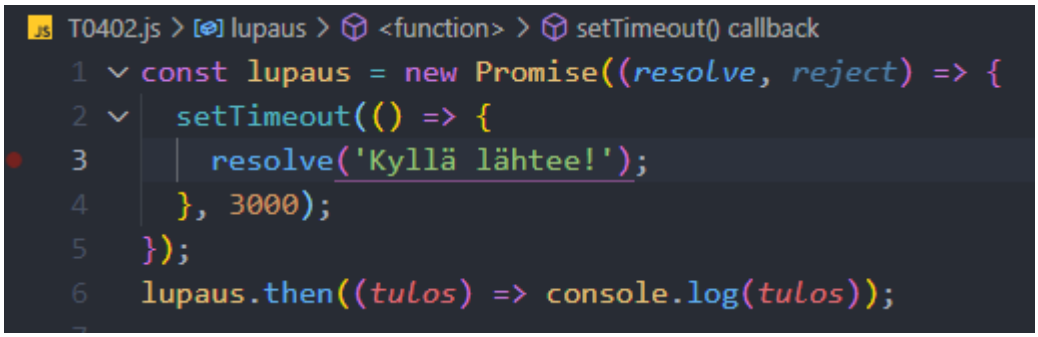
Tehtäväsetti 4.

Joonas Kangaskoski, N3303

Tehtävä 1.

- Asynkronisella ohjelmoinnilla otetaan useampi koneen ydin käyttöön ja näin ollen, toimintoja voidaan ajaa yhden aikaisesti jumittamatta tietokonetta.
- Asynkronista koodia tehdään parilla tapaa, perinteisesti callback funktioilla (kaikki callbackit ei ole asynkronisia) tai sitten modernimmin promiseilla.
- Asynkroniset operaatiot menevät jonoon ja suoritetaan sitä mukaa kun edellisestä saadaan vastaus.
- Async toiminto voidaan ketjuttaa promiseissa, näin vältytään callback helvetiltä johonka joutuu jos niiden kanssa lähtee pelailemaan.
- errorit voidaan napata näppärästi ketjun lopussa.
- promise executor toimii automaattisesti, argumentit resolve,reject
  resolve -> onnistunut työ
  reject -> error
- tuloksia voi olla vain yksi joko resolve tai reject ja ne ovat aina lopullisia
-

Tehtävä 2.



Tehtävä 3.

A.

T0403.js > ...

```
const tayttyykoLupaus = (x) =>
  new Promise((resolve, reject) =>
    x === true
      ? setTimeout(() => {
          resolve('Kyllä!');
        }, 3000)
      : reject(new Error('Ei'))
  );

tayttyykoLupaus(true).then((tulos) => console.log(tulos));
tayttyykoLupaus(false).then((error) => console.log(error.message));
```

B.

T0403.js > tayttyykoLupaus > <function>

```
const tayttyykoLupaus = (x) =>
  new Promise((resolve, reject) =>
    x === true
      ? setTimeout(() => {
          resolve('Kyllä!');
        }, 3000)
      : reject(new Error('Ei Onnistunut'))
  );

tayttyykoLupaus(true).then((tulos) => console.log(tulos));
// catchella error
tayttyykoLupaus(false).catch((error) => console.log(error.message));
```

Tehtävä 4.

T0404.js > ...

```
const ekaPromise = new Promise((resolve, reject) => {
  let ongelma;
  setTimeout(() => {
    ongelma = Math.round(Math.random());
    if (ongelma === 0) {
      resolve('dataa');
    } else {
      reject(new Error('virhe'));
    }
  }, 2000);
});
ekaPromise.then(
  (value) => console.log(value),
  (error) => console.log(error.message)
);
//.finally metodi kutsutaan kun promise on ratkennut
// joko fulfilled tai rejected
ekaPromise.finally(() => console.log('Valmista tuli!'));
```

Tehtävä 5.

T0405.js > then() callback

```
// noudetaan kaikki postit
fetch('https://jsonplaceholder.typicode.com/posts')
  // tarkistetaan että nouto on ok ja response tulee muutoin error
  .then((response) => {
    if (!response.ok) {
      throw new Error('Virhe käsiteltäessä tietoja');
    } else {
      return response.json();
    }
  })
  //tulosta on edellisen promisen tuotos.
  // tulostetaan jokaisen postauksen title eli otsikko foreachilla.
  .then((tulosta) => {
    tulosta.forEach((a) => {
      console.log(a.title);
    });
  })
  // errori nappastaan kiinni jos jossain sellainen ilmeneee
  .catch((error) => console.log(error.message));

```

Tehtävä 6.

