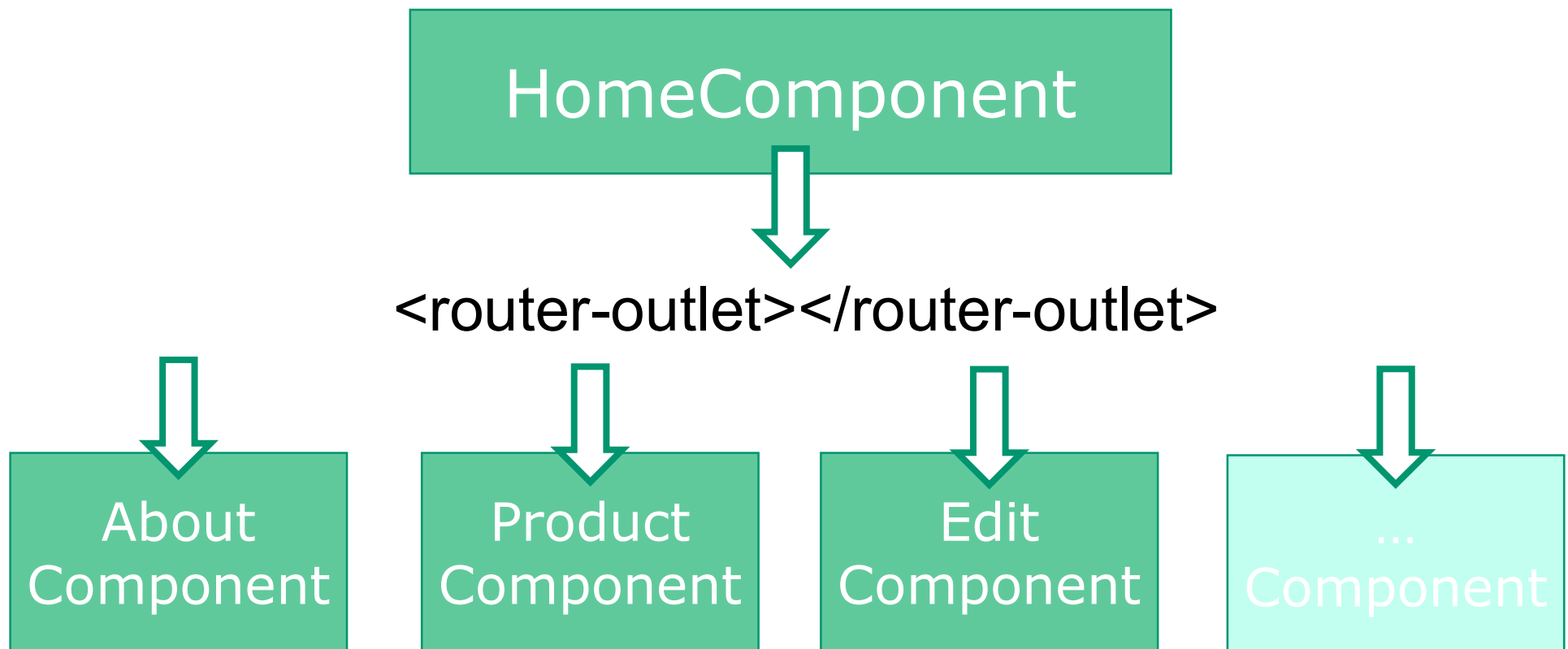
Daarna `$routeProvider` configureren (of `$stateProvider` bij ui-router)

