Oppimistehtävä 3

Objektit (oliot) ja funktiot

Tee seuraavat tehtävät. Pakkaa tiedostot yhteen .zip –pakettiin ja palauta Moodlen kautta. Kirjoita jokaisen tehtävän ratkaisu niille tarkotettuihin tiedostoihin (ot3a.js, ot3b.js, ot3c.js, ot3d.js ja ot3e.js). Älä muuta muita koodeja!

Tämän tehtäväsarjan tehtävät koskevat javascript.info:n kappaleita 2.14 - 2.16 ja 4.1. Käytä näitä kappaleita materiaalina tehtävien tekemiseen. Voit myös käyttää allaolevaa cheatsheettia tärkeimpien pointtien kertaamiseen. Lisäksi voit tutustua aiheeseen esimerkkien avulla tutustumalla oppimistehtävän yhteydessä olevaan ot3_demot-tiedostoon, minkä sisällä olevien html-tiedostojen lähdekoodista löydät aiheeseen liittyviä skriptejä.

Mikäli haluat syventyä vielä lisää JavaScript-ohjelmointiin, voit tutustua tehtäväsarjan muihin koodeihin, kuten ot3.html:ään, js/app.js:ään ja ohjelmassa käytettyyn jQuery ja Bootstrap -kirjastoihin ja tutkia miten tämä koko tehtäväsarja on koodattu.

Cheatsheet

Objektit

Objektit (suomenkielessä käytetty myös oliot, mistä tulee myös termi olio-ohjelmointi), ovat JavaScriptin perus elemenettejä. Objekteilla on kenttiä, tai properteja, joihin voi tallentaa mielivaltaista dataa, kuten numeroita, stringejä, funktioita tai vaikka muita objekteja. Objekteilla voidaan siis muodostaa ohjelmistorakenteita, jotka tekevät koodista helpommin ymmärrettävää ja selkeämpää. Objektin luominen tapahtuu seuraavalla syntaksilla:

```
let olio1 = new Object();
```

tai

```
let olio2 = {};
```

Jälkimmäinen syntaksi mahdollistaa sen, että objektille annetaan heti luontivaiheessa joitakin kenttiä:

```
let henkilo = {
  nimi: 'Pirjo',
  ika: 42
};
```

Objektin kenttiä voidaan muuttaa ja asettaa myös jälkikäteen:

```
henkilo.nimi = 'Esko';
```

tai

```
henkilo['nimi'] = 'Esko';
```

Jälkimmäinen on harvemmin käytetty tapa, ja sitä tulisi välttää, sillä syntaksi sekoittuu taulu-syntaksin kanssa, joista kerrotaan lisää myöhemmillä oppitunneilla. Joissain tilanteissa jälkimmäistä syntaksia on kuitenkin pakko käyttää, esim. jos objektin kentän nimi päätellään ohjelmallisesti yhdistelemällä stringejä tms.

Tässä esimerkki hieman hierarkisemmasta objektista:

```
let maapallo = {
  nimi: 'Maapallo'
  nimiEng: 'Earth'
  populaatio: 7700000000,
  pintaAla: 510072000,
  ilmasto: {
    keskilampotila: 14.6,
    elinkelpoisuus: 'Hyvä'
  },
  johtavaElainlaji: {
    nimi: 'Ihminen',
    ravinto: {
      paaRavinto: 'hiilihydraatit',
      toinenRavinto: 'rasvat',
      kolmasRavinto: 'proteiinit'
    }
  }
};
```

Yllä oleva esimerkki on naiivi demonstraatio siitä, mitä objekteilla voidaan tehdä. Oleellista tässä vaiheessa on ymmärtää, että JavaScript- ja muu ohjelmointi ei ole pelkkien numeroiden ja stringien parissa nikkarointia, vaan objektit (kuten funktiotkin) mahdollistavat **abstraktioiden** muodostamisen, jolloin jonkin objektin yksinkertaisella käsittelyllä voi saada aikaan merkittäviä tuloksia, kuten hienoja animaatioita, fysiikkalaskentaa, tietokoneiden etähalliintaa, suurien konesalien kontrollointia jne. Näihin monimutkaisempiin objekteihin ja funktioihin pääset tutustumaan myöhemmin, kun jatkat ohjelmoinnin opiskelua, ja alat tutustumaan erilaisiin libraryihin (kirjastoihin) ja frameworkeihin.

