



This Moodle environment is for archive use only. All the active courses at the beginning of the year 2020 are held in the [Learn](#) and [Open Learn](#) environment.

Oppimistehtävä 5

Arrayt ja array funktiot

Tee seuraavat tehtävät. Pakkaa tiedostot yhteen .zip -pakettiin ja palauta Moodlen kautta. Kirjoita jokaisen tehtävän ratkaisu niille tarkotettuihin tiedostoihin (ot5a.js, ot5b.js, ot5c.js ja ot5d.js) kommentteilla merkatuihin alueihin. Älä muuta muita koodeja!

Tämän tehtäväsarjan tehtävät koskevat javascript.info:n kappaleita 5.4 ja 5.5. Käytä näitä kappaleita materiaalina tehtävien tekemiseen. Voit myös käyttää allaolevaa cheatsheettia tärkeimpien pointtien kertaamiseen.

Mikäli haluat syventyä vielä lisää JavaScript-ohjelmointiin, voit tutustua tehtäväsarjan muihin koodeihin, kuten ot5.html:ään, js/app.js:ään ja ohjelmassa käytettyyn jQuery ja Bootstrap -kirjastoihin ja tutkia miten tämä koko tehtäväsarja on koodattu.

Cheatsheet

Arrayt

Arrayt ovat JavaScript objekteja, joihin voidaan tallentaa useita elementtejä, eli alkioita ja muodostaa näin joukkoja. Array eroaa tavallisista objekteista mm. siten, että arrayn sisältämät objektit eivät tarvitse avainta itselleen, toisin kuin objektit. Arrayn luominen tapahtuu seuraavilla syntakteilla:

```
let array = [];  
let array = new Array();
```

Molemmat lauseet luovat tyhjän arrayn, joka ei sisällä yhtäkään objektia.

Voit myös alustaa arrayn antamalla sille sisältöä:

```
let nopanLuvut = [ 1, 2, 3, 4, 5, 6 ];
```

Vaikka arrayn sisältämät objektit eivät tarvitsekaan avainta, niillä on kuitenkin indeksinumero, joka kertoo kuinka mones objekti on kyseessä. Indeksointi alkaa nollasta, joten ylläolevan esimerkin indeksissä 0 on numero 1. Vastaavasti, seuraavan esimerkin indeksissä 2 on merkkijono 'Kolme':

```
let nopanLuvut = [ 'Yksi', 'Kaksi', 'Kolme', 'Neljä', 'Viisi', 'Kuusi' ];
nopanLuvut[2] === 'Kolme' //Tämä lause on true
```

Numeroiden kanssa pelatessa laskennan aloittaminen nollasta voi tuntua hankalalta, mutta siihen on monia hyviä syitä, ja arraylle on paljon muita käyttötarkoituksia, missä ei käytetä numerosarjoja.

Arrayn sisältöä voidaan käydä läpi erilaisilla loopeilla. Seuraavassa esimerkissä etsitään arraysta suurin numero:

```
let array = [ 1, 5, 7, 3, 4, -5 ];
let suurin = array[0] //Alustetaan suurimman luvun arvo arrayn ensimmäisellä alkiolla
array.forEach(alkio => {
  if(suurin < alkio)
    suurin = alkio;
});
console.log(suurin); //Tulostaa numeron 7
```

Edelläoleva esimerkki siis käy koko arrayn läpi niin, että yksi kerrallaan jokainen arrayn alkio asetetaan parametriksi forEach-funktion sisältämään funktioon. Jos alkio on suurempi kuin forEach-loopin edellä alustettu muuttuja "suurin", niin suurin-muuttujan arvoksi asetetaan alkion arvo, jolloin arrayn läpikäynnin jälkeen suurin-muuttujan arvoksi pitäisi jäädä taulun sisältämä suurin lukuarvo. forEach on JavaScriptin Array-tyypin metodi, eli se on array-funktio. Samaan lopputulokseen päästäisiin myös seuraavilla loopeilla:

```
for (let alkio of array) {
  if(suurin < alkio)
    suurin = alkio;
}
```