T0406.js > ...

```
fetch('https://jsonplaceholder.typicode.com/posts/14')
  // tarkistetaan että nouto on ok ja response tulee muutoin error
  .then((response) => {
    if (!response.ok) {
      throw new Error('Virhe haettaessa posteja');
    } else {
      return response.json();
    }
  })
  // etsitään postin 14 kirjoittaja.
  .then((posti14) => {
    const kirjoittaja = posti14.userId;
    return fetch(`https://jsonplaceholder.typicode.com/users/${kirjoittaja}`);
  })
  // tarkistetaan uusi fetchi
  .then((response) => {
    if (!response.ok) {
      throw new Error('Virhe haettaessa kirjoittajaa');
    } else {
      return response.json();
    }
  })

  // tulostellaan halutut tiedot
  .then((data) => console.log(data.name, data.phone))
  .catch((error) => console.log(error.message));
```

Tehtävä 7.

T0407.js > haeMaa

```
function haeMaa(x) {
  return (
    fetch('https://restcountries.com/v2/all')
      .then((response) => {
        if (response.status === 200) {
          return response.json();
        } else {
          throw new Error('Virhe haettaessa tietoa');
        }
      })
      // haun tuloksesta näkyy että FI on alpha2codeissa,
      // joten haen function parametrilla alphoista oikeaa maata
      // selvitellään mihin indexiin tuo haku mätsää ja tulostetaan maantiedot
      .then((code) => {
        const haku = x;
        const index = code.findIndex((x) => x.alpha2Code === haku);
        if (index === -1) {
          throw new Error('virhe haettaessa maata');
        }
        return code[index];
      })
  );
}
// kokeilin myös muilla mailla
haeMaa('FI')
  .then((data) => console.log(data))
  .then((error) => console.log(error));
```

Tehtävä 8.

T0408.js > ...

```
const haeMaa = async (x) => {
  const maatResponse = await fetch('https://restcountries.com/v2/all');
  if (!maatResponse.ok) {
    throw new Error('Virhe haettaessa maita');
  }
  const code = await maatResponse.json();
  const haku = x;
  const index = code.findIndex((x) => x.alpha2Code === haku);
  if (index === -1) {
    throw new Error('virhe haettaessa maata');
  }
  return code[index];
};
haeMaa('FI')
  .then((data) => console.log(data))
  .then((error) => console.log(error));
```

Tehtävä 9.

T0409.js > haeIPmaakoodi > maakoodiResponse

```
const haeIPmaakoodi = async () => {
  const maakoodiResponse = await fetch('https://api.country.is');
  if (!maakoodiResponse.ok) {
    throw new Error('Virhe haettaessa ip:tä');
  }

  const koodi = await maakoodiResponse.json();
  console.log(koodi.ip);
};
haeIPmaakoodi();
```

Tehtävä 10.

Tehtävä 12.

1. Tämä syksy on ollut mielestäni käänteentekevä oman oppimisen taholta. Olen ymmärtänyt asiat mielestäni erittäin hyvin, varmasti senkin vuoksi, että aiemmin kasasin itselleni hirvesti taakkaa töiden, perheen ja opiskelun myötä. Nyt on ollut paremmin aikaa perehtyä itse opiskeluun ja siihen mitä lähdin koulusta hakemaan. Hallitsen opetetut asiat hyvin eikä heti meni sormi suuhun. Vaikka virheitä ja erehdyksiä tulee vieläkin osaan itse löytää ongelmat sekä lähteä ratkomaan niitä.
2. Aiemmin olin kyseisistä asioista aivan pihalla, nyt ja ymmärrän ja osaan käyttää omassa ohjelmoinnissani prototyyppejä sekä luoda asyncronista koodia. Varsinkin tehtävien viimeisissä tehtävissä tajusin, että tämähän taitaa sujua.

3. Ohjelmointi ei missään nimessä ole ”helppoa” eikä pidäkkään mielestäni olla, mutta ei se itselleni ole ylipääsemätönkään pala vaan pikemminkin nälkä kasvaa syödessä ja mitä haastavampaa tehtävää edessä sitä mukavampi sitä on lähteä pilkkomaan. Opin syksyn aikana mielestäni oikean tavan lähteä koodaamaan eli ensin pohtia päässä monelta kantilta ja sitten vasta tietokone auki.
4. Viimeisen tehtäväsetin tehtävät rupesivat olemaan jo haasteellisempia hyvällä tavalla, mutta aukesivat sitten tehtäviä tehdessä. Aina tulee haasteita vastaan, mutta en jämähdä niihin tuntikausiksi niin kuin ennen.
5. Tänä syksynä olen päässyt hyvin opiskelemaan ja saanut aikataulutettua paremmin muun elämän ohelle. Olen tehnyt paljon itsenäistä opiskelua luentojen lomassa sekä tehtäviä tehdessä. Ajallisesti kurssin alkupään tehtäviin ja luentoihin meni luonnollisesti ehkä tuon 10-12h mutta jo puolen välin ylityttyä meni reilusti enemmän.