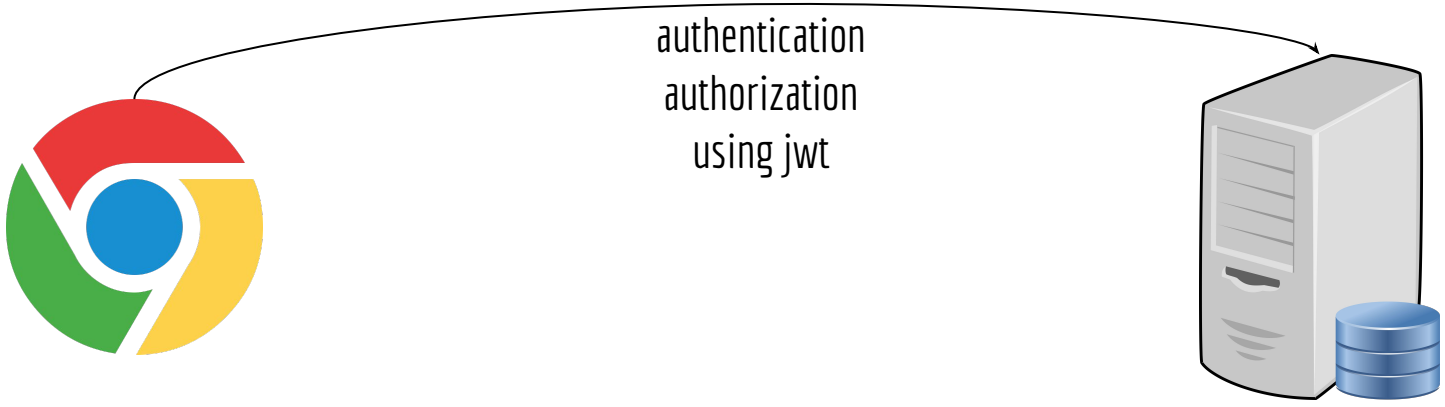
Authentication mechanism using JWT

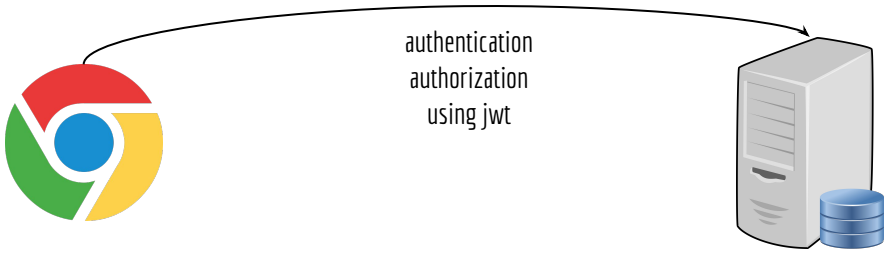
By Fanis Prodromou

Agenda

1. What is JWT
2. Guards
3. Authentication with Guards
4. Authorization with Guards
5. HTTP Interceptors

The Problem


authentication
authorization
using jwt



User_X is Authenticated

User_X is Authorized to access **Page_A**

User_Y is **NOT** Authorized to access **Page_A**

What is JWT

JWT stands for **J**SON **W**eb **T**oken

A compact and self-contained way for securely transmitting information between parties

JWT

xxxxx.yyyyy.zzzzz

- Header
- Payload
- Signature

Header

Two properties

1. Hashing algorithm being used
2. Type of the token

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Payload

Contains the claims.

Claims are statements about an entity (typically, the user)

```
{
  "username": "profanis",
  "first": "Fanis",
  "last": "Prodromou"
}
```

Signature

Recipe

1. Encoded header
2. Encoded payload
3. A secret
4. The algorithm specified in the header
5. Sign that
6. Enjoy :)

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

Encoded PASTE A TOKEN HERE


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
1c2VybmFtZSI6IlByb2ZhbmlzIiwiZmlyc3QiOiJ
mYW5pcyIsImxhc3QiOiJwcm9kcm9tb3UiLCJpZCI
6MTIzNDU2Nzg5MCwicm9sZXMiOlsiYWRtaW4iXX0
.BYCbmcm1hPJ-F8jwVyPpmJsXl-B0jbH6B-
zLz3d1Aog
```


Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```


PAYLOAD: DATA


```
{
  "username": "Profanis",
  "first": "fanis",
  "last": "prodromou",
  "id": 1234567890,
  "roles": [
    "admin"
  ]
}
```


VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) secret base64 encoded
```