TAI

```
for (let i = 0; i < array.length; i++) {
  let alkio = array[i];
  if(suurin < alkio)
    suurin = alkio;
}
```

Nämä molemmat tekevät saman asian, kuin ensimmäisenä esitetty `forEach`-funktio, mutta `forEach` ollee näistä elegantein tapa käydä arrayn alkioita läpi, ja sitä tulisi suosia. Erityisesti viimeisen esimerkin käyttäminen ei ole modernissa JavaScript -kehityksessä yleensä perusteltua. Huomaa, että jos sinun tarvitsee käyttää alkion indeksia `forEach`-funktiossa, saat sen käyttöösi antamalla toisen parametrin funktioparametrille ja sulkemalla parametrit sulkeisiin:

```
let arr = [ 'Eka', 'Toka', 'Kolmas' ];
arr.forEach( (alkio, indeksi) => {
  console.log(alkio + indeksi); //Tulostaa Eka0, Toka1 ja Kolmas2
});
```

Array funktiot

JavaScripti tarjoaa arrayille kattavan määrän erilaisia metodeja, jotka helpottavat arrayiden käsittelyssä. Tässä on esimerkkejä metodeista, joita tarvitset tehtävissä, mutta metodeja on huomattavasti enemmän joihin kannattaa tutustua jatkaessasi JavaScript -ohjelmoinnin opettelua.

forEach

Kuten edellä on esitelty, `forEach`-funktioita voi käyttää arrayn läpikäyntiin:

```
[ 1, 2, 3, 4 ].forEach(num => console.log(num));
```

find

Find-metodilla voidaan etsiä joukosta ensimmäinen alkio, joka toteuttaa find-metodille annetun ehtolauseen:

```
let arr = [
  { arvo: 1, teksti: 'Yksi' },
  { arvo: 2, teksti: 'Kaksi' },
  { arvo: 3, teksti: 'Kolme' },
];
const objekti = arr.find(alkio => alkio.arvo === 2);
console.log(objekti.teksti); //Tulostaa tekstin "Kaksi"
```

push

Push-metodilla voit lisätä arrayhin uuden arvon. Metodi lisää arvon arrayn perään uudelle indeksille:

```
let array = [ 1, 2, 3 ];
array.push(4); //Array on tämän jälkeen [ 1, 2, 3, 4]
```

Huomaa, että push-metodia ei aina ole suositeltavaa käyttää. Jos esimerkiksi taulu on annettu parametrina funktiolle, niin sen manipulointi funktiossa muuttaa arrayta myös siellä mistä funktiota on kutsuttu. Funktionaalisen ohjelmoinnin periaatteiden mukaan on

suositeltavampaa luoda uusi array, johon on lisätty haluttu alkio, esim. concat-funktiolla, joka yhdistää kaksi arrayta uudeksi arrayksi:

```
let array = [ 1, 2, 3 ]
let largerArray = array.concat( [4] ); //Nyt largerArray sisältää [ 1, 2, 3, 4 ], mutta
array säilyy koskemattomana
```

filter

Filter-metodi toimii samankaltaisesti kuin find-metodi, mutta sen sijaan että metodi palauttaisi ensimmäisen alkion joka sopii ehtolauseeseen, filter-funktio palauttaa uudessa taulussa kaikki alkiot jotka sopivat ehtolauseeseen. Jos yksikään alkio ei sovi ehtolauseeseen, palauttaa filter-funktio tyhjän taulun:

```
let kortit = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 ];
let kuvakortit = kortit.filter(kortti => kortti > 10); //kuvakortit -taulun sisältö on [
11, 12, 13, 14 ]
```

map

Map-metodi on myös hyvin oleellinen metodi arrayden käsittelyssä. Sillä voidaan tehdä taulusta uusi taulu, missä alkioista on muodostettu jotakin uusia olioita. Map-metodille annetaan parametrikseen funktio, joka ottaa vastaan alkion, ja funktio palauttaa jonkin uuden arvon, mikä asetetaan uudessa taulussa vastaavan indeksin arvoksi:

```
let asteet = [ 45, 90, 180, 270 ];
let vastakkaisetAsteet = asteet.map(aste => {
  if(aste >= 180)
    return aste - 180;
  else
    return aste + 180;
});
```

