

Motivation

Ausgangsbasis

- Liste von allen Todos

Wunsch

- Back-End-Aufrufe programmieren
- Back-End-Aufrufe konform der Schnittstelle konfigurieren
- Daten eines Back-End-Aufrufs verarbeiten

Lernziele

- Sie beschreiben die Bedeutung und den Nutzen des HttpClient-Modules
- Sie benennen die unterstützten HTTP-Methoden
- Sie benennen die Konzepte, die für die Programmierung von Back-End-Aufrufen zur Verfügung stehen, und beschreiben deren Bedeutung
- Sie beschreiben, wie Back-End-Aufrufe konfiguriert werden
- Sie erklären, auf welche Weise die Daten solcher Aufrufe in der Webanwendung weiter verwendet werden

Kommunikation mit dem Server

Situation

- Fehlende Daten müssen nachgeladen werden
- Nutzereingaben sollen in einer Datenbank auf dem Server gespeichert werden

Theorie

- HTTP → Client- (Browser)/Server-Kommunikation
 - Browser stellt XMLHttpRequest (XHR)
 - XHR bildet Grundbaustein für Ajax-Technik

Kommunikation mit dem Server

Umsetzung

Http-Module (bis Version 4.2.6)

- Wird hier nicht weiter behandelt

HttpClient-Module (ab Version 4.3.0)

- Baut auf XHR und Fetch API auf
- Grundlage für HTTP-Services / REST-Services

Module

HttpClientModule from '@angular/common/http'

HTTP-Service

- Implementierung eines REST-Clients für spezifische Ressource
- Entspricht einer TypeScript-Klasse
- Stellt passende Funktionen für typische HTTP-Verben wie GET, POST, PUT, DELETE zur Verfügung

HTTP-Service

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

@Injectable({ providedIn: 'root'})
export class HeroService {
    private heroesUrl = 'api/heroes'; // URL to web api

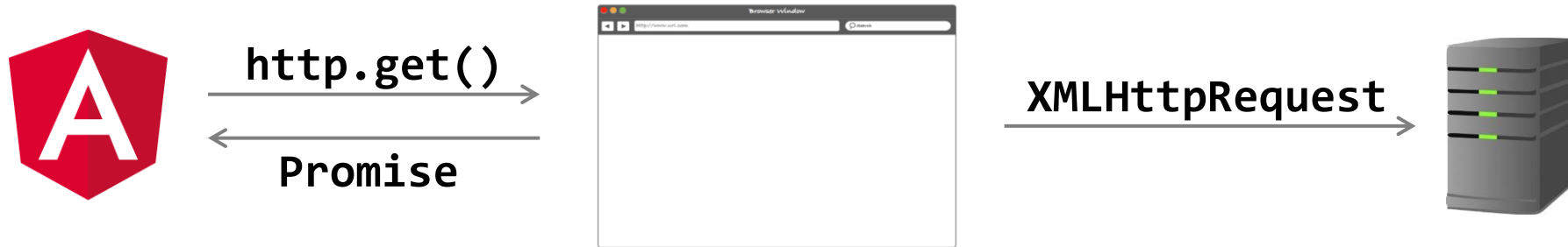
    constructor(private http: HttpClient) { }

    getHeroes() {
        return this.http.get(this.heroesUrl)
    }
}
```

Umgang mit Rückgabe (Response)

- Response → durch HttpClient ein JSON-Objekt
- AJAX = **Asynchronous** JavaScript and XML
- Observable
 - Angular HttpClient-Module hat als Rückgabe den Typ Observable
 - Observable = Asynchroner Stream von Events (mit Daten)
 - **Tipp:** Umwandlung in Promises möglich

Intermezzo: Promises und Asynchronität



Promise
[1,2,3,5,8]
+ then + catch

```
this.http.get(endPointUrl)  
  .then( processData )  
  .then( displayData )  
  .catch( handleErrors )
```


HTTP-Service

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

@Injectable()
export class HeroService {
  private heroesUrl = 'api/heroes'; // URL to web api

  constructor(private http: HttpClient) { }

  getHeroes() : Promise<Hero[]> {
    return firstValueFrom(this.http.get<Hero[]>(this.heroesUrl))
  }
}
```