Funktiot

Myös funktiot ovat JavaScriptin peruselementtejä. Funktiot sisältävät jotakin koodia, jota voi suorittaa **kutsumalla** funktiota. Oltkin jo tähän mennessä nähnyt kurssin tehtävissä useita funktioita, joiden sisälle olet kirjoittanut koodiasi. Näitä funktioita on kutsuttu tehtävää

hallinnoivasta koodista, jolloin funktion koodi on suoritettu. Funktion luomiseen voidaan käyttää muutamaa erilaista syntaksia:

```
const funk1 = (param1, param2) => {
  return param1 + param2;
};
```

tai

```
const funk2 = function(param1, param2) {
  return param1 + param2;
};
```

tai

```
function funk3(param1, param2) {
  return param1 + param2
}
```

Kaksi ensimmäistä esimerkkiä tekevät käytännössä täysin saman asian, eli luo funktion, joka asetetaan muuttujan arvoksi. Tämä onkin tärkeä konsepti JavaScriptissa! Funktio voi siis olla muuttujan arvo, ja sitä voidaan käsitellä samalla tavoin kuin muitakin muuttujia. JavaScriptissa funktiot ovat siis ns. "First-class citizen", sillä niitä käsitellään yhtävertaisesti muiden muuttujien kanssa.

Viimeinen esimerkki eroaa kahdesta edellisestä siten, että funktiota ei ole asetettu muuttujaan, vaikkakin funktion voi edelleen antaa arvoksi jollekin muuttujalle. Se tärkeä ero viimeisellä esimerkillä kuitenkin on, että tällä tavoin alustettu muuttuja on jo olemassa heti skriptin tai koodiblokin alussa, kun taas muuttujiin asetetut funktiot ovat olemassa vasta muuttujan alustamisen jälkeen. Kutsuminen tapahtuu seuraavasti:

```
funk1(1, 2);
```

```
funk2(1, 2);
```

```
funk3(1, 2);
```

Jos kuitenkin funktioita funk1 tai funk2 kutsuttaisiin ennen funktion luomista, aiheuttaisi se virheen ohjelmassa.

Funktion määritelmään (signature) kuuluu **parametrien** määritteleminen. Funktiolle voidaan siis syöttää mielivaltaisia muuttujia parametreina, jolloin funktiolla on käytettävissä nämä muuttujat. Parametrit määritellään funkion luomisvaiheessa sulkeiden sisään. Esimerkeissä param1 ja param2 olivat funktioiden parametreja. Näin ollen esimerkkien funkitoita kutsuttaessa, funktiolle syötetään myös kaksi parametria, esim. funk1(1, 2), missä parametrit ovat 1 ja 2.

Lopuksi, funktioilla on vielä return -lause, jolla funktio voi palauttaa jonkin arvon. Esimerkin funktioissa return -lause palauttaa parametrien summan. Jos return -lausetta ei kirjoiteta lainkaan, suoritetaan se automaattisesti funktion lopuksi, ja funktio palauttaa undefined -arvon. Näin funktioiden palauttaman arvon voi asettaa jonkin muuttujan arvoksi, esim:

```
const tulos = funk1(2,3);
console.log(tulos) //Tämä tulostaa "5"
```

On hyvä huomata, että funktion sisällä luotuja muuttujia ei voi käyttää funktion ulkopuolella, esim:

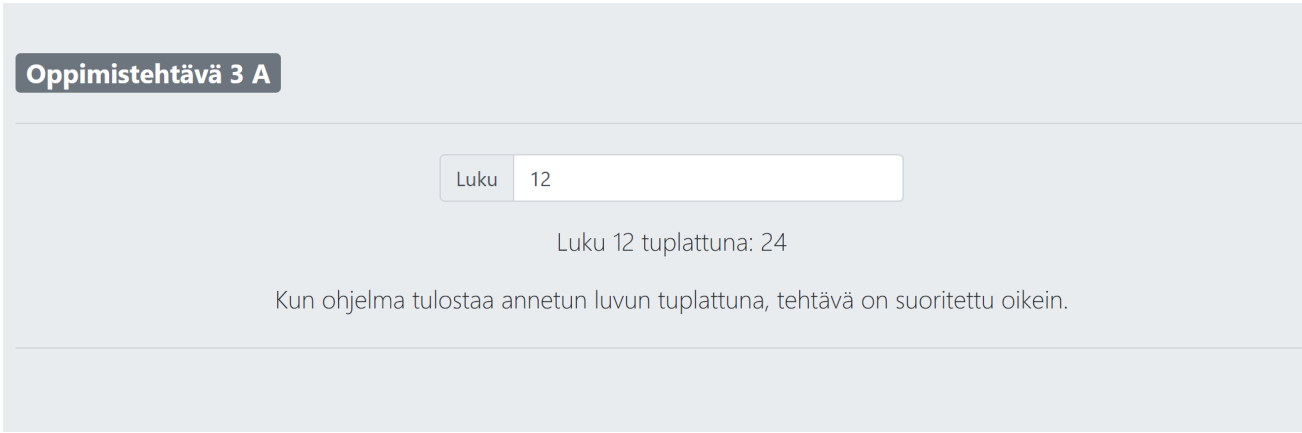
```
const tulostaNimi = (nimi) => {
  let prefix = 'Tulostetaan: ';
  console.log(prefix + nimi);
};
let nimi = 'Timppa';
tulostaNimi(nimi);
console.log(prefix + nimi); //Tämä aiheuttaa virheen, koska prefix-muuttuja on alustettu
vain funktion sisällä!
```