Ylläolevassa esimerkissä vastakkaisetAsteet-array sisältää [225, 270, 0, 90]. Jokaisesta asteluvusta siis muodostetaan sen vastasuuntaan osoittava asteluku, ja näistä vastakkaisista asteluvuista muodostetaan uusi array.

reduce

Reduce-metodi käy läpi arrayn kaikki alkiot siten, että niistä muodostetaan uusi arvo, joka annetaan seuraavan alkion mukana parametrina seuraavalle kierrokselle. Funktion kierrokselta toiselle kuljetettava arvo siis kumuloituu jokaisen kierroksen aikana. Ensimmäisenä parametrina reduce-metodille annetaan funktio, joka saa parametrikseen kumuloituvan arvon, sekä seuraavan alkion. Reduce-funktiolle voi myös antaa toisena parametrina lähtöarvon, jota aletaan kumuloimaan. Muussa tapauksessa lähtöarvo on arrayn ensimmäinen alkio. Tässä on esimerkki taulun arvojen summan laskennasta:

```
let array = [ 2, 5, 8, 11, 14 ];
const summa = array.reduce( (kum, alkio) => { kum + alkio }, 0 );
```

Ylläoleva esimerkki laskee taulun alkioden summan arvoksi 40, ja suorittaa laskennan seuraavasti:

$$0 + 2 = 2,$$

$$2 + 5 = 7,$$

$$7 + 8 = 15,$$

$$15 + 11 = 26,$$

$$26 + 14 = 40$$

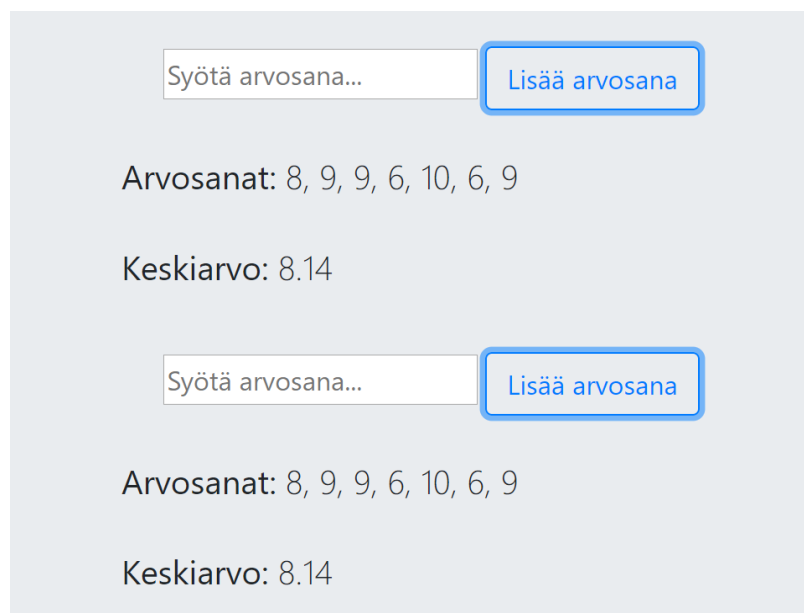
Tehtävät

a)

Tee ohjelma, joka laskee sille annettujen numeroiden keskiarvon.

Toteuta keskiarvon laskenta skriptin ot5a.js:ssä olevassa laskeKeskiarvo-funktiossa. Funktio saa parametrikseen taulun, joka sisältää käyttäjän syöttämiä numeroita. Funktion tulee palauttaa taulun lukujen keskiarvo.

Vinkki: Käytä reduce -funktiota summan laskentaan ja length getteriä taulun alkioden määrän selvittämiseen



Syötä arvosana... Lisää arvosana

Arvosanat: 8, 9, 9, 6, 10, 6, 9

Keskiarvo: 8.14

Syötä arvosana... Lisää arvosana

Arvosanat: 8, 9, 9, 6, 10, 6, 9

Keskiarvo: 8.14

b)

Jalostetaan tehtävän 2b noppaohjelmaa käyttämään taulukoita ja eri kielivaihtoehtoja. Tee ohjelma, joka tulostaa noppien silmäluvut erikseen, jos ne ovat eri lukuja, tai jos ne ovat parit, niin tulosta mikä pari on kyseessä.