Angular 2+: Component Router

- All-in-one solution
- NOT available for older AngularJS applications

Routing – every route is a Component

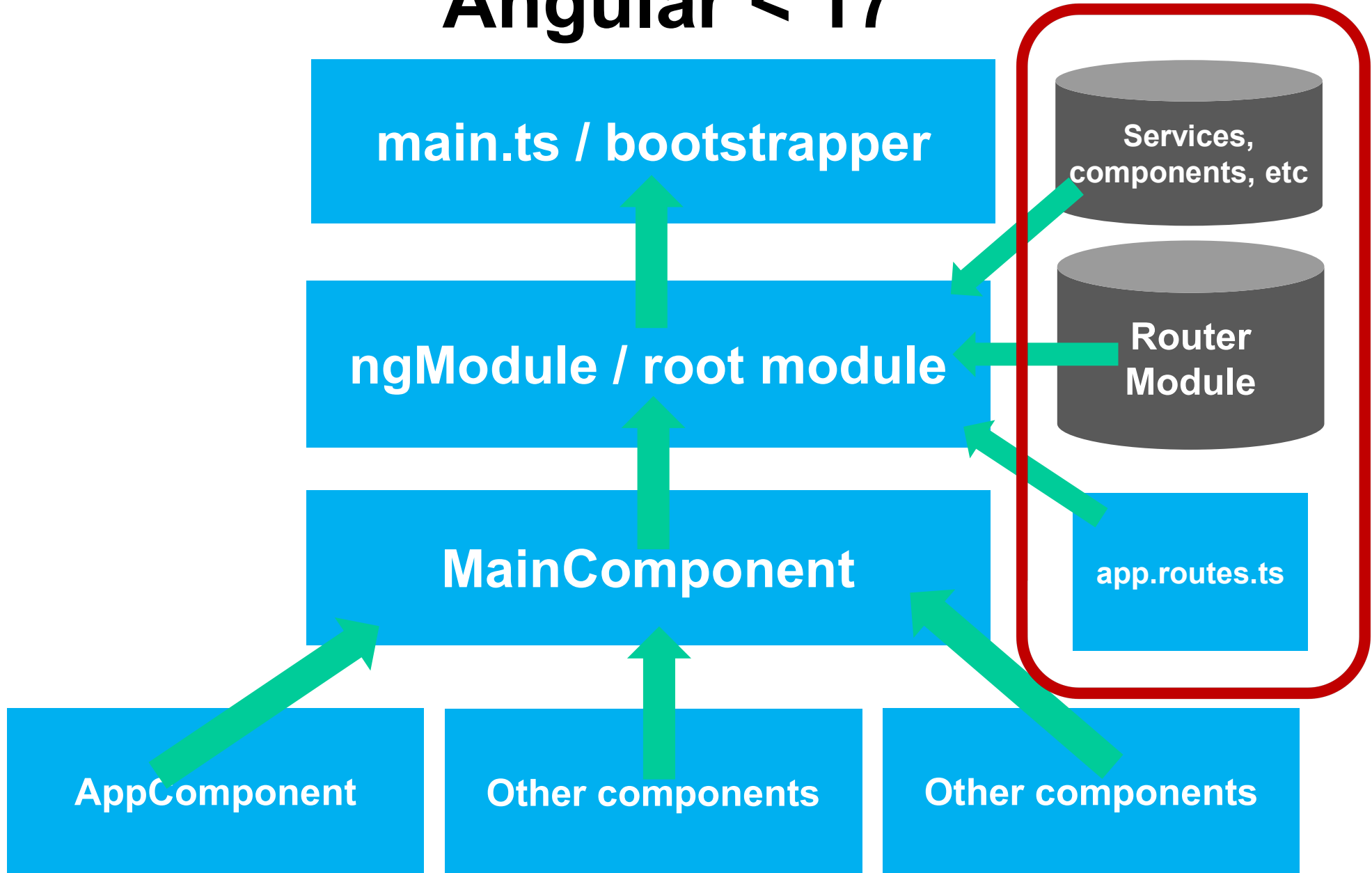
- HomeComponent (or: RootComponent, whatever) with **main menu**
- Components are injected in `<router-outlet></router-outlet>`



