SECURE BY ACCIDENT

by André Jaenisch

5th February 2023

IT'S A ME!

André Jaenisch Web-Development & -Consulting

Freelancer

Mastodon: @RyunoKi@layer8.space

WHOM THIS TALK IS FOR?

Experience with Angular, TypeScript and Webpack

Interest in security and performance

WHAT YOU WILL LEARN TODAY

1. Steps to reproduce
2. Interpret webpack build
3. Enumerating child routes
4. Protecting routes with guards
5. Code splitting by route in Angular

BACKGROUND STORY

Idea sponsored by percidae (https://twitter.com/percidae_public)

Chat in fourth quarter of 2022 about the structure of Angular builds

Learned about what information should better

BEFORE WE BEGIN

Angular is used as an *example* here. The following applies to other frameworks as well. It *cannot* be handled on a framework level. The responsibility lies with the app developer. That's **YOU**.

BEFORE WE BEGIN

Angular is used as an *example* here. The following applies to other frameworks as well. It *cannot* be handled on a framework level. The responsibility lies with the app developer. That's **YOU**.

DEPENDENCIES

- Angular v15.0.0
- Prettier v2.8.3

```
$ npx ng new fosdem
$ cd fosdem
# Because we are talking about Angular here,
# let's fix the build
$ npm install @types/node
$ npm install merge-descriptors
$ npm install license-webpack-plugin
$ npm run build
```

```
$ npx ng new fosdem
$ cd fosdem
# Because we are talking about Angular here,
# let's fix the build
$ npm install @types/node
$ npm install merge-descriptors
$ npm install license-webpack-plugin
$ npm run build
```

Right now, we don't change anything on any file

```
$ tree dist
dist/
└── fosdem
    ├── 3rdpartylicenses.txt
    ├── favicon.ico
    ├── index.html
    ├── main.91ffdd2e12df072d.js
    ├── polyfills.451f8e5f75f526a0.js
    ├── runtime.2ad8f73bb7b39640.js
    └── styles.ef46db3751d8e999.css

1 directory, 7 files
```

Right now, we don't change anything on any file

```
$ tree dist
dist/
└── fosdem
    ├── 3rdpartylicenses.txt
    ├── favicon.ico
    ├── index.html
    ├── main.91ffdd2e12df072d.js
    ├── polyfills.451f8e5f75f526a0.js
    ├── runtime.2ad8f73bb7b39640.js
    └── styles.ef46db3751d8e999.css

1 directory, 7 files
```

ANATOMY OF A WEBPACK BUILD

A quick look into the files generated by Angular before we move on

`index.html`

This is the app shell. Containing minimal HTML5 to load CSS and reference the above JavaScript files.

`styles.[hash].css`

At this point in time it is empty. The hash is generated by Webpack

`runtime.[hash].js`

Contains the Angular runtime that parses Angular templates and manages all the dependency injection and other magic of the framework for you

`polyfills.[hash].js`

Contains extensions to the browser runtime for things Angular expects like Zone, certain Promise features or fetch

`main.[hash].js`

Mainly your code + webpack boilerplate for RxJS, Angular template parser

THE CASE

ROUTER

```
 1 // src/app/app-routing.module.ts
 2 import { NgModule } from '@angular/core';
 3 import { RouterModule, Routes } from '@angular/router';
 4
 5 const routes: Routes = [];
 6
 7 @NgModule({
 8   imports: [RouterModule.forRoot(routes)],
 9   exports: [RouterModule]
10 })
11 export class AppRoutingModule { }
```

ROUTER

```
 1 // src/app/app-routing.module.ts
 2 import { NgModule } from '@angular/core';
 3 import { RouterModule, Routes } from '@angular/router';
 4
 5 const routes: Routes = [];
 6
 7 @NgModule({
 8   imports: [RouterModule.forRoot(routes)],
 9   exports: [RouterModule]
10 })
11 export class AppRoutingModule { }
```

ROUTE DEFINITION

Partial interface

```
 1
 2 // @angular/router/index.d.ts
 3 interface Route {
 4   path?: string;
 5   pathMatch?: 'prefix' | 'full';
 6   component?: Type<any>;
 7   redirectTo?: string;
 8   canActivate?: Array<CanActivateFn | any>;
 9   children?: Routes;
10   loadChildren?: LoadChildren;
11 }
```

ROUTE DEFINITION

Partial interface

```
 1
 2 // @angular/router/index.d.ts
 3 interface Route {
 4   path?: string;
 5   pathMatch?: 'prefix' | 'full';
 6   component?: Type<any>;
 7   redirectTo?: string;
 8   canActivate?: Array<CanActivateFn | any>;
 9   children?: Routes;
10   loadChildren?: LoadChildren;
11 }
```