Tehtävät

Tehtäviä tehdessäsi pidä huoli, että funktioiden nimet ovat tismalleen tehtävänannon mukaisia! Muuten funktioita kutsuva tehtäväkoodi ei toimi oikein.

a)

Tee funktio nimeltä ***tuplaa()***, joka saa kokonaisluvun ja palauttaa sen tuplattuna (luku kerrotuna kahdella).

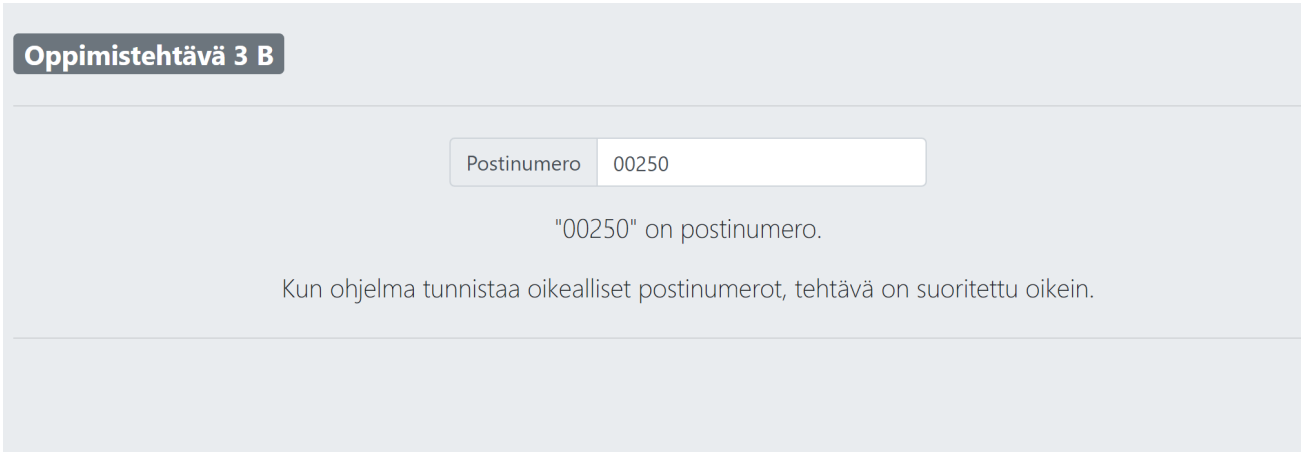


b)

Tee funktio tarkastaPostinumero(), joka saa parametrina postinumeron ja tarkastaa postinumeron oikeellisuuden. Funktio palauttaa arvon true, jos syöttötiedossa on viisi numeroa. Muuten palautetaan false.

Merkkijonon pituuden laskentaan voit käyttää JS:n sisäänrakennettua ominaisuutta *length* (esim. 'tietskari'.length).