Tässä tehtävässä sinulle on annettu silmälukujen ja parien tekstit tauluina, jotka sisältävät sekä suomen-, että englanninkielisen tekstin.

Toteuta nopanHeitto -funktio siten, että se palauttaa esim. 'Silmäluvut ykkönen ja vitonen' parametrina saatujen silmälukujen mukaan, mikäli parametrina saatu **kieli** on 'suomi'. Mikäli kieli on 'englanti', palauttaa funktio esim. 'Numbers one and five'.

Mikäli numerot ovat pareja, tulostaa ohjelma suomeksi esim. 'Ykköspari' ja englanniksi 'Pair of ones'.

Vinkki: Käytä find-metodia oikeiden tekstien hakemiseen silmäluvun perusteella



c)

Tässä tehtävässä toteutat oman alkeellisen Twitter-sovelluksesi, missä voit lisätä tweettejä ja hakea niitä käyttäjän ja otsikon perusteella.

Toteuta Twitter-luokan lisääTweetti -metodi ja etsiTweetti -metodi.

lisääTweetti -metodi lisää uuden tweetti-objektin tweetit-tauluun samassa muodossa kun ensimmäinen esimerkkinä oleva tweetti on käyttäen parametrina saatuja arvoja.

etsiTweetti-metodi filtteröi tweetit-aulusta ne alkiot, joiden otsikko tai käyttäjä sisältävät parametrina annetun hakuehto-merkkijonon.

Vinkki: Käytä push-metodia tweettien lisäämiseen ja filter-metodia tweettien filteröimiseen.

Käyttäjänimi

Juhani

Otsikko

Random ränttäys

Teksti:

Kyllä minä niin mieleni pahoitin kun...

Lisää tweetti

Etsi tweetteja

Hakuehto

Kirjoita tähän...

Esimerkkiotsikko

By: Keijo Käyttäjä

Tweetin sisältö

Koulutehtävät

By: Teemu

Koulutehtävät ovat kuin konvehtirasia. Jokainen on uusi ja ihmeellinen yllätys. Ja joskus vastaan tulee ananasliköörikonvehti.

d)

Tässä tehtävässä tehdään ohjelma, joka tulostaa käyttäjän lisäämät nimet. Käyttäjälle voi myös valita näytetäänkö käyttäjistä etunimi ja/tai sukunimi tai ei kumpaakaan.

Toteuta haeNimet -funktio, joka palauttaa **nimilista**-taulusta saadut nimet parametrina saatujen näkyvyysvalintojen mukaan. Jos naytaEtunimi on true, palautetaan etunimi. Jos naytaSukunimi on true, palautetaan sukunimi. Jos molemmat on true, palautetaan koko nimi muodossa "Elli Esimerkki". Mikäli kumpikin on false, palautetaan tyhjä merkkijono: "".

Funktion siis tulee palauttaa uusi taulu, joka muodostuu käyttäjän valinnan mukaisista merkkijonoista.

Vinkki: Käytä palutettavan taulun muodostamiseen map-metodia.

Etunimi Krista

Sukunimi Kaunainen

- ☐ Näytä etunimi
☒ Näytä sukunimi

Lisää henkilö

Nimet: Esimerkki, Salminen, Palmu, Erottaja,
Kaunainen

Palautuksen tila

| | |
|------------------------|-------------------------|
| Suorituskerran numero | Tämä on suorituskerta 1 |
| Palautuksen tila | Ei suorituskertoja |
| Arvioinnin tila | Ei arvioitu |
| Viimeksi muokattu | - |
| Palautuksen lisätiedot | ► Kommentit (0) |

Lisää palautus

Muokkaa palautustasi

NAVIGOINTI



Työpöytä

Sivuston etusivu

Omat opintojaksoni

Opintojaksokategoriat

Moodle-alustan tilaus (Opettajille)

ASETUKSET



Opintojakson ylläpito



Oppaat- ja ohjeet - Kysy eTuutorilta - Moodle-tuki