Decode the token

Decode

1. Using a ready library
   a. @auth0/angular-jwt
   b. jwt-decode
   c. jsonwebtoken
2. Custom

```
getDecodedToken(token: string) {
  const tokenPayload = token.split(".")[1];
  return JSON.parse(base64.Base64.decode(tokenPayload));
}
```

Authentication

Procedure

1. **HTTP** call

2. Set the token in **localStorage**

3. On each route check if the user is authenticated

Auth form

Username

Enter username

Password

Enter password

Submit

HTTP Call

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { tap } from 'rxjs/operators';

@Injectable()
export class LoginService {

  constructor(private http: HttpClient) {}

  login(username, password) {
    return this.http.post("/login", {username, password}).pipe(
      tap((data: any) => localStorage.setItem("token", data))
    );
  }
}
```

Guards

Type of Guards

- **CanActivate**
  Validates whether the user can visit the route or not

- **CanDeactivate**
  Validates whether the user can leave a route or not

- **CanActivateChild**
  Validates whether the user can visit a routes children or not

- **CanLoad**
  Validates whether the user can visit a lazy load module

CanActivate Guard

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

  constructor (private router: Router,
               private userService: UserService,
               private toastr: ToastrService) {}

  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): boolean {

      if(!this.userService.isLoggedIn) {
        this.toastr.warning("You are not authenticated");
        this.router.navigate(["/auth"]);
      }

    return this.userService.isLoggedIn;
  }
```

Use the CanActivate Guard

```
const routes: Routes = [
  {path: "hotels",
  component: HotelListComponent,
  canActivate: [AuthenticatedGuard]}
];
```

Authentication on the Server Side

Always send an **http header** with the **token**

HTTP Call With Headers

```
@Injectable()
export class HotelService {

  constructor(private http: HttpClient) { }

  getHotels() {
    return this.http.get("/hotel", {
      headers: {"Authorization": localStorage.getItem("token")}
    });
  }
}
```

Authorization

CanActivate Guard

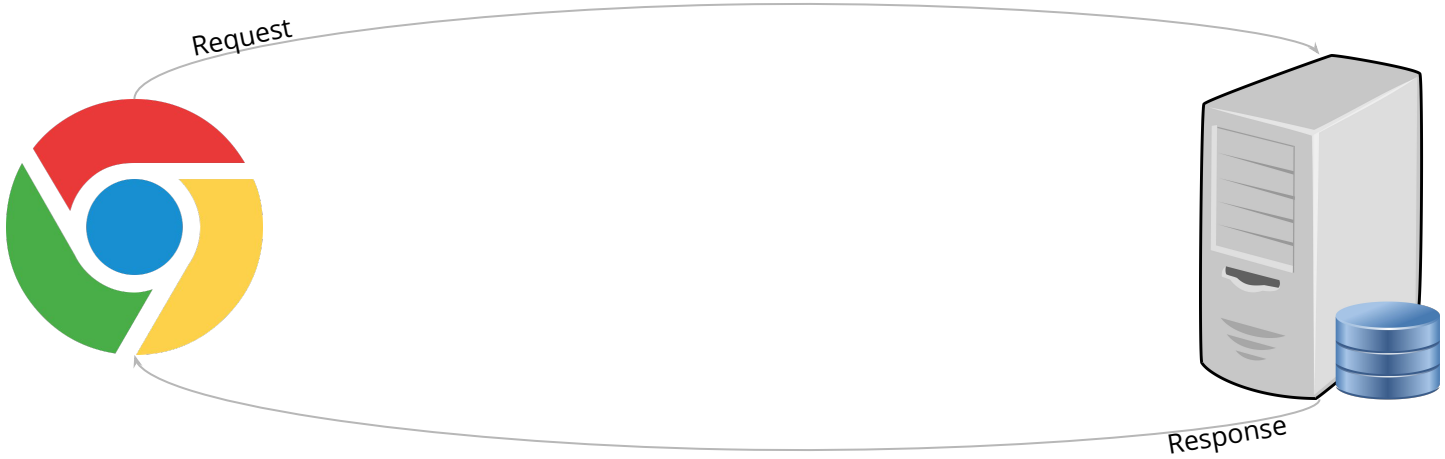
```
@Injectable({providedIn: 'root'})
export class IsAdminGuard implements CanActivate {

  constructor (private router: Router, private userService: UserService, private
toastr: ToastrService) {}

  canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {

      const isAdmin = this.userService.roles.includes("admin");

      if(!isAdmin) {
        this.toastr.warning("You are not authorized to access this page");
        this.router.navigate(["/auth"]);
      }

    return isAdmin;

  }
}
```