ROUTE DEFINITION

Partial interface

```
 1
 2 // @angular/router/index.d.ts
 3 interface Route {
 4   path?: string;
 5   pathMatch?: 'prefix' | 'full';
 6   component?: Type<any>;
 7   redirectTo?: string;
 8   canActivate?: Array<CanActivateFn | any>;
 9   children?: Routes;
10   loadChildren?: LoadChildren;
11 }
```

ROUTE DEFINITION

Partial interface

```
 1
 2 // @angular/router/index.d.ts
 3 interface Route {
 4   path?: string;
 5   pathMatch?: 'prefix' | 'full';
 6   component?: Type<any>;
 7   redirectTo?: string;
 8   canActivate?: Array<CanActivateFn | any>;
 9   children?: Routes;
10   loadChildren?: LoadChildren;
11 }
```

ROUTE DEFINITION

Partial interface

```
 1
 2 // @angular/router/index.d.ts
 3 interface Route {
 4   path?: string;
 5   pathMatch?: 'prefix' | 'full';
 6   component?: Type<any>;
 7   redirectTo?: string;
 8   canActivate?: Array<CanActivateFn | any>;
 9   children?: Routes;
10   loadChildren?: LoadChildren;
11 }
```

ROUTE DEFINITION

Partial interface

```
 1
 2 // @angular/router/index.d.ts
 3 interface Route {
 4   path?: string;
 5   pathMatch?: 'prefix' | 'full';
 6   component?: Type<any>;
 7   redirectTo?: string;
 8   canActivate?: Array<CanActivateFn | any>;
 9   children?: Routes;
10   loadChildren?: LoadChildren;
11 }
```

BEFORE ANY CHANGES

Large chunk of boilerplate bloat before starting with implementation

```
function FR(e, t) {
1 & e && (C(0, "pre"), Q(1, "ng generate component xyz"), I())
```

GENERATING COMPONENTS

```
$ ng generate component page-not-found
$ ng generate component speaker # To be protected
$ ng generate component slides
```

No changes on build (tree-shaking)

DECLARING ROUTES

```
// src/app/app-routing.module.ts
/* Imports from above plus additionally */
import { SlidesComponent } from './slides/slides.component';
import { SpeakerComponent } from './speaker/speaker.component'

const routes: Routes = [
  { path: 'slides', component: SlidesComponent },
  { path: 'speaker', component: SpeakerComponent },
];

/* Continue as above */
```

DECLARING ROUTES (CONTINUED)

```
return new (t || ui)();
      }),
        (ui.ɵcmp = Xn({
          type: ui,
          selectors: [["app-slides"]],
          decls: 2,
          vars: 0,
          template: function (t, n) {
            1 & t && (C(0, "p"), Q(1, "slides works!"), I()
          },
        }));
      class li {}
      (li.ɵfac = function (t) {
        return new (t || li)();
      }),
```

DECLARING ROUTES (CONTINUED)

```
13        (li.ɵfac = function (t) {
14          return new (t || li)();
15        }),
16          (li.ɵcmp = Xn({
17            type: li,
18            selectors: [["app-speaker"]],
19            decls: 2,
20            vars: 0,
21            template: function (t, n) {
22              1 & t && (C(0, "p"), Q(1, "speaker works!"), I(
23            },
24          }));
25        const kR = [
26          { path: "slides", component: ui },
27          { path: "speaker", component: li },
```

DECLARING ROUTES (CONTINUED)

```
16          (li.ɵcmp = xn({
17            type: li,
18            selectors: [["app-speaker"]],
19            decls: 2,
20            vars: 0,
21            template: function (t, n) {
22              1 & t && (C(0, "p"), Q(1, "speaker works!"), I(
23            },
24          }));
25        const kR = [
26          { path: "slides", component: ui },
27          { path: "speaker", component: li },
28        ];
29        class Wr {}
30        function LR(e, t) {
```

ADDING CATCH ALL ROUTES

```
// src/app/app-routing.module.ts
/* Imports from above plus additionally */
import { PageNotFoundComponent } from './page-not-found/page-n

const routes: Routes = [
  { path: 'slides', component: SlidesComponent },
  { path: 'speaker', component: SpeakerComponent },
  { path: '', redirectTo: '/slides', pathMatch: 'full' },
  { path: '**', component: PageNotFoundComponent },
];

/* Continue as above */
```

REPLACE app.component.html

```
<!-- Remove everything inside -->
<div class="content" role="main"><!-- *snip * --></div>
<!-- Use this instead -->
<main>
  <h1>{{ title }} app is running</h1>
  <nav>
    <ul>
      <li><a routerlink="slides">Slides</a></li>
      <li><a routerlink="speaker">Speaker</a></li>
    </ul>
  </nav>
</main>
```

