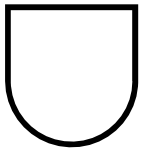
Code Academy

Gytis Juozenas

Bendraujam su back-endu: Fetch API pagrindai

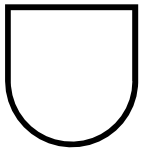
2022

JavaScript programavimo kalba



Šiandien išmoksite

01 **Bendraujam su back-endu: Fetch API pagrindai**



Fetch API

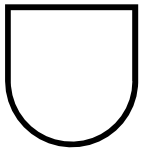
Fetch API (žiūrėkite iki 4:18)

Yra keli būdai, kaip siųsti tinklo užklausą ir gauti informaciją iš serverio. fetch() metodas yra modernus ir universalus.

Sintaksė (naudojant Promise):

```
fetch(url, options)
  .then(response => response.json())
  .then(result => /* process result */)
  .catch(error => console.log(error))
```

- *url* - prieigos URL.
- *options* - neprivalomi parametrai:
  - method - HTTP metodas (GET, POST, PUT, DELETE);
  - headers - objektas su užklausos antraštėmis ;
  - body - duomenys, kuriuos reikia siųsti (užklausos tekstas) kaip eilutė, „FormData“, „BufferSource“, „Blob“ arba „UrlSearchParams“ objektas.



Fetch API

[Fetch API]

Atsakymo savybės (response properties):

response.status – HTTP atsakymo kodas,

response.ok – true yra statusas tarp 200-299.

response.headers – Map tipo objektas su HTTP antraštėmis.

Metodai gauti atsakymo kūną/turinį (methods to get response body):

response.text() – grąžinti atsakymą kaip tekstą;

response.json() – analizuoti atsakymą kaip JSON objektą;

response.formData() – grąžinti atsakymą kaip FormData duomenų objektą;

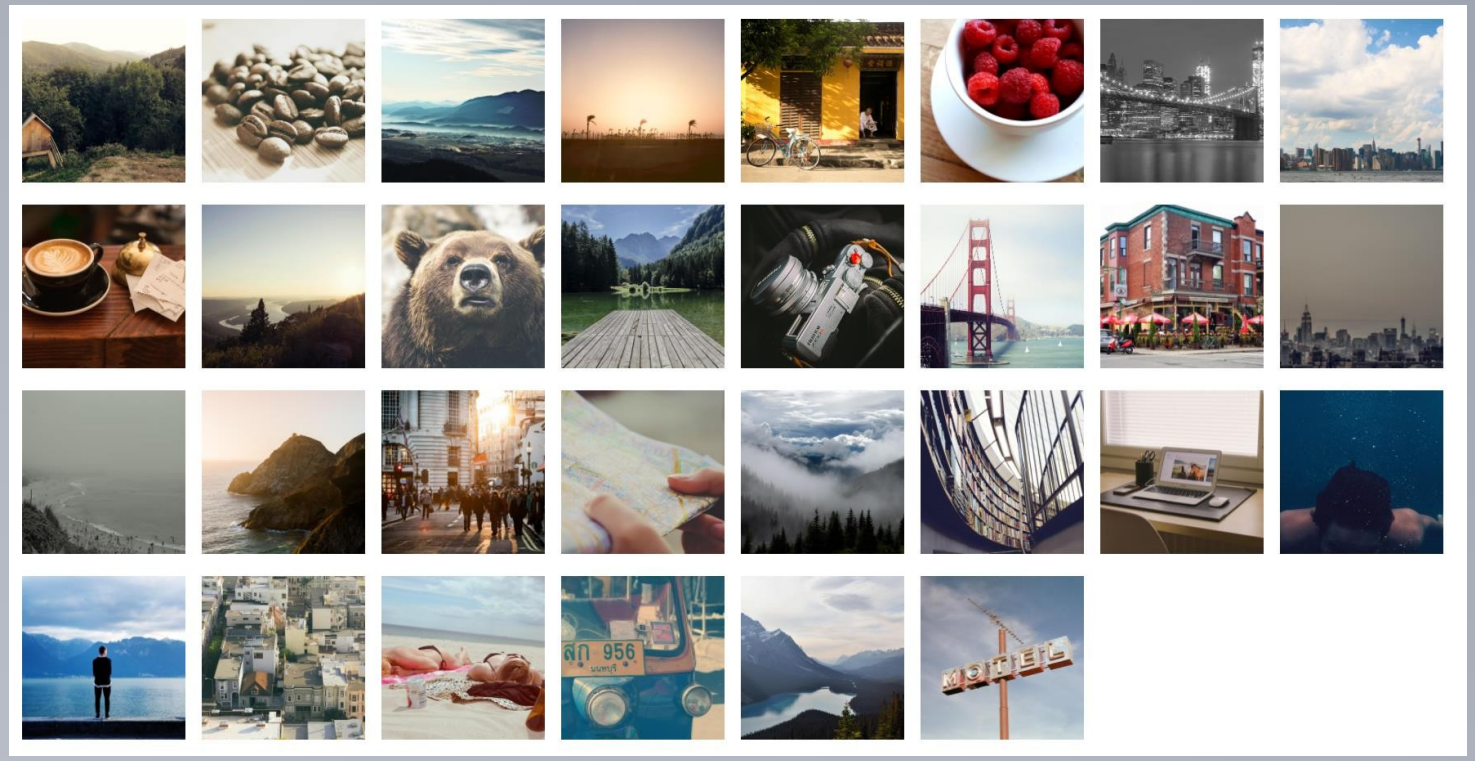
response.blob() – grąžinti atsakymą kaip Blob (dvejetainiai duomenys su tipu),