Routing with Angular CLI < v17

- 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

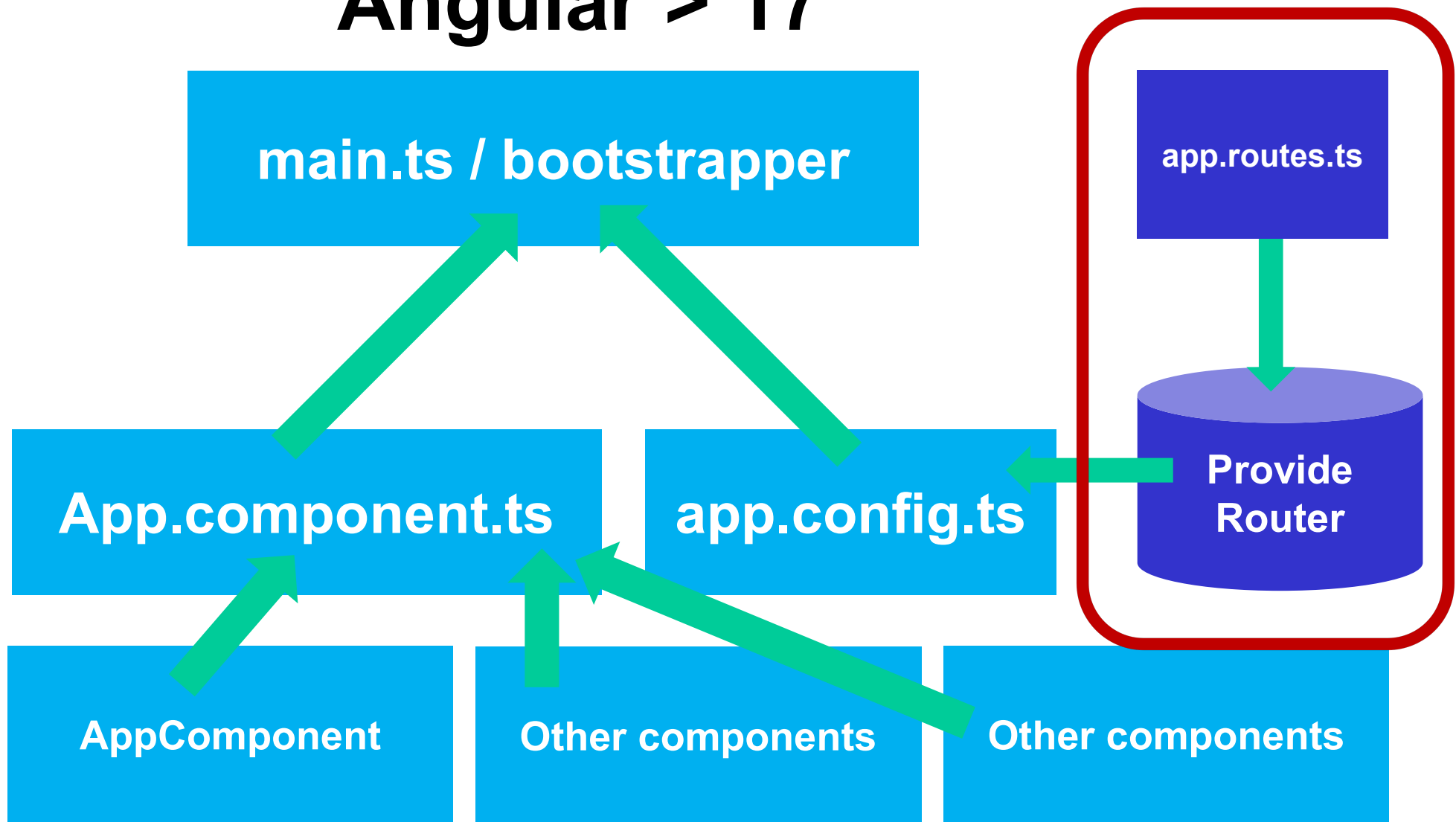
Angular < 17



Routing with Angular CLI > v17

- Default: **routing available** in CLI-projects
- Look at `<router-outlet />` at the bottom of `app.component.html`
- Look at created `app.routes.ts`
- Router is added to application in `app.config.ts`
- Newer Angular applications apply **the same principles**, though the syntax is a bit different!

Angular > 17



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
- But it is always good to check!

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*

3a. Angular < 17: 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

3b. Angular > 17: Provide your routes

Use `app.config.ts` in application

```
// app.config.ts
import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';

export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes)
  ]
};
```

