JavaScript ohjelmointi

Virheenkäsittely

MARGIT TENNOSAARIN MATERIAALISTA
LAURA JÄRVISEN MUOKKAAMA

Miksi virheiden käsittely on tärkeää?

- Auttaa löytämään ja korjaamaan virheitä koodissa.
- Estää ohjelmaa kaatumasta yllättäen.
- Parantaa debuggausprosessia tehden koodista luotettavampaa ja helpommin ylläpidettävää.

Missä vika?

```
function getOddYears(years) {
  return yeas.filter((year) ⇒ year % 2 =/= 0);
```

Debuggaus

- Debuggausta suoritetaan silloin, kun ohjelma toimii, mutta se toimii väärin.
- Eli jos ohjelma on kaatunut, sieltä etsitään kyllä virhettä, mutta se ei *varsinaisesti* ole debuggausta.

- Konsoliin lokittaminen ja debuggerit ovat tässä apuna.

Debuggausta

```
function getOddYears(years) {
  console.log(years); // Tarkista tulostus
  const oddYears = years.filter((year) => year % 2 !== 0);
  console.log(oddYears); // Tarkista tulostus
  return oddYears;
}


function getOddYears(years) {
  debugger; // Suoritus taukoaa tässä
  return years.filter((year) => year % 2 !== 0);
}
```

Try…catch

Ajon aikaisten virheiden käsittelyyn

- **try-**lohkoon laitetaan koodi, josta halutaan siepata poikkeukset.
- **catch**-lohko ajetaan vain, jos try-lohkon sisällä **heitetään** virhe (throw).
- Eli kuin erityinen if-lause virhetilanteisiin.
- Näin koodi ei kaadu, vaan virhe voidaan kirjata tai hoitaa muuten hallitusti.

Milloin?

- Try...catch ei tarvitse käyttää koodissa joka paikassa. Säästä se niihin, joihin liittyy **riski.**
- Esimerkiksi data saattaa puuttua tai olla väärässä muodossa.

```
try {
  // Yritetään suorittaa koodilohkoa, jossa voi tapahtua virhe
  riskyOperation();
} catch (error) {
  // Virheen sattuessa tämä lohko ajetaan, ja error sisältää virheviestin
  console.error("Virhe:", error);
}
```

Esimerkki 1

```
function getOddYears(years) {
  try {
    return years.filter((year) => year % 2 !== 0);
  } catch (error) {
    console.error("Tapahtui virhe:", error);
  }
}

getOddYears("hei");
```

```
[Running] node "c:\Users\jarla\OneDrive - Business College Hels
Tapahtui virhe: TypeError: years.filter is not a function
    at getOddYears (c:\Users\jarla\OneDrive - Business College
    at Object.<anonymous> (c:\Users\jarla\OneDrive - Business C
```

Esimerkki 2

```
function divideNumbers(a, b) {
  try {
    // Try-lohko: yritetään suorittaa jako
    let result = a / b; // Jos b on määrittelemätön, tulos on NaN (Not-a-Number)
    if (isNaN(result)) {
      console.log("Virheellinen operaatio: jakaminen tuotti NaN");
    } else {
      console.log(`Tulos on: ${result}`);
    }
  } catch (error) {
    // Catch-lohko: käsittelee mahdolliset ajonaikaiset virheet (virheitä ei välttämättä synny)
    console.error(`Tapahtui virhe: ${error.message}`);
  } finally {
    // Finally-lohko: suoritetaan aina, riippumatta siitä tapahtuiko virhe vai ei
    console.log("Suoritus valmis.");
  }
}

// Esimerkkikutsut
divideNumbers(10, 2); // Kelvollinen jako
divideNumbers(10); // Jako määrittelemättömällä arvolla (tulos on NaN)
```

Esimerkki 2

```
[Running] node "c:\Users\jarla\OneDrive - Business College
Tulos on: 5
Suoritus valmis.
Virheellinen operaatio: jakaminen tuotti NaN
Suoritus valmis.
```

Virhetyypit JavaScriptissä

- **SyntaxError:** Virheellinen syntaksi.
- **ReferenceError:** Viittaus määrittelemättömään muuttujaan.
- **TypeError:** Väärän tietotyypin käyttäminen.
- **RangeError:** Arvo on sallitun rajan ulkopuolella.
- **URIError:** Virheellinen käyttö URI-funktioissa.

Omat custom-virheilmoitukset

- Luo mukautettuja virheitä tarvittaessa paremman virheenkorjauksen tueksi.

```
function checkNumber(num) {
  if (isNaN(num)) {
    throw new Error("Syötteen tulee olla numero");
  }
  console.log("Numero on", num);
}
try {
  checkNumber("test"); // Heittää virheen kun 'test' ei ole numero
} catch (error) {
  console.error(error.message); // Lokittaa "Syötteen tulee olla numero"
}
```

Oman virheilmoituksen tekeminen

1. **Heitä:** throw new Error("Selkeä viesti");
2. **Tartu:** try { ... } catch (err) { console.error(err.name, err.message); }

- Kirjoita kohdassa, jossa haluat virheilmoituksen syntyvän, throw new Error("Tässä oma viesti");
- Jos haluat tyypittää virheet tai erotella eri virhesyyt, tee uusi luokka, joka perii Error-luokan.
  - Aseta konstruktorissa this.name = "OmaVirhe"; ja kutsu super(message).
  - Näin voit catch-lohkossa tarkistaa if (err instanceof OmaVirhe).
- Kun virhe on heitetty, try...catch-rakenteessa voit lukea error.name ja error.message ja reagoida niihin, esim. näyttää käyttäjälle selkeä ilmoitus tai kirjata lokiin.

Harjoitellaan

Löydät tehtäviä virheiden käsittelyyn harjoittelemiseen viikon 9 kansiosta

https://github.com/bc-web-ohjelmistokehitys/WP25K-JS

Huom! Opettele itse - älä pyydä tekoälyä ratkaisemaan tehtävää puolestasi.