HTTP-Service verwenden

```
import { HeroService } from '../hero.service';  
  
export class HeroListComponent implements OnInit {  
  
    heroes: Hero[];  
    constructor( private heroService: HeroService) { }  
  
    ngOnInit() {  
        this.heroService.getHeroes()  
            .then(heroes => this.heroes = heroes);  
    }  
}
```

} Service in Komponente importieren

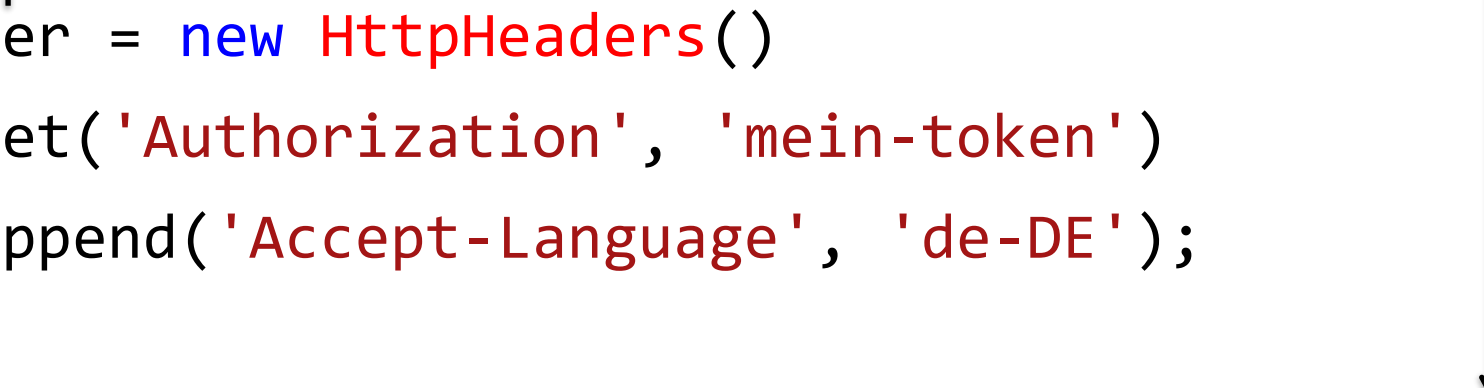
} Service Initiiieren

} Service sendet Request ab und aktualisiert heroes

HTTP konfigurieren

■ Header

```
header = new HttpHeaders()  
    .set('Authorization', 'mein-token')  
    .append('Accept-Language', 'de-DE');  
  
firstValueFrom(http.post(this.heroesUrl, body, {headers: header}));
```

A diagram consisting of a horizontal line that starts at the 'header' variable in the first line of code, extends to the right, and then turns downward into an arrow pointing to the 'headers: header' property in the third line of code.

HTTP konfigurieren

■ URL Parameter

```
parameter = new HttpParams()  
    .set('name', 'Han')  
    .append('sort', 'asc');
```

```
firstValueFrom(http.get(this.heroesUrl, {params:  
parameter}))
```

■ Aufgerufene URL

```
'api/heroes?name=Han&sort=asc'
```

Zusammenfassung

Um was ging es in diesem Modul?

- Überblick über das HTTP-Modul
- Back-End-Aufrufe konfigurieren und absenden
- Möglichkeit und Nutzen für die Implementierung von Rest-Services

Wozu brauche ich das? Was will ich damit machen?

- Implementierung von neuen Back-End-Aufrufen
- Back-End-Aufrufe konform der Schnittstelle konfigurieren
- Verarbeitung von Daten eines Back-End-Aufrufs

Kontrollfragen

- Mit welcher Syntax werden URL-Parameter definiert?
- Welche Bedeutung hat das HTTP-Module?
- Welche Methoden stellt die HTTP-Klasse zur Verfügung?
- Wie werden HTTP-Anfragen konfiguriert (z. B. Header)?
- Auf welche Weise werden die Daten solcher Aufrufe in der Webanwendung weiter verwendet?