Import Router
stuff

Add Router to app

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() { }
}
```

**"Main Menu".
Notice routerLink**

<router-outlet>

Empty Component

6. Create new components and import

Every component is a route

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

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

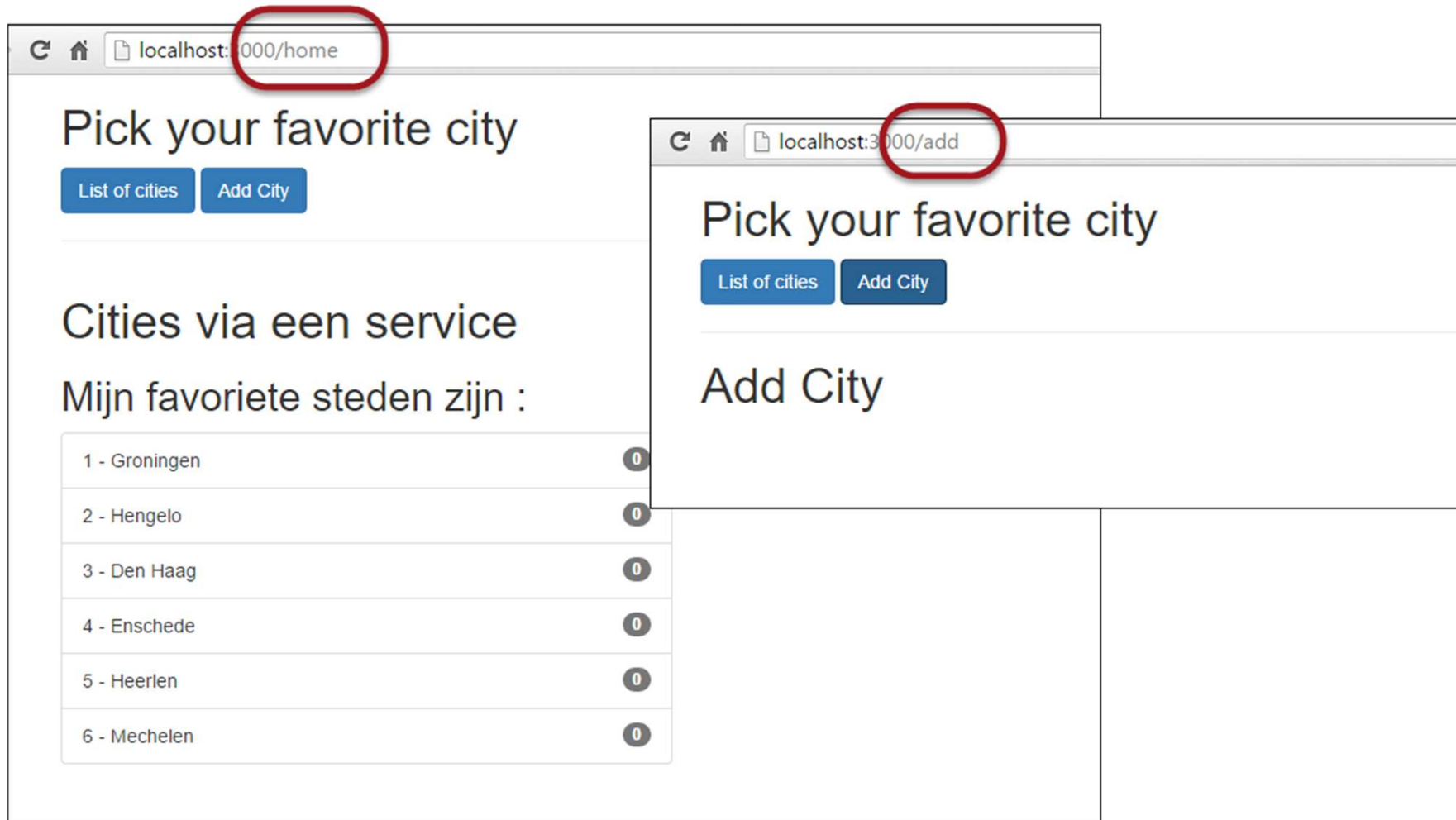
```
// city.edit.component.ts
import { Component } from '@angular/core';
```

```
@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:

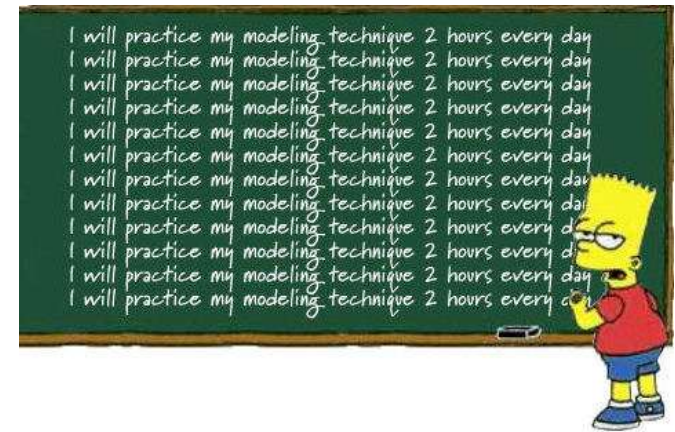
- `redirectTo`: route you want to show in address bar.
- The component is mentioned in the route that is pointed at.

To summarize – Steps

1. Check `<base href>` in your index.html
2. Create a routing table with `const myRoutes: Routes`
3. Add routing table with `RouterModule.forRoot(...)`
 1. OR: use `app.routes.ts` and/or `app.config.ts`, depending on Angular version
4. Create Main menu with `` and
`<router-outlet />`
5. Run the application

Workshop

- 1. **Own application:**
 - Follow the steps. Remember to inject `RouterModule`, create `app.routes.ts` and `<base href="/">` and so on.
- 2. **Example:** /400-routing
 - (`npm install, npm start`)
 - Add a new component to the routing example and make sure users can navigate to this component/route.
- 3. **Optional:**
 - Create new Angular 17+ app, add routing + components from scratch
 - Use the CLI, `ng new <...>`
 - Make sure use can navigate to different routes
- **Pick one (1) workshop**
- Official Docs : <https://angular.dev/guide/routing/common-router-tasks>





