



Tehtävä 1



Samuli Rukkila



3

Skills completed

109

Badges

Joined Oct 27, 2017

430 total points

4 day streak



Last coded 32 minutes ago

Completed Skills 3

Learn HTML		>
Learn CSS		>
Introduction To JavaScript	<div></div>	<div>✓</div> >

ity Catalog

Upgrade to Pro



Introduction to JavaScript

Learn JavaScript fundamentals and build interactive websites and pages.

COMPLETED

Asynkronisia pyyntöjä on 3 erilaista

1. CALLBACK

Callback-metodi on funktio mikä suoritetaan VASTA kun toinen metodi on suoritettu loppuun. Esim:

```
function teeTehtava(aihe, callback) {  
  alert(`Aloitetaan kotitehtävä ${aihe}.`);  
  callback();  
}  
function kerroLopetuksesta(){  
  alert('Kotitehtävä tehty.');}  
teeTehtava('matikka', kerroLopetuksesta);
```

Metodi "kerroLopetuksesta" ei siis suoriudu ennen kuin teeTehtava-metodi on suoritettu. Tämä tyyli on hyvä tapa lyhyemmissä asynkronisissa-pyyntöissä, mutta kun pyyntöjä tulee paljon tämä tyyli tekee koodista hyvin vaikealukuista ja monimutkaista. Tästä syystä keksittiin uudempi asynkroninen pyyntö joka on..

2. PROMISE/THEN

Promise edustaa lopullista suoriutumista (tai epäsuoriutumista) asynkronisessa toiminnassa. Metodiin annetaan kaksi parametria "resolve" ja "reject". Metodin suoritettua "resolve"-metodin jatkotoimenpiteet ajetaan. Virheen sattuessa siirrytään "reject"-metodiin. Promise-metodissa ei tiedetä koska promise luodaan. Promisea voidaan jatkaa ".then"-avainsanalla, joka ajetaan vasta, kun promise on valmis. Promise on suosittu tapa siinä vaiheessa kun callback-metodeja tulee paljon.

3. ASYNC/AWAIT

ES6:ssa saimme uudenlaisen, helpomman tavan tehdä asynkronisia pyyntöjä. Async-avainsana voidaan laittaa esim. metodin eteen (async function x() {}). Metodin edessä async tarkoittaa, että metodi palauttaa aina lupauksen (promise). Async voidaan yksinkertaisesti siis luoda esim:

```
async function f() {  
  return 1;  
}  
f().then(alert); // 1
```

Jos Async funktio ei palauta promisea - se luodaan semmoiseksi; tästä siis sanonta "se palauttaa aina lupauksen".

Asyncia voidaan käyttää myös await-avainsanan kanssa.

```
let value = await promise;
```

Await laittaa Javascriptin odottamaan, että promise on valmis, ennenkuin se alustaa muuttujaan kyseisen arvon.

Await-avainsanaa ei tosin voida käyttää ilman Async-sanaa. Jos virhe sattuu async-metodissa, se voidaan napata

try..catch -metodeilla. Async-metodin ulkopuolella voimme myös käyttää .catch-funktiota napataksemme virheitä.

Async on nykyään yksi suosituimmista ja helpoimmista tavoista luoda asynkronisia pyyntöjä.

Ainoa huono puoli

Async-metodeissa on ollut sen vaikea tapa käsitellä virheitä.