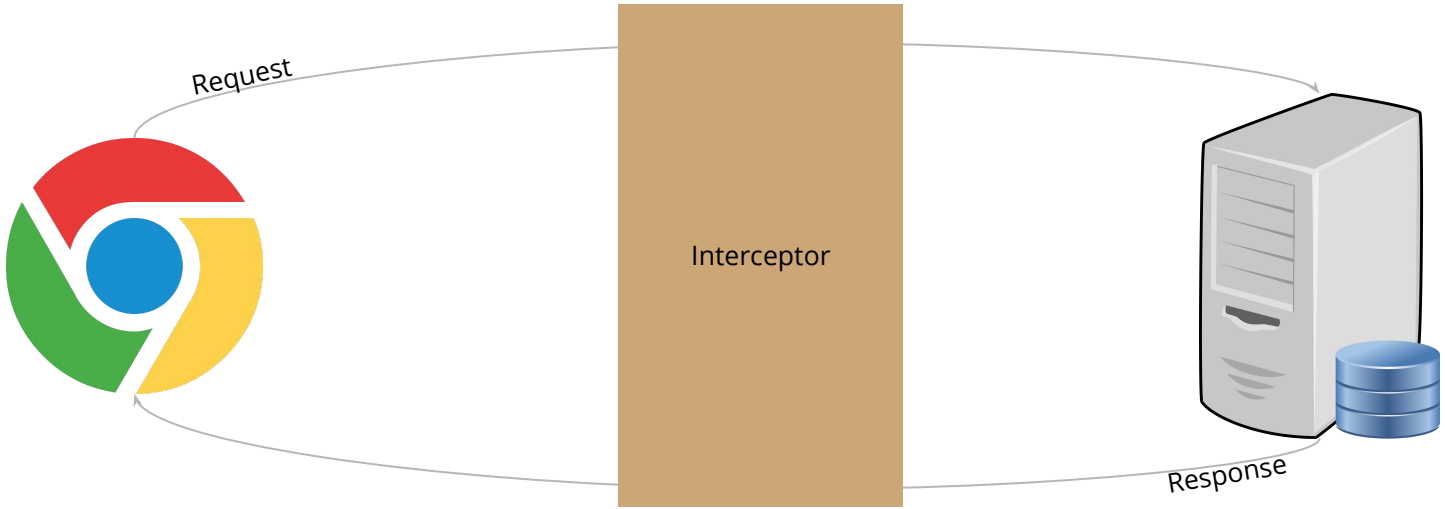
Use the CanActivate Guard

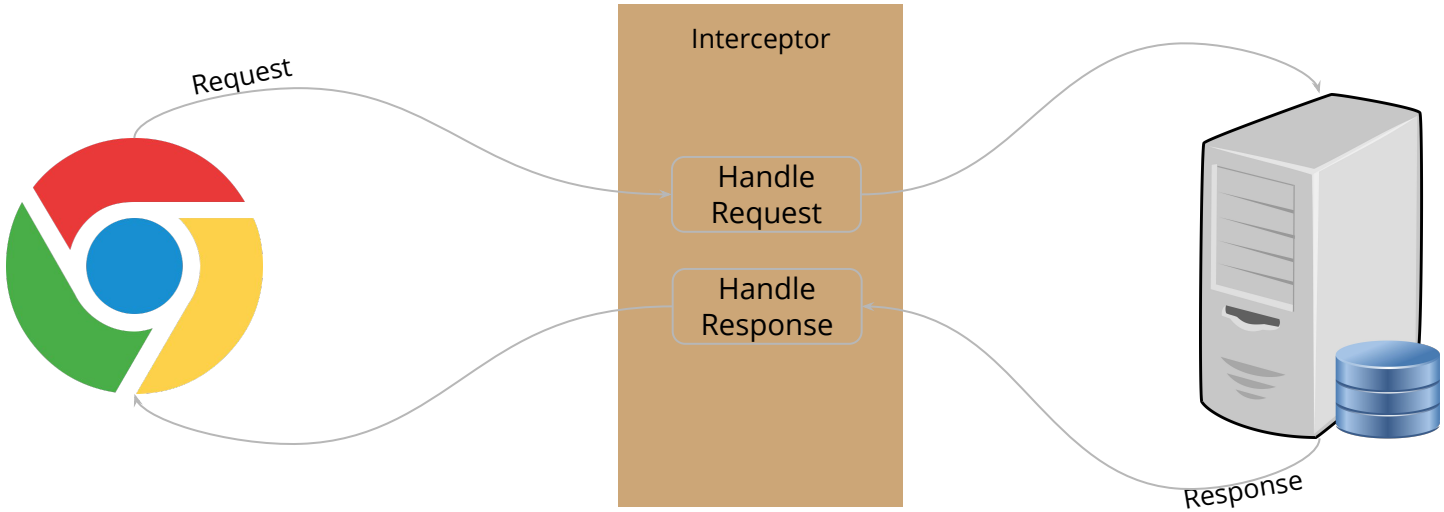
```
const routes: Routes = [
  {path: "hotels",
  component: HotelListComponent,
  canActivate: [AuthenticatedGuard, IsAdminGuard]}
];
```