Merkkijonon numeerisuuden voi selvittää esim. isNaN() -funktiolla (is not a number?), esim. isNaN('Aasi') === true ja isNaN('241') === false. Voit yhdistää tähän negaatio-operaattorin selvittääksesi onko merkkijono numeerinen.



c)

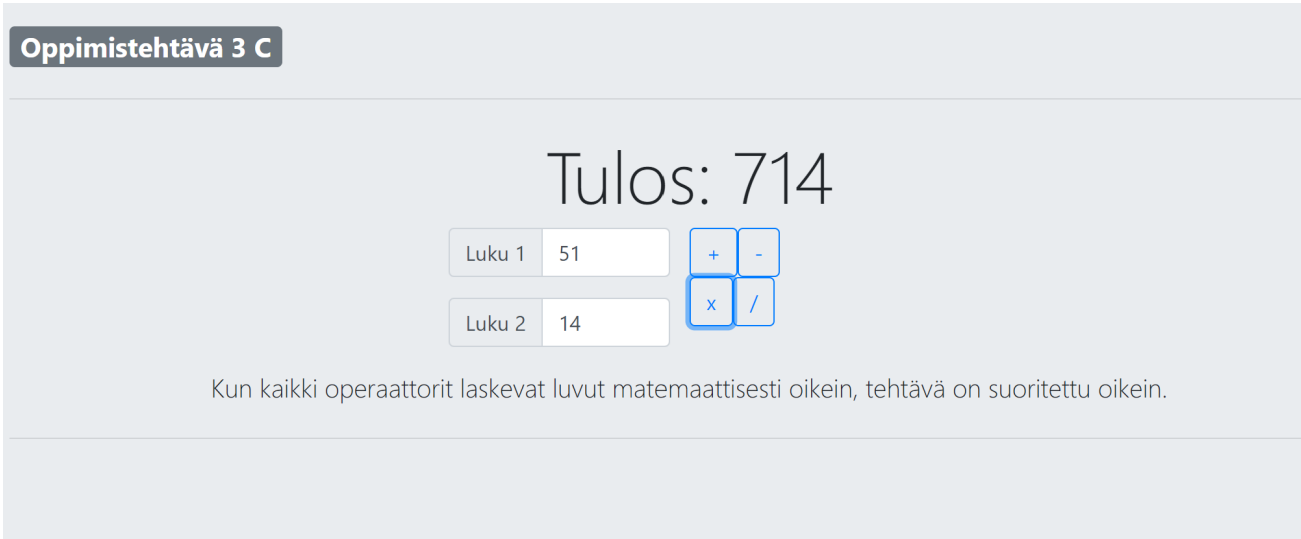
Luo ensin seuraavat funktiot:

- **laskeYhteen(luku1, luku2)**
- **vahenna(luku1, luku2)**
- **kerro(luku1, luku2)**
- **jaa(luku1, luku2)**

Nämä funktiot summaavat, vähentävät, kertovat ja jakavat parametreina annetut luvut toisillaan. Voit katsoa mallia ot1.js -tiedostosta ensimmäisestä tehtäväsarjasta.

Luo sitten funktio: **valitseLasku(merkki)**, joka **palauttaa jonkin edellä luomista funktioistasi.** Merkki voi olla '+', '-', '*' tai '/', ja riippuen siitä minkä merkin funktio saa parametrikseen, palautetaan jokin edellämainituista laskufunktioista.

Viimeiseksi, luo funktio: **suoritaLaskutoimitus(luku1, luku2, merkki)**, joka hakee ensin oikean laskutoimituksen valitseLasku-funktiolta, ja laskee haetulla funktiolla oikean tuloksen ja palauttaa sen.