response.arrayBuffer() – grąžinti atsakymą kaip ArrayBuffer (low-level binary data),

Practice Time

Jūsų tikslas yra atvaizduoti nuotraukas ištrauktas iš API kurio link‘as yra toks: https://picsum.photos/v2/list?page= Už lygybės gali eiti skaičius nuo 1 iki 33 ir kiekvienas iš šitų variantų duos skirtingus JSON formato metodus turinčius po 30 nuotraukų. Jums reikės:

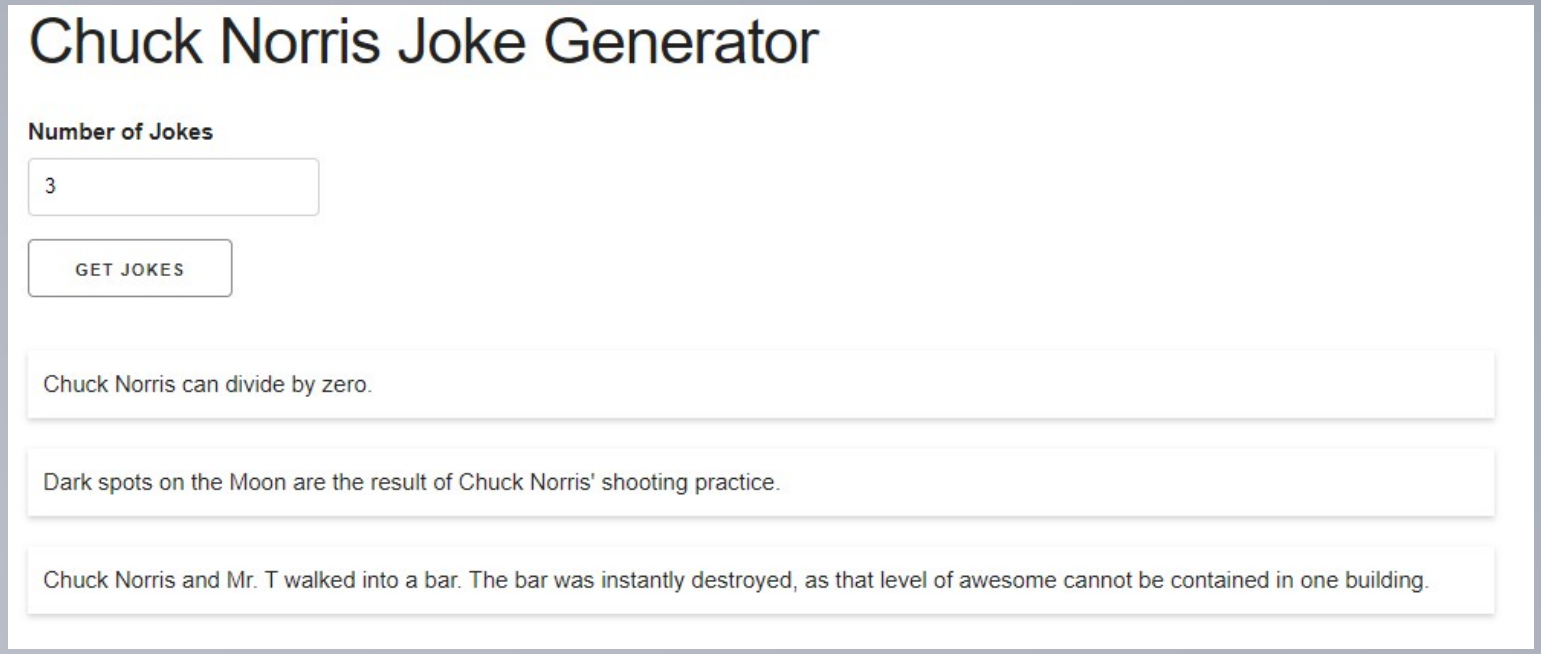
1. HTML faile sukurti tik VIENĄ tėvinį elementą, skirtą talpinti visoms nuotraukoms.
2. Kiekvieną kartą užkraunant puslapį fetch‘inti su random parašytu generatoriumi nuotraukas iš 33 skirtingų puslapio variantų;
3. Responsinti ir atvaizduoti kiekvieną nuotrauką puslapio lange vieną šalia kitos (kol telpa) kurios turėtų tokius nustatymus:
   width = "200px";
   height = "200px";
   objectFit = "cover";
   margin = "10px";



