Veebiarendus

Martti Raavel

martti.raavel@tlu.ee

Tarkvaraarendus

- Eelmise loengu meeldetuletus
- ESLint + Airbnb
    - ESLint + Airbnb

Eelmise loengu meeldetuletus

ESLint - Mis?

ESLint on populaarne avatud lähtekoodiga tööriist JavaScripti koodi staatilise koodi analüüsimiseks. Seda kasutatakse levinud kodeerimisprobleemide leidmiseks ja parandamiseks ning ühtse koodistiili rakendamiseks kogu projektis.

Airbnb stiilijuhend

Airbnb on üks populaarseim JavaScripti stiilijuhend. See on laialt levinud stiilijuhend, mis põhineb parimatel tavadel ja mille eesmärk on edendada koodi järjepidevust ja hooldatavust. Airbnb JavaScripti stiilijuhendit haldab Airbnb meeskond ning seda kasutavad paljud arendajad ja organisatsioonid üle maailma.

ESLint-i paigaldamine NodeJS projektile (VSCode) - Pistikprogramm

Kõigepealt lisa VSCode-le (kui ei ole veel lisatud) ESLint psitikprogramm. Leiad selle pistikprogrammide alt või siis siit: https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint

ESLint-i paigaldamine NodeJS projektile (VSCode) - Arendussõltuvus

Paigalda ESLint arendussõltuvusena järgmiselt:

```
npm install eslint --save-dev
```

Käivita oma projekti juurkaustast ESLint seadistamine:

```
npx eslint --init
```

Vasta küsimustele.

Progammeerimine

- Nodemon moodul
- Sünkroonne ja asünkroonne kood
  - callback
  - async/await
- Try/Catch
- Failide salvestamine ja lugemine (`fs` moodul)
- JSON.stingify()

Nodemon - Mis?

Nodemon on utiliit, mis aitab arendajatel automatiseerida Node.js rakenduste arendusprotsessi. Nodemon jälgib teie projekti failide muudatusi ja taaskäivitab automaatselt Node.js rakenduse, kui tuvastab muudatusi.

Nodemon - Paigaldamine

Nodemoni saab paigaldada globaalselt või arendussõltuvusena. Soovitan paigaldada arendussõltuvusena.

```
npm install nodemon --save-dev
```

Nodemon - Kasutamine

Käivita Node.js rakendus Nodemoniga järgmiselt:

```
nodemon app.js
```

Nodemon - Info

Paistab, et Nodemon ei taha töötada korralikult koos `prompt-sync` mooduliga.

Sünkroonne vs Asünkroonne

Mida tähendab asünkroonsus Javascriptis?

Asünkroonne kood

Asünkroonne kood on kood, mis ei blokeeri põhilõime. Asünkroonne kood võimaldab teha mitut asünkroonset toimingut korraga, ilma et peaks ootama, kuni üks toiming on lõpule viidud, enne kui teine toiming algab.

Sünkroonne kood

Sünkroonne kood on kood, mis blokeerib põhilõime. Sünkroonne kood täidetakse järjestikku, üks rida korraga. Kui üks rida on täidetud, siis järgmine rida täidetakse.

Asünkroonsus - Kuidas sellega toime tulla?

- Callback
- Async/Await

Callback - Mis?

Callback on funktsioon, mis antakse teisele funktsioonile argumendina ja see käivitatakse, kui esimene funktsioon on lõpetanud oma töö.

Callback - Näide

```
function callbackExample() {
  console.log("Callback function has been called!");
}

function greet(name, callback) {
  console.log("Hello, " + name + "!");
  callback();
}

greet("John", callbackExample);
```

Async/Await - Mis?

`async/await` on moodsa Javascripti võimalus kirjutada asünkroonset koodi viisil, mis näeb välja ja käitub nagu sünkroonne kood.

`async` ja `await` muudavad asünkroonse koodi lugemise ja kirjutamise lihtsamaks.

Aync/Await - Olekud

`async` võtmesõna kasutatakse asünkroonse funktsiooni defineerimiseks. Sellisel viisil defineeritud funktsioon tagastab alati *promise*, millel võib olla kolm olekut:

1. Ootel
2. Täidetud
3. Tagasilükatud

Async/Await - Näide

```
const url = "https://jsonplaceholder.typicode.com/posts/1"; // API URL

async function fetchData() {
  const response = await fetch(url);
  const data = await response.json();
  return data;
}
```

Async/Await - Mis juhtub, kui midagi läheb valesti?

Try/Catch - Mis?

`try...catch` lause on Javascripti konstruktsioon, mida kasutatakse vigade haldamiseks. See võimaldab meil proovida (try) täita koodi, mis võib tekitada vea, ja kui viga tekib, siis püüda (catch) see kinni ja teha midagi selle käsitlemiseks.

`try...catch` lause on kasulik, kuna see aitab vältida programmi kokkujooksmist vigade tõttu. See võimaldab käsitleda programmi täitmisel tekkivaid vigu ja jätkata programmi hoolimata olukorrast, kus tekib viga, mida NodeJS muidu ei suudaks ilma tööd lõpetamata käsitleda.