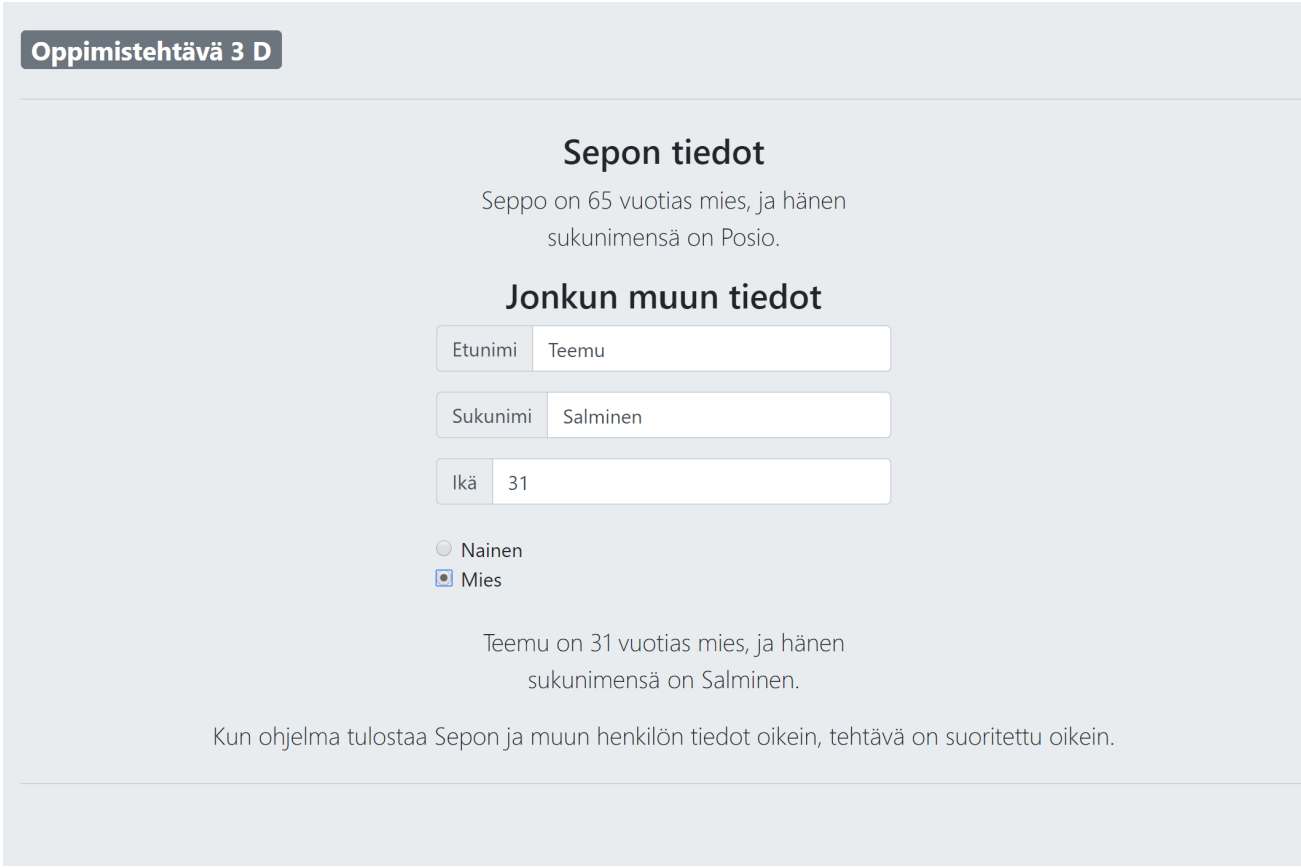


d)

Luo objekti nimeltä "seppo", jolle annat kentät etunimi, sukunimi, ika ja sukupuoli. Sepon sukunimi on Posio, ikä on 65 ja sukupuoli on mies.

Tee sitten funktio **teeIhminen(etunimi, sukunimi, ika, sukupuoli)**, joka palauttaa objektin, jolla on vastaavat kentät kun Sepolla, mutta ne on asetettu parametrina saatujen arvojen mukaan.

Lopuksi luo funktio **muodostaInfoTeksti(ihminen)**, joka ottaa edellä mainittua muotoa olevan objektin parametrinaan ja tulostaa ihmisen tiedot (Sepon tapauksessa) näin: "Seppo on 65 vuotias mies, ja hänen sukunimensä on Posio."



e)

Kiinteistönvälittäjä tarvitsee ohjelman, jolla voi tallentaa ja hakea myytävien asuntojen tietoja.

Asunnon ominaisuuksia ovat *kohdenumero, osoite, hinta ja pinta-ala*. Asunnon olennaisena tietona on myös neliöhinta, joka voidaan laskea hinnasta ja pinta-alasta.