Practice Time

Jūsų tikslas yra atvaizduoti tokį kiekį juokelių, kiek vartotojas įrašys į duotą input. Atvaizduojama viskas bus po įvesto skaičiaus paspaudus mygtuką. Juokeliai bus ištraukti iš API kurio link'as yra toks: http://api.icndb.com/jokes/random/ Už pasviro brūkšnio gali eiti skaičius nuo 1 iki 619 tik šiuo atveju nėra būtina random generatoriaus pasirassyti, užtenka panaudoti skaičių kurį žmogus įrašys į input.

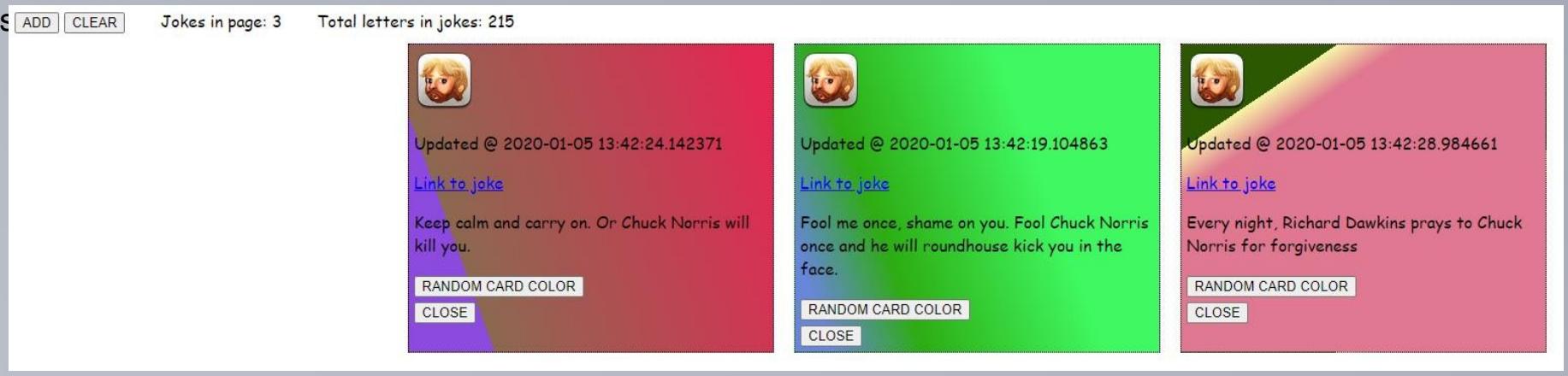
1. HTML faile šį kartą sukurti reikės daugiau, pasižiūrėkite į nuotrauką, plius papildomai tuščią div'ą juokeliams atsirasti;
2. Kiekvieną kartą įrašius skaičių į input ir pasapudus mygtuką išmes tiek juokelių koks skaičius buvo parašytas input laukelyje.

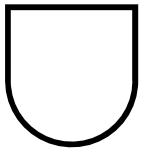


Practice Time

Jūsų tikslas yra atvaizduoti tuos pačius Chuck Norris juokelius bet per sukuriamas korteles. Atvaizduojama bus po vieną juokelį kaskart paspaudus mygtuką pridėti juokelį. Juokeliai bus ištraukti iš API kurio link‘as yra toks: https://api.chucknorris.io/jokes/random
Už pasviro brūkšnio gali eiti skaičius nuo 1 iki 619 random tvarka. Prieš tai paminėtas link‘as kiekvieną kartą jų užkrovus iš naujo, sugeneruoja random juokelį kuris sudėtas į JSON metodą.