Routing in Modern Angular

Summary: Angular 17+ applications have a different notation/structure


App.routes.ts en app.config.ts

```
// app.routes.ts -  
// Default: EMPTY routing table. Define your own routes.  
import { Routes } from '@angular/router';  
  
export const routes: Routes = [];
```

```
// app.config.ts - provide your routes to the application.  
import { ApplicationConfig } from '@angular/core';  
import { provideRouter } from '@angular/router';  
  
import { routes } from './app.routes';  
  
export const appConfig: ApplicationConfig = {  
  providers: [provideRouter(routes)]  
};
```

Main.ts

```
// main.ts - pass in appConfig to configure the application,  
// containing the routes you created.  
import { bootstrapApplication } from '@angular/platform-browser';  
import { appConfig } from './app/app.config';  
import { AppComponent } from './app/app.component';  
  
bootstrapApplication(AppComponent, appConfig)  
  .catch((err) => console.error(err));
```





Routeparameters

Master-Detail views en –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;
      });
  }
}
```



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'];  
    });  
}
```

```
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. Add Detail component to Module

(Angular 17+ : import detail component in main component, instead of 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')

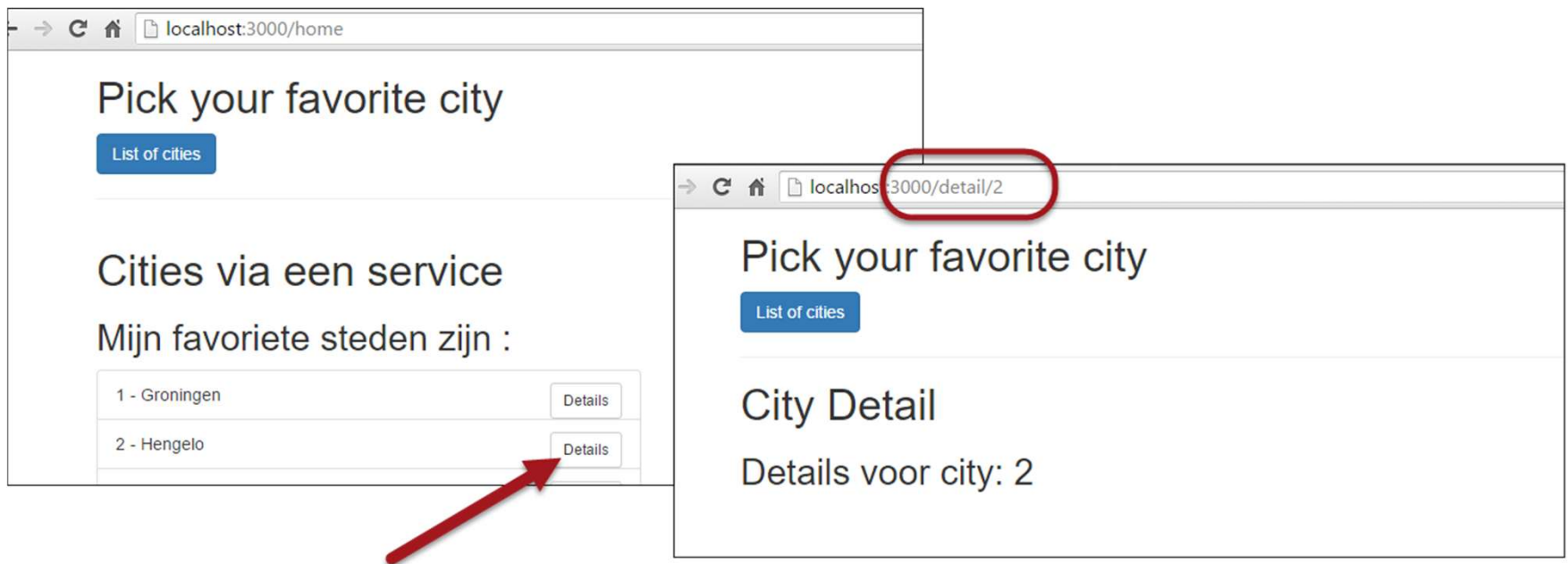
```
<li *ngFor="let city of cities" class="list-group-item">
  <a [routerLink]="['/detail', city.id]">
    {{ city.id }} - {{ city.name }}
  </a>
</li>
```



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





Optionele parameters

Optionele routeparameters meegeven op de querystring

Optional parameters : [queryParams]

In HTML

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

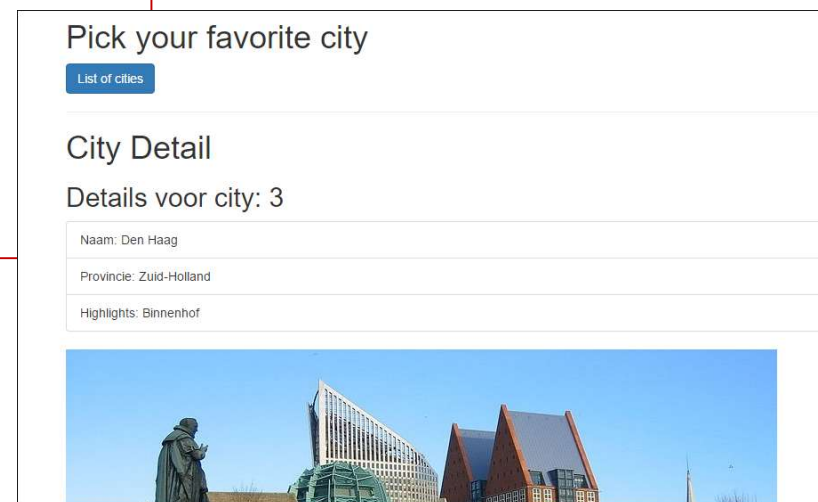
In class

