JavaScript

Trinaesti dio

Pregled

- Fetch API
  - Uvod
  - Uvodni primjer
  - Zadatak (PP)
  - Response objekat
  - Request objekat
  - Header
  - Request options
  - Aplikacija, TMDB, API key
  - Alternative

Obnavljanje

- Promises
  - Kako se kreira
  - Koja je glavni razlog uvođenja
  - Metodi - navesti neke i opisati ih

Fetch, uvod

- Fetch API omogućava JSu da pristupi i manipuliše HTTP tokom kao što su zahtjevi ka serveru i odgovori sa servera. Takođe, fetch() metod omogućava jednostavan, logičan pristup za asinhrono prihvatanje/slanje resursa putem mreže
- Ovo se nekad radilo putem XMLHTTPRequest objekta.
- Neke od prednosti Fetch-a u odnosu na XMLHTTPRequest:
    - Jednostavnost
    - Obezbjeđuje da cjelokupnu HTTP logiku kao što je CORS, hederi, itd. definišemo na jednom mjestu
- CORS (cross origin resource sharing) služi za definišemo koji izvor (najčešće po IP adresi) može da pristupi serveru. Ko bude pratio backend dio, vidjeće detalje vezane za CORS
- Sintaksa
    - **fetch(url, options)**
        - url - definišemo url do resursa
        - options - definišemo opcije request-a ka serveru

Fetch, uvodni primjer

- Pogledajmo prvi primjer poziva **fetch()** metoda

```
fetch('http://example.com/movies.json')
  .then(function(response) {
    return response.json();
  })
  .then(function(myJson) {
    console.log(JSON.stringify(myJson));
  });
```

- U ovom primjer možete vidjeti kako biste učitali JSON fajl sa servera. Najjednostavnija verzija fetch() metoda je sa jednim argument - taj argument predstavlja putanja do resura koji tražimo.
- Metod fetch() **vraće Promise** objekat koji sadrži odgovor sa servera (Response objekat) - o Response objektu govorićemo u nastavku

Fetch, uvodni primjer, nastavak(1)

- Reponse objekat predstavlja HTTP Response, ne stvarni JSON koji smo tražili.
- Pogledajmo u konzoli pretraživača kako izgleda response ako je vrijednost argument fetch() funkcije https://api.github.com/users
- Uporedimo ga sa Response u Network tabu
- Da bismo izvukli JSON sadržaj iz odgovora pozivamo json() metod nad Response objektom (objektom koji nam je vratio server ako je uspješno obradio zahtjev)
    - Nalazi se u Body dijelu
- Pogledajmo još jedan primjer bez options argumenta (code-10)

Zadatak

- Zadatak:
  - Kreirati mini aplikaciju koja prikazuje listu follow-era unijetog korisnik u tabeli koja sadrži tri kolone i to: username (login) , avatar (avatar_url) i link (html_url). Ako korisnik kog ste tražili ne postoji štampati poruku: Korisnik ne postoji. Kući dodati pagination (straničenje)

Response objekat

- Objekat koji sadrži Response sa server ka kome je poslat request
- Kreira se na sledeći način: **new Response()**
- Sadrži sledeća svojstva:
  - **headers** - sadrži header objekat povezan sa odgovorom
  - **ok** - sadrži boolean vrijednost koja predstavlja da li je odgovor uspješan (status range 200-299) ili ne
  - **redirected** - sadrži informaciju o tome da li je odgovor stigao kao redirect
  - **status** - sadrži status kod odgovora (npr. 200 za success)
  - **statusText** - sadrži status poruku za odgovarajući status kod (npr. OK za 200)
  - **type** - sadrži informaciju o tipu odgovora (npr. basic, cors)
  - **url** - sadrži URL odgovora
  - **useFinalURL** - sadrži boolean koji nam govori da li je finalni URL odgovora
- Kako Response implementira **Body**, takođe može da koristi sledeća svojstva
  - **body** - učitava stream podataka
  - **bodyResponse**

Response objekat, metode

- Neki od metoda Response objekta:
  - **clone**() - klonira Response objekat
  - **error**() - vraće novi Response objekat sa odgovarajućom greškom
  - **redirect**() - kreira novi response sa drugim URLom
- Kako Request implementira **Body**, takođe može da koristi sledeća svojstva:
  - **arrayBuffer**() - uzima Response stream i čita ga. Vraće promise koji se resolvuje u ArrayBuffer
  - **blob**() - uzima Response stream i čita ga. Vraće promise koji se resolvuje u Blob
  - **formData**() - uzima Response stream i čita ga. Vraće promise koji se resolvuje u FormData objekat (vidjećemo primjer)
  - **json**() - uzima Response stream i čita ga. Vraća promise koji se resolvuje u tekst parsiran u JSON
  - **text**() - uzima Response stream i čita ga. Vraće promise koji se resolvuje u tekst

