

a JavaScript funktiot

JavaScript on funktionaalinen kieli. Funktiot ovat siinä keskeisessä roolissa. JavaScriptillä ei voi oikeastaan tehdä paljon kummoista ilman funktioita. Sen vuoksi olemme jo ehtineet luoda omia funktioita ja ajaa niitä painonappulasta käynnistäen onclick –tapahtuman käsittelijän käynnistämänä web –sivuilla. Nyt syvennymme funktioihin tarkemmin.

Funktiolla ohjelmoinnissa tarkoitetaan selkeästi roolitettua ohjelmaa, joka on helposti ymmärrettävissä. Se siis suorittaa selkeän tehtävän tai laskennan. Esimerkiksi keskiarvon laskenta olisi tällainen, josta saisi tehtyä hyvän funktion. Keskiarvoa tarvitaan erilaisissa tilanteissa, kuten hintojen keskiarvon laskentaan, pituuksien keskiarvon tai nopeuksien keskiarvon tms. laskemiseen. Uudelleen käytettävyyttä siis tällaiselle funktiolle olisi. JavaScriptissä tällaisen funktion määrittely näyttäisi seuraavalta:

```
function keskiarvo(numero1, numero2) {  
    var keskiarvo;  
    keskiarvo = (numero1+numero2)/2;  
    return keskiarvo;  
}  
var karvo = keskiarvo(5,6);
```

JavaScriptin avainsanalla function aloitetaan uuden funktio –ohjelman ohjelmointi. Funktion nimi kirjoitetaan heti function avainsana jälkeen, joka yllä olevassa tapauksessa on keskiarvo. Sitä seuraavat sulkeet () kertovat itse asiassa viimeistään JavaScript tulkille, että kyseessä on funktio. Funktion siis tunnistaa sulkeista (). Näiden sulkeiden sisällön voi jättää tyhjäksi niin kuin olemme usein jo ehtineet tehdä tällä kurssilla. Mutta sinne voi myös antaa lähtötiedot, joita funktio voi käyttää tuloksen laskentaan. Yllä olevassa keskiarvo funktiossa on annettu funktiolle kaksi lähtötietoa numero1 ja numero2. Ne ovat keskiarvo funktion ajon aikana olemassa olevia funktiotason muuttujia. Niitä ei ole enää olemassa, kun keskiarvo funktion ajo loppuu. Keskiarvo funktiomme voi siis laskea vain kahden luvun välisen keskiarvon tässä vaiheessa. Myöhemmin kun tutustumme taulukoihin, voimme antaa niin monta numeroa kuin tarvitsemme funktiomme laskettavaksi. Mutta tässä vaiheessa joudumme vielä tyytymään kahteen numeroon. Funktion sisällä ensimmäisenä suoritettavana rivinä on `var keskiarvo;` Siinä otetaan käyttöön eli määritellään (define) uusi funktiotason muuttuja nimeltä keskiarvo. Se on olemassa myös vain tämän funktion ajon ajan. Sitä seuraavalla rivillä lasketaan numero1:n ja numero2:n välinen yhteenlasku, jonka jälkeen se jaetaan 2:lla. Jos funktiomme olisi käynnistetty lauseella: `keskiarvo(5, 