```
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;  
  });
```



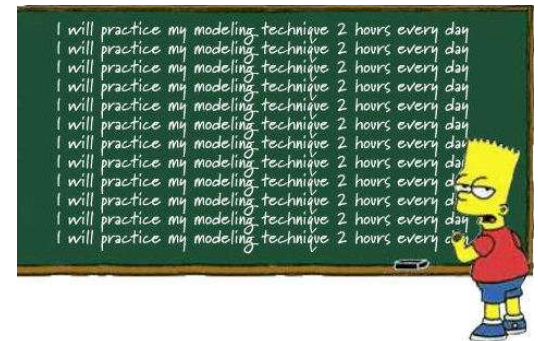
For instance, in `city.service.ts`:

- 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);
        }))
    )
}
```

Workshop

- 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.
- Create dynamic `[routerLink]` attributes
- Goal: **Add dynamic routes to your own application!**
- Example: `\401-route-parameter`



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

Achtergrondinfo

- <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

On Component Router

The video player displays a presentation slide titled "On Component Router". The slide illustrates a component hierarchy for routing in Angular. At the top, a browser address bar shows the URL `http://my.app/user/123/details`. Below this, a diagram shows a "Root Component" (green box) containing an "Outlet" (dashed box). An arrow points from the "Outlet" to a stack of "Child Component" boxes (yellow). Another arrow points from the "Child Component" stack to a stack of "Leaf Component" boxes (blue). Below the "Child Component" stack, there is a puzzle piece icon labeled "Lazy Loading". The video player interface at the bottom shows a progress bar at 2:48 / 16:36 and the title "Routing - Misko Hevery".