Try/Catch - Näide

```
async function main() {
  try {
    const data = await fetchData(); // Ootab fetchData funktsiooni täitmist
    console.log(data);
  } catch (error) {
    console.error(error);
  }
}
```

`fs` moodul

`fs` moodul on NodeJS-i sisseehitatud moodul, mis võimaldab lugeda ja kirjutada faile. Selle mooduli abil saad põhimõtteliselt teha kõike, mis on seotud failidega - lugeda, kirjutada, muuta, kustutada jne.

Faili lugemine

```
const fs = require("fs");

fs.readFile("file.txt", "utf8", (error, data) => {
  if (error) console.log("Tekkis viga");
  console.log(data);
});

console.log("Reading file...");
```

Faili kirjutamine

```
const fs = require("fs");

fs.writeFile("file.txt", "Hello, World!", (error) => {
  if (error) console.log("Tekkis viga");
  console.log("File has been written!");
});
```

Faili lõppu lisamine

```
const fs = require("fs");

fs.appendFile("file.txt", "Hello, World!", (error) => {
  if (error) console.log("Tekkis viga");
  console.log("Data has been appended to file!");
});
```

`fs` moodul sünkroonselt

Sünkroonne faili lugemine

```
const fs = require("fs");

const data = fs.readFileSync("file.txt", "utf8");

console.log(data);
```

Mis juhtub, kui faili ei leita?

Faili lugemisel tekkinud vea käsitlemine

```
const fs = require("fs");

try {
  const data = fs.readFileSync("file.txt", "utf8");
  console.log(data);
} catch (error) {
  console.error(error);
}
```

Sünkroonne faili kirjutamine

```
const fs = require("fs");

fs.writeFileSync("file.txt", "Hello, World!");

console.log("File has been written!");
```

Sünkroonne faili lõppu lisamine

```
const fs = require("fs");

fs.appendFileSync("file.txt", "Hello, World!");

console.log("Data has been appended to file!");
```

Objektide ja massiivide salvestamine failidesse

Kui me tahame lugeda ja kirjutada failidesse objekte ja massiive, siis tuleb kasutada `JSON.stringify()` ja `JSON.parse()` meetodeid, sest muidu loetakse ja kirjutatakse need failidesse stringidena.

JSON - Mis?

JSON on lühend *JavaScript Object Notation*ist ja see on lihtne ja loetav andmevahetusvorming, mida kasutatakse sageli andmete vahetamiseks kliendi ja serveri rakenduste vahel - näiteks veebirakenduse *Front-end*i ja API vahel.

JSON - Näide

```
{
  "name": "John",
  "age": 30,
  "isStudent": true,
  "favoriteFoods": ["pizza", "sushi", "ice cream"]
}
```

JSON - Erinevus Javascripti objektist

JSON objekt näeb väga sarnane välja Javascripti objektiga, kuid JSON-i puhul tuleb tähele panna, et JSON on andmevahetusvorming, mitte JavaScripti objekt ise. Kõige suurem erinevus on selles, et JSON-i atribuudid (võtmed) on alati ümbritsetud jutumärkidega, samas kui JavaScripti objektides võtmeid jutumärkidega tavaliselt ei kasutata.

JSON vs JavaScripti objekt - Näide

```
{
  "name": "John",
  "age": 30,
  "isStudent": true,
  "favoriteFoods": ["pizza", "sushi", "ice cream"]
}
```

```
const person = {
  name: "John",
  age: 30,
  isStudent: true,
  favoriteFoods: ["pizza", "sushi", "ice cream"],
};
```

`JSON.stringify()`

`JSON.stringify()` on JavaScripti funktsioon, mis teisendab JavaScripti objekti JSON-stringiks. See funktsioon võtab objekti sisendiks ja tagastab selle JSON-stringina.

`JSON.stringify()` - Näide

```
const person = {
  name: "John",
  age: 30,
  isStudent: true,
  favoriteFoods: ["pizza", "sushi", "ice cream"],
};

const jsonPerson = JSON.stringify(person);

console.log(jsonPerson);
```

Väljund:

```
{
  "name": "John",
  "age": 30,
  "isStudent": true,
  "favoriteFoods": ["pizza", "sushi", "ice cream"]
}
```

`JSON.parse()`

`JSON.parse()` on JavaScripti funktsioon, mis teisendab JSON-stringi JavaScripti objektiks. See funktsioon võtab JSON-stringi sisendiks ja tagastab selle JavaScripti objektina.

`JSON.parse()` - Näide

```
const jsonPerson =
  '{"name":"John","age":30,"isStudent":true,"favoriteFoods":["pizza","sushi","ice cream"]}';
const person = JSON.parse(jsonPerson);

console.log(person);
```

Väljund:

```
{
  name: 'John',
  age: 30,
  isStudent: true,
  favoriteFoods: [ 'pizza', 'sushi', 'ice cream' ]
}
```