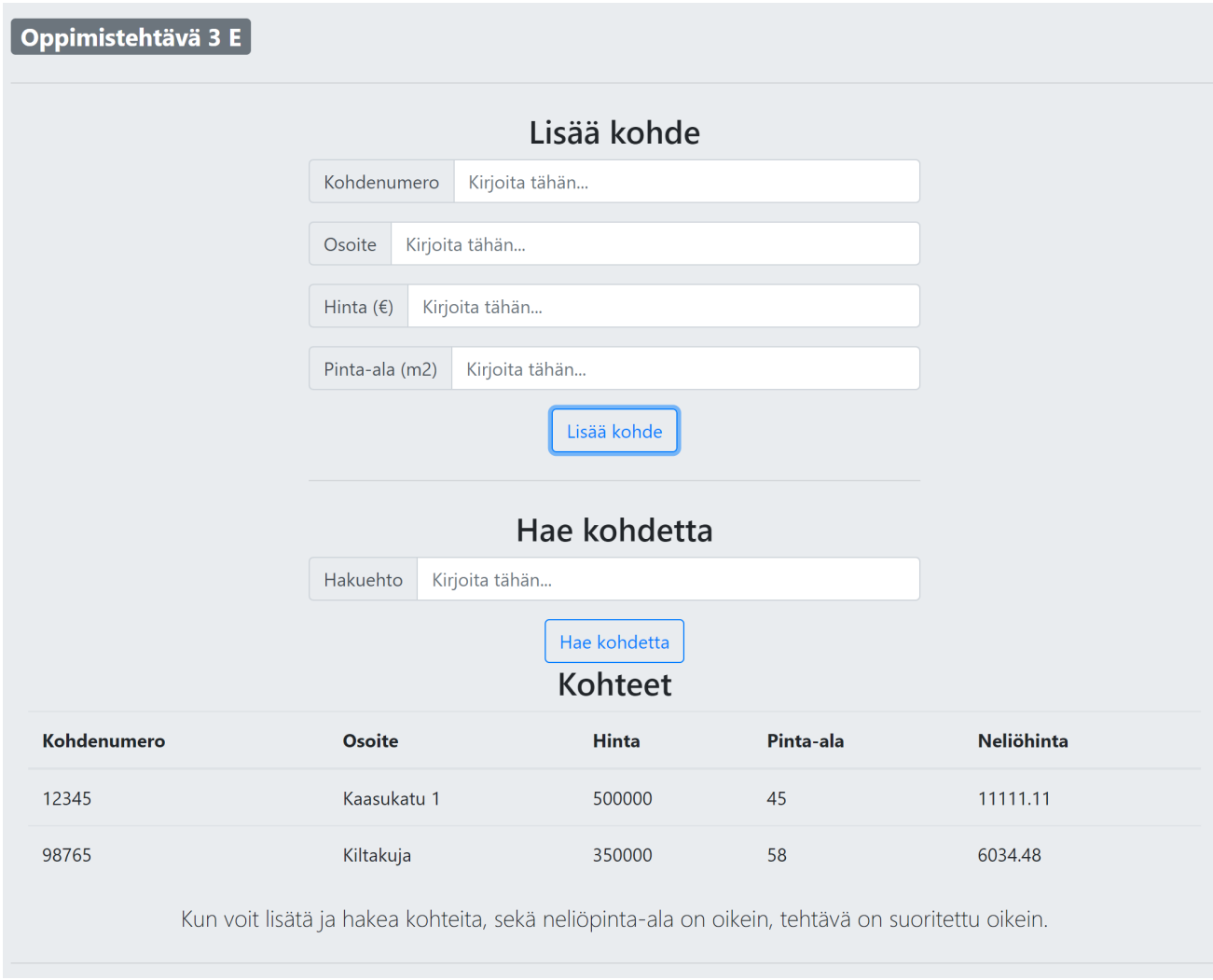
Luo funktio: **teeKohde(kohdenumero, osoite, hinta, pintaala)**, joka palauttaa kohteen tiedot objektina, kohteen kenttiä ovat:

- kohdenumero
- osoite
- hinta
- pintaala
- neliohinta

Funktion siis täytyy laskea kohteen neliöhinta ja asettaa se muiden tietojen mukana objektille.

Kiinteistövälittäjän täytyy myös voida etsiä lisättyjä kohteitaan. Ohjelma hakee lisätyistä kohteista ne, joiden tiedot sopivat hakuehtoihin. Sinun tehtäväsi on tuoteuttaa funktio: **haeKohde(kohde, hakuehto)**, joka saa parametrinaan kohde-objektin, joka on edellämainittua muotoa, sekä hakuehdon, joka on käyttäjän kirjoittama merkkijono. haeKohde-funktion tulee palauttaa boolean true, mikäli käyttäjän kirjoittama stringi sisältyy kohteen kohdenumeroon TAI osoitteeseen.

Voit käyttää hakuehdon tarkastamisessa includes-funktiota, jolla voit tarkastaa stringistä, sisältyykö jokin toinen stringi tähän, esim 'sisältyykö'.includes('isä') === true. Voit myös halutessasi muuttaa funktion sisällä kohdenumeron, osoitteen ja hakuehdon pieniin kirjaimiin funktiolla toLowerCase, jolloin haku toimii kirjainkoosta huolimatta.



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