1. HTML faile reikės sukurti du mygtukus, du span/p bei div‘ą kortelėms atsirasti;
2. Kiekvieną kartą paspaudus mygtuką – pridėti juokelį atsiras iš API ištraukti šie dalykai: kortelė su nuotrauka, kada buvo tas juokelis updated, link‘as į juokelį, pats juokelis (čia baigiasi info traukiama iš API) ir du mygtukai:
   Pirmas mygtukas sugeneruos random linear-gradient background spalvą;
   Antras mygtukas uždarys tą kortelę.
3. Taip pat puslapyje kur yra du span/p juose bus rašoma tokia informacija:
   Pirmas rodys kiek juokelių kortelių išviso yra atidarytų;
   Antras rodys kiek išviso yra raidžių juokelyje.
4. CSS dalis





Fetch API

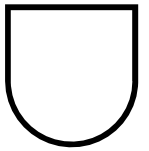
Fetch API (žiūrėkite iki 4:18)

Yra keli būdai, kaip siųsti tinklo užklausą ir gauti informaciją iš serverio. fetch() metodas yra modernus ir universalus.

Sintaksė (naudojant Promise):

```
fetch(url, options)
  .then(response => response.json())
  .then(result => /* process result */)
  .catch(error => console.log(error))
```

- *url* - prieigos URL.
- *options* - neprivalomi parametrai:
  - method - HTTP metodas (GET, POST, PUT, POST, DELETE);
  - headers - objektas su užklausos antraštėmis ;
  - body - duomenys, kuriuos reikia siųsti (užklausos tekstas) kaip eilutė, „FormData“, „BufferSource“, „Blob“ arba „UrlSearchParams“ objektas.



Fetch API

Fetch API options

Yra trys options kuriuos reikia/galima užpildyti:

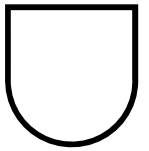
- Method – vadinamas properties nurodantis kokį HTTP metodą naudosim (privalomas). Galimi tokie metodai: GET, POST, PUT, PATCH, DELETE
- Headers – nurodo request/response HTTP headerį, kolkas visuomet naudosim: 'Content-type': 'application/json; (nereikalingas su DELETE metodu).
- Body – vadinamas body methods nurodantis kaip pasiekti response body ir ką returninti (nereikalingas su DELETE metodu);

Sintaksė: *fetch('https://*

jsonplaceholder.typicode.com/posts/1', { method:

'metodas kurį naudosime',

```
  headers:{
    'Content-type': 'application/json
  },
  body: JSON.stringify({
    key: 'value',
  })
})
```



Fetch API

Fetch API methods

Yra keturi/penki pagrindiniai (tačiau ne vieninteliai) metodai:

- GET – gauti duomenis iš API (default), pvz.: gauti Twitter user‘į pagal jo username;
- POST – įdėti naujus duomenis į API, pvz.: sukurti naują user‘į su jo vardu, username ir pašto adresu;
- PUT/PATCH – atnaujinti esamus duomenis API, pvz.: atnaujinti user‘io pašto adresą;
- DELETE – ištrinti duomenis iš API, pvz.: ištrinti user‘į iš duomenų bazės.

Sintaksė: *fetch('https://*

jsonplaceholder.typicode.com/posts/1', { method:

‘metodas kurį naudosime’,

```
  headers:{
    'Content-type': 'application/json
  },
  body: JSON.stringify({
    key: ‘value’,
  })
})
```

PUT vs POST

Practice Time

Jūsų tikslas yra sukurti Front-End dalį savo serveriui kad būtų galima rašyti naujus, trinti senus ir update'inti esamus post'us puslapyje. Jūs turite serverį, jums reikia HTML, (CSS) ir JS kuriame pagrinde dirbsite su fetch ir jo methods (properties). Jūsų puslapis turi būti susijungęs su jūsų serveriu ir viskas kas keičiama puslapyje turi atsispindėti serveryje, jokio local/session storage ar cookies čia nereikia naudoti. Jums reikės aprašyti:

1. GET, DELETE, POST, PATCH metodus;
2. Susikurti funkciją kuri renderintų kiekvieną post'ą;

Nauji dalykai kurių jums labai galimai prireiks:

- event.target.id – kai norėsite pasiekti būtent to post'o mygtuką o ne kažkurio kito;
- event.target.parentElement – kai norėsite pasiekti parent elementą target'inamo mygtuko;
- event.target.parentElement.dataset.id – kai norėsite pasiekti parent elemento dataset fetch'indami DELETE method'e.