CHILD ROUTES WITHOUT GUARD

Add `FormsModule` to the `imports` in the `AppModule`

Usually go with Reactive forms for more advanced behaviour

```
// src/app/speaker/speaker.component.ts
class Auth {
  public password = '';
}

/* @Component decorator here */
export class SpeakerComponent {
  public model = new Auth();
}
```

CHILD ROUTES WITHOUT GUARD (CONTINUED)

```
<form #myForm="ngForm">
  <label>
    Enter the secret password to access special content:
    <input
      #password="ngModel"
      name="password"
      type="password"
      [(ngModel)]="model.password"
      required
    />
  </label>

  <a *ngIf="myForm.valid" routerLink="./slides">
    Access speaker slides
    </a>
```

CHILD ROUTES WITHOUT GUARD (CONTINUED)

```
<form #myForm="ngForm">
  <label>
    Enter the secret password to access special content:
    <input
      #password="ngModel"
      name="password"
      type="password"
      [(ngModel)]="model.password"
      required
    />
  </label>

  <a *ngIf="myForm.valid" routerLink="./slides">
    Access speaker slides
    </a>
```

CHILD ROUTES WITHOUT GUARD (CONTINUED)

```
 3     Enter the secret password to access special content.
 4     <input
 5       #password="ngModel"
 6       name="password"
 7       type="password"
 8       [(ngModel)]="model.password"
 9       required
10     />
11   </label>
12
13   <a *ngIf="myForm.valid" routerLink="./slides">
14     Access speaker slides
15     </a>
16 </form>
17 <router-outlet></router-outlet>
```

CHILD ROUTES WITHOUT GUARD (CONTINUED)

```
3       Enter the secret password to access special content:
4       <input
5         #password="ngModel"
6         name="password"
7         type="password"
8         [(ngModel)]="model.password"
9         required
10      />
11    </label>
12
13    <a *ngIf="myForm.valid" routerLink="./slides">
14      Access speaker slides
15      </a>
16  </form>
17  <router-outlet></router-outlet>
```

CHILD ROUTES WITHOUT GUARD (CONTINUED)

```
// src/app/app-routing.module.ts
/* Same as before */
const routes: Routes = [
  { path: 'slides', component: SlidesComponent },
  {
    path: 'speaker',
    component: SpeakerComponent,
    children: [{ path: 'slides', component: SlidesComponent }]
  },
  { path: '', redirectTo: '/slides', pathMatch: 'full' },
  { path: '**', component: PageNotFoundComponent },
];
/* Keep as before */
```

LAZY-LOADING CHILD ROUTES

```
$ ng generate module speaker --route speaker --module app.modu
```

```
// src/app/app-routing.module.ts
/* Remove SpeakerComponent import */
const routes: Routes = [
  {
    path: 'speaker',
    loadChildren: () => import('./speaker/speaker.module').the
  }
];
/* Continue as above */
```

LAZY-LOADING CHILD ROUTES (CONTINUED)

```
// src/app/speaker/speaker-routing.module.ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

import { SlidesComponent } from '../slides/slides.component';

import { SpeakerComponent } from './speaker.component';

const routes: Routes = [
  {
    path: '',
    component: SpeakerComponent,
    children: [{ path: 'slides', component: SlidesComponent }]
  }
];
```

LAZY-LOADING CHILD ROUTES (CONTINUED)

Remove `SpeakerComponent` from `src/app/app.module.ts`

LAZY-LOADING CHILD ROUTES (CONTINUED)

LAZY-LOADING CHILD ROUTES (CONTINUED)

Observe new build artifacts being generated

```
Lazy Chunk Files | Names                    |  Raw Size
[hash].[hash].js | speaker-speaker-module |   5.83 kB |
```

WRITING A GUARD

Generate a new guard with the CLI (Command Line Interface).

```
$ ng g guard CanActivateSpeaker
```

Use proper Permissions implementation below

```
// src/app/can-activate-speaker.guard.ts
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

export class UserToken {}
export class Permissions { canActivate(currentUser: UserToken, id: unknown): boolean { return true; } }

@Injectable({
  providedIn: 'root'
})
export class CanActivateSpeakerGuard implements
```

WRITING A GUARD

Generate a new guard with the CLI (Command Line Interface).

```
$ ng g guard CanActivateSpeaker
```

Use proper Permissions implementation below

```
// src/app/can-activate-speaker.guard.ts
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, UrlTree }
from '@angular/router';
import { Observable } from 'rxjs';

export class UserToken {}
export class Permissions { canActivate(currentUser:
UserToken, id: unknown): boolean { return true; } }

@Injectable({
  providedIn: 'root'
})
export class CanActivateSpeakerGuard implements
```

WRITING A GUARD

Generate a new guard with the CLI (Command Line Interface).

```
$ ng g guard CanActivateSpeaker
```