Request objekat

- Predstavlja resource request
- Kreira se sa: **new Request()**
- Sadrži sledeća svojstva:
    - **cache** - sadrži cache mode zahtjeva (npr. default, reload, no-cache)
    - **context** - sadrži context zahtjeva (npr. audio, image, iframe)
    - **credentials** - sadrži credentials zahtjeva (omit, **same-origin**, include)
    - **destination** - vraće string opisujući request destinaciju. Ovaj string ukazuje na tip sadržaja koji se traži
    - **headers** - sadrži odgovarajuće header-e zahtjeva
    - **integrity** - sadrži subresource integrity (omogućava pretrađivaču da verifikuje integritet resursa)
    - **method** - metod request-a (GET, POST, PUT, DELETE, OPTIONS, etc.)
    - **mode** - sadrži mode zahtjeva (cors, no-cors, same-origin, navigate)
    - **redirect** - sadrži mode kad redirect treba da bude obrađen (follow, error, manual)
    - **referrer** - sadrži referrer zahtjeva (npr. client)
    - **referrerPolicy** - referrer policy zahtjeva (npr. no-referrer)
    - **url** - sadrži url zahtjeva
- Ima iste metode kao i Response objekat

Header

- Služi za obavljanje raznih akcija nad HTTP request i response hederima
- Neki od metoda hedera
  - **append**() - dodaje novi vrijednost na postojeći header objekat ili dodaje header ako ne postoji
  - **delete**() - briše header iz Header objekta
  - **entries**() - vraće iterator koji omogućava prolazak kroz sve key/value parove unutar header objekta
  - **get**() - vraće ByteString sekvencu svih vrijednosti u header objektu sa zadatim imenom
  - **has**() - vraće boolean koji nam govori da li Header objekat sadrži odgovarajući header
  - **keys**() - vraće iterator koji omogućava prolazak kroz sve ključeve header objekta
  - **set**() - postavlja novu vrijednost postojećeg headera u Header objektu ili dodaji novi ako ne postoji
  - **values**() - vraće iterator koji omogućava prolazak kroz sve vrijednosti header objekta
- Za više informacija o header-ima pročitati na: link

Header, primjeri

```
var myHeaders = new Headers();

myHeaders.append('Content-Type', 'text/xml');
myHeaders.get('Content-Type') // should return 'text/xml'
```

```
var myHeaders = new Headers({
    'Content-Type': 'text/xml'
});

// or, using an array of arrays:
myHeaders = new Headers([
    ['Content-Type', 'text/xml']
]);

myHeaders.get('Content-Type') // should return 'text/xml'
```

Fetch, definisanje opcija za request

- Metod fetch() ima drugi argument, koji je opcioni, a predstavlja kontrolisanja raznih parametara.
- Sada ćemo kroz jedan primjer testirati razne opcije
  - **method** - GET, POST, PUT, DELETE, itd
  - **mode** - same-origin, cors, no-cors
  - **cache** - default. No-cache, reload, force-cache, only-if-cached
  - **credentials** - same-origin, include, omit
  - **headers** - header objekat
  - **redirect** - follow, manual, error
  - **referrer** - client, no-referrer
  - **body** - content (body data type isti kao “Content-Type” definisan u header objektu)

Fetch, primjer sa options argumentom

- Pogledajmo primjer sa options argumentom (code-11)
- Za više informacije pročitati sadržaj sa link-a

Aplikacija

- TMDB
  - Napraviti naloge
- API Key
  - Pronaći ključ
- Postman
- **Zadatak**: Prikazati 10 najnovijih filmova sortiranih po ocjeni od najmanjeg do najvećeg
- Kako biste sada napravili više paralelnih API poziva i znali da su svi odgovori stigli uspješno i kako biste pristupili tim podacima? Isprobati tako što ćete postali dva odvojena poziva ka dva različita APIa (npr. TMDB i IGDB).
- Kako biste štampali rezultate provog pristiglog odgovora, a prekinuli ostale API pozive
- Kako da provjerite jesu li svi pristigli odgovori resolved?
- Ograničenja
- Istražimo još neki API
  - https://api.igdb.com/
- Probajte sad da dovršite test

Alternative, Polyfill

- Third party paketi
  - JQuery Ajax
  - Axios
- XMLHTTPRequest
- Polyfill

Pitanja