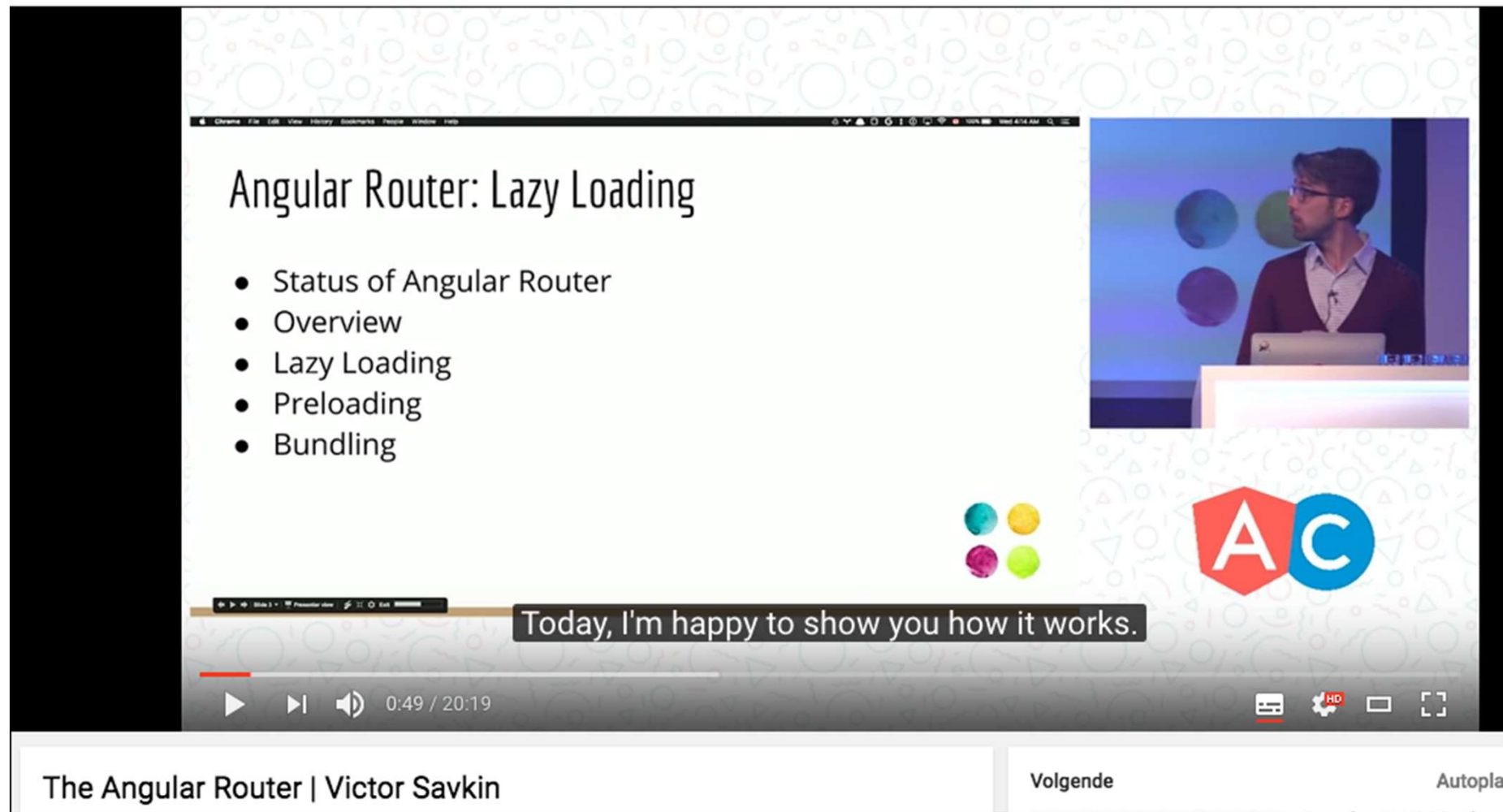
<https://www.youtube.com/watch?v=d8yAdeshpcw>

Victor Savkin (=creator of the router)



<https://leanpub.com/router>

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



The image shows a YouTube video player interface. The video content is a presentation slide titled "Angular Router: Lazy Loading" with a bulleted list of topics: Status of Angular Router, Overview, Lazy Loading, Preloading, and Bundling. The slide also features the Angular logo and a small inset video of the presenter, Victor Savkin. The video player controls at the bottom show the video is at 0:49 / 20:19. The video title "The Angular Router | Victor Savkin" is displayed on the left, and "Volgende" (Next) and "Autoplay" options are on the right.

Angular Router: Lazy Loading

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

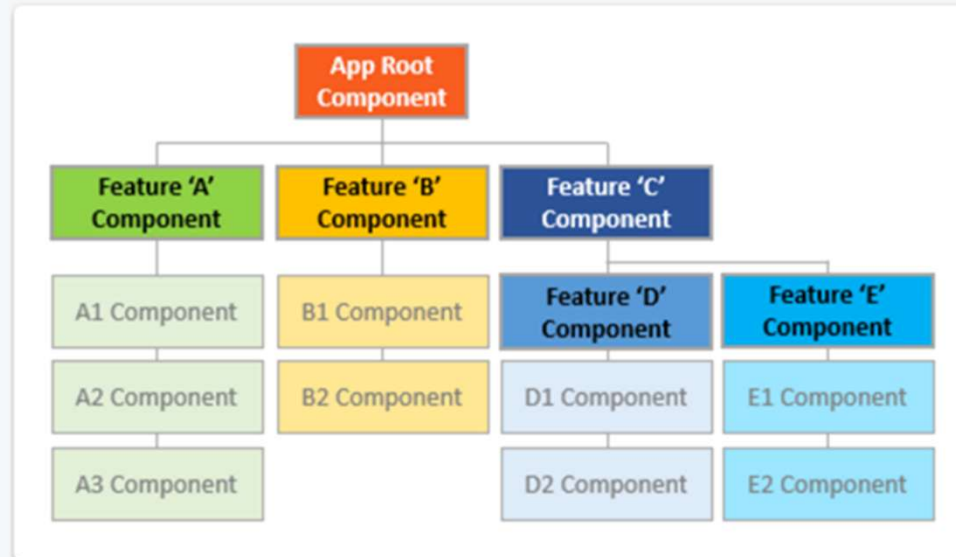
Today, I'm happy to show you how it works.