Use proper Permissions implementation below

```
 9 @Injectable({
10   providedIn: 'root'
11 })
12 export class CanActivateSpeakerGuard implements
   CanActivate {
13   constructor(
14     private permissions: Permissions,
15     private currentUser: UserToken
16   ) { }
17   public canActivate (route: ActivatedRouteSnapshot
18     ): Observable<boolean | UrlTree> | Promise<boolean |
   UrlTree> | boolean | UrlTree {
19       return
```

WRITING A GUARD

Generate a new guard with the CLI (Command Line Interface).

```
$ ng g guard CanActivateSpeaker
```

Use proper Permissions implementation below

```
export class UserToken {}
export class Permissions { canActivate(currentUser:
UserToken, id: unknown): boolean { return true; } }

@Injectable({
  providedIn: 'root'
})
export class CanActivateSpeakerGuard implements
CanActivate {
  constructor(
    private permissions: Permissions,
    private currentUser: UserToken
  ) { }
  public canActivate (route: ActivatedRouteSnapshot
```

WRITING A GUARD

Generate a new guard with the CLI (Command Line Interface).

```
$ ng g guard CanActivateSpeaker
```

Use proper Permissions implementation below

```
11 })
12 export class CanActivateSpeakerGuard implements
   CanActivate {
13   constructor(
14     private permissions: Permissions,
15     private currentUser: UserToken
16   ) { }
17   public canActivate (route: ActivatedRouteSnapshot
18     ): Observable<boolean | UrlTree> | Promise<boolean |
   UrlTree> | boolean | UrlTree {
19       return
   this.permissions.canActivate(this.currentUser,
   route.params['id']);
20   }
```

WRITING A GUARD

Generate a new guard with the CLI (Command Line Interface).

```
$ ng g guard CanActivateSpeaker
```

Use proper Permissions implementation below

```
11 })
12 export class CanActivateSpeakerGuard implements
   CanActivate {
13   constructor(
14     private permissions: Permissions,
15     private currentUser: UserToken
16   ) { }
17   public canActivate (route: ActivatedRouteSnapshot
18     ): Observable<boolean | UrlTree> | Promise<boolean |
   UrlTree> | boolean | UrlTree {
19       return
   this.permissions.canActivate(this.currentUser,
   route.params['id']);
20   }
```

CHILD ROUTE WITH GUARD

```
// src/app/app-routing.module.ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { CanActivateSpeakerGuard, Permissions, UserToken }

const routes: Routes = [
  {
    path: 'speaker',
    canActivate: [CanActivateSpeakerGuard],
    loadChildren: () => import('./speaker/speaker.module').
  },
  /* Same as before */
];

/* Added provider! */
```

CHILD ROUTE WITH GUARD

```
 7    {
 8      path: 'speaker',
 9      canActivate: [CanActivateSpeakerGuard],
10      loadChildren: () => import('./speaker/speaker.module').
11    },
12    /* Same as before */
13  ];
14
15  /* Added provider! */
16  @NgModule({
17    imports: [RouterModule.forRoot(routes)],
18    exports: [RouterModule],
19    providers: [CanActivateSpeakerGuard, Permissions, UserTok
20  })
21  export class AppRoutingModule {}
```

THE REMEDY

This is the part where I would like to use Webpack's named chunks. (https://webpack.js.org/api/module-methods/#magic-comments)

But Angular does not support them. (https://github.com/angular/angular-cli/issues/16697)

THE REMEDY

This is the part where I would like to use Webpack's named chunks. (https://webpack.js.org/api/module-methods/#magic-comments)

But Angular does not support them. (https://github.com/angular/angular-cli/issues/16697)

THE REMEDY (CONTINUED)

The idea being to protect that specific chunk with HTTP headers.

Speaking of, the Security guide on Content Security Policy (https://angular.io/guide/security#content-security-policy) declares that at the very minimum Angular requires

```
default-src 'self'; style-src 'self' 'unsafe-inline';
```

You can still apply SHA hashes or nonces with some effort for protection (https://stackoverflow.com/a/68460908)

THE REMEDY (CONTINUED)

If you validate the password, don't list forbidden passwords in Angular. Otherwise these entries will be excluded from Credential stuffing attacks (https://owasp.org/www-community/attacks/Credential_stuffing)

In a similar vein load password classes (length, special characters) asynchronously to make criminals' life harder

Best to check passwords on the server and display validation errors from the response

THE REMEDY (CONTINUED)

At least guarding paths protect them from being listed but not from being cURLed. Therefore they can still be enumerated for attacks.

Load more data **after** a successful login. Check the `Authorization` header on the server!

WHAT HAVE YOU LEARNED TODAY?

- Understanding webpack bundles
- Named chunks in Angular builds
- Content Security Policy options
- More options to secure static files

IMAGE CREDITS

Thank you