Veebiarendus

Front-End arendus

Martti Raavel

martti.raavel@tlu.ee

Tänased teemad

- Meenutame eelmist loengut
- Autentimine ja autoriseerimine
- Veebilehitseja mälutehnoloogiad
- JWT
- Päiste saatmine Axiose abil
- Autentimise ja autoriseerimise rakendamine

Millest rääkisime eelmises loengus?

Autentimine ja autoriseerimine

Autentimine on protsess, millega üks kasutaja, süsteem või muu olem (objekt) saab kontrollida teise olemi väidetava identiteedi tõesust, tavaliselt mingit tüüpi identsustõendi alusel:

- miski, mida Sa tead (näiteks parool, PIN-kood, robotilõks, turvaküsimus);
- miski, mis Sul on (näiteks ID-kaart, pangakaart, telefoni number, e-mail, riistvarapääsmik, paroolikaart, sertifikaat) või
- miski, mis Sa oled (sõrmejälg, näo pilt või topograafia, iirise struktuur, ...).

Autoriseerimine

`Autoriseerimine` ehk volitamine on protsess, mis annab (või keelab) õiguse ligi pääseda (võrgu)ressurssidele. Näiteks enamik e-kaubanduse turvasüsteeme põhineb kaheastmelisel protsessil. Esiteks toimub `autentimine`, kus kontrollitakse, et kasutaja on tõesti see, kellena ta esineb, seejärel toimub `autoriseerimine`, mis lubab kasutajal juurde pääseda temale ette nähtud ressurssidele.

Kuidas toimub autentimine ja autoriseerimine API-s?

- Kasutaja tuvastamine - kuidas?
- Kasutaja õigused?
- Millal ja kuidas kasutaja õiguseid kontrollitakse?

Veebilehitseja mälutehnoloogiad

- LocalStorage
- SessionStorage
- Cookies
- IndexedDB

LocalStorage

Local Storage on püsiv mälu, mis võimaldab veebirakendustel salvestada võtme-väärtuse paare. Andmed jäävad alles ka pärast lehe uuesti laadimist või veebilehitseja sulgemist. Local Storage on kasulik kasutaja eelistuste, seansside oleku ja muude püsivate andmete salvestamiseks.

LocalStorage näide

```
// Andmete salvestamine
localStorage.setItem('key', 'value');

// Andmete lugemine
const value = localStorage.getItem('key');

// Andmete eemaldamine
localStorage.removeItem('key');

// Kõigi andmete puhastamine
localStorage.clear();
```

SessionStorage

Session Storage on ajutine mälu, mis säilitab andmeid ainult sirvimisseansi jooksul. Andmed kustutatakse pärast lehe uuesti laadimist või veebilehitseja sulgemist.

SessionStorage näide

```
// Andmete salvestamine
sessionStorage.setItem('key', 'value');

// Andmete lugemine
const value = sessionStorage.getItem('key');

// Andmete eemaldamine
sessionStorage.removeItem('key');

// Kõigi andmete puhastamine
sessionStorage.clear();
```

Küpsised (*Cookies*)

Cookies on väikesed andmeplokid, mida veebilehitseja salvestab ja saadab serverile iga kord, kui tehakse sama päritoluga päring. Küpsiseid kasutatakse sageli kasutajate autentimiseks ja seansside haldamiseks.

Küpsiste kasutamine

```
// Küpsise seadmine
document.cookie = "username=JohnDoe; expires=Fri, 31 Dec 2021 12:00:00 UTC; path=/";

// Küpsise lugemine
const cookies = document.cookie;
console.log(cookies);

// Küpsise eemaldamine (seades aegumiskuupäeva minevikku)
document.cookie = "username=JohnDoe; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";
```

IndexedDB

IndexedDB on madala taseme API, mis võimaldab salvestada märkimisväärseid andmekoguseid (sealhulgas struktureeritud objekte) kasutaja seadmesse. See on mõeldud keerukamate andmebaasioperatsioonide jaoks ja toetab indekseid ja tehinguid.

JSON Web Token (JWT)

JWT (*JSON Web Token*) on avatud standard (RFC 7519), mida kasutatakse andmete turvaliseks edastamiseks osapoolte vahel JSON formaadis. See on eriti kasulik autentimise ja informatsiooni vahetamise jaoks, kuna see võimaldab teabe kindlat ja tõhusat edastamist. JWT on kompaktne ja hõlpsasti kasutatav mitmesugustes stsenaariumides, sealhulgas veebirakenduste autentimises ja autoriseerimises.

JWT komponendid

JWT koosneb kolmest osast, mis on eraldatud punktidega (.). Need osad on:

- Header (Päis)
- Payload (Koormus)
- Signature (Allkiri)

Header (Päis)

See sisaldab teavet tokeni tüüp ja kasutatava allkirja algoritmi kohta (nt HMAC SHA256 või RSA).

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

Payload (Koormus)

See sisaldab väiteid (*claims*), mis on JWT-sse kodeeritud teave. Väited võivad olla standardiseeritud, avalikud või privaatsed.

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

Allkiri (Signature)

Allkiri genereeritakse kasutades header ja payload osi ning salajast võtit. Allkiri võimaldab kontrollida, kas JWT on kehtiv ja usaldusväärne.

JWT struktuur

JWT struktuur on järgmine:

```
xxxxx.yyyyy.zzzzz
```

kus: xxxxx on päis, yyyyy on koormus ja zzzzz on allkiri, kõik base64-url kodeeringus.

JWT näide

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

JWT kasutamine

Kuidas me saame JWT-d kasutada autentimise ja autoriseerimise protsessides?

Andmete saatmine Express API-le - Header

Sageli on vaja saata andmeid API-le, mis on seotud kasutaja autentimise ja autoriseerimisega. Kuidas seda teha?

Authorization Header

`Authorization` päises saab saata autentimiseks vajalikku teavet. Näiteks JWT tokenit.

Päiste saatmine Axiosega

Sageli tahame saata API-dele koos päringutega kaasa ka päised. Näiteks on see vajalik autentimistokeni saatmiseks, kui API eeldab autentimist. Siin on näide, kuidas saata päised Axiose abil:

```
async function fetchUser(userId) {
  try {
    const response = await axios.get(`https://jsonplaceholder.typicode.com/users/${userId}`, {
      headers: {
        Authorization: 'Bearer my-auth-token'
      }
    });
    console.log(response.data);
  } catch (error) {
    console.error('Viga päringus:', error);
  }
};

fetchUser(1);
```

Autentimise ja autoriseerimise rakendamine

Kodune töö

- Loe läbi tänase loengu materjalid
- ...