The Angular Router | Victor Savkin

Volgende Autoplay

Advanced routing

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

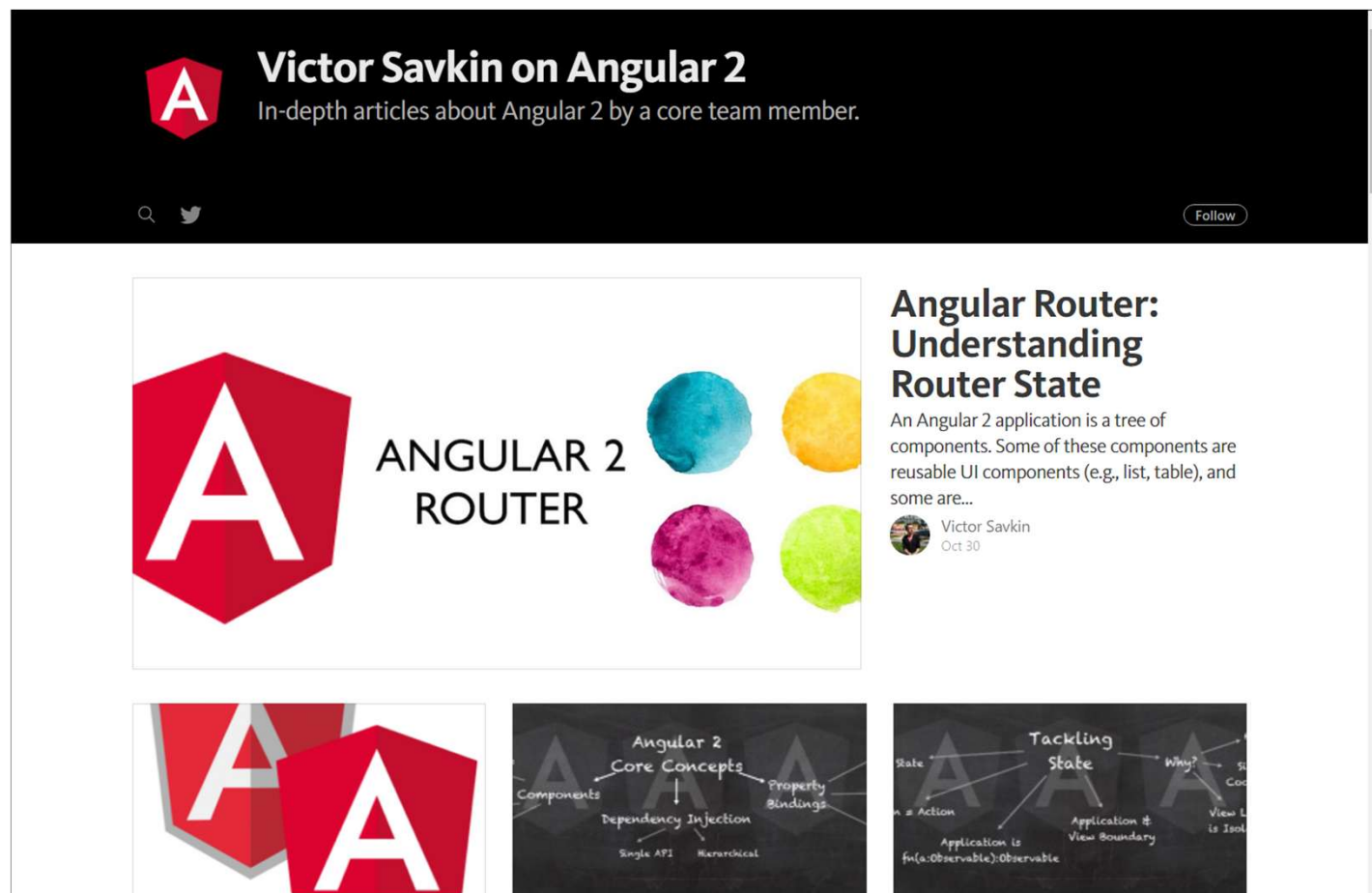


- each feature area in its own module folder
- each area with its own root component
- each area root component with its own router-outlet and child routes
- area routes rarely (if ever) cross

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

Victor Savkin on Routing

Victor Savkin – creator of the router



<https://vsavkin.com/>