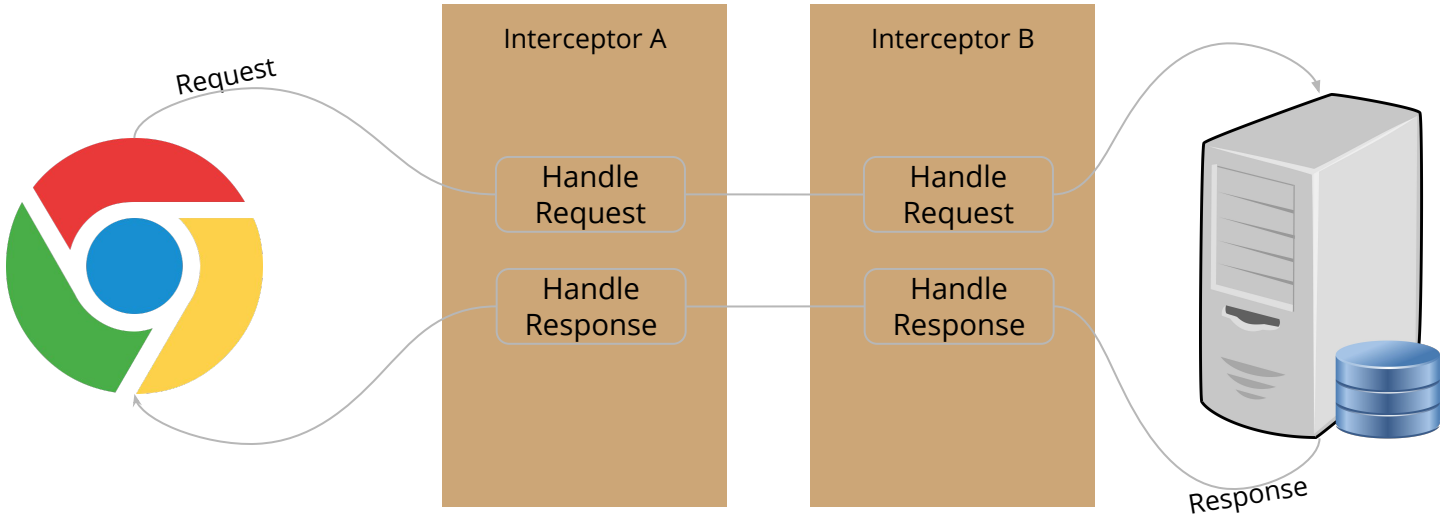
HTTP Interceptor

For the rescue of DRY :)


Request
Response


Request
Interceptor
Response


Request
Interceptor
Handle Request
Handle Response
Response


Request
Interceptor A
Handle Request
Handle Response
Interceptor B
Handle Request
Handle Response
Response

HTTP Interceptor - Snippet

```
import { Injectable } from '@angular/core';
import { HttpInterceptor,
         HttpEvent,
         HttpHandler,
         HttpRequest } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable()
export class HeaderInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>
    return next.handle(req);
  }
}
```

HTTP Interceptor - Snippet

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [
    {
      provide: HTTP_INTERCEPTORS,
      useClass: HeaderInterceptor,
      multi: true
    }
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

HTTP Interceptor - Handle Response

```
@Injectable()
export class HeaderInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>
    return next.handle(req).pipe(
      map((event: HttpEvent<any>) => {
        if (event instanceof HttpResponse) {
          if (event.status === 200) {
            console.log("I am 200");
          }
        }
        return event;
      })
    )
  }
}
```

1. What is JWT
2. Guards
3. Authentication with Guards
4. Authorization with Guards
5. HTTP Interceptors

Q & A

Thank you!!

https://www.linkedin.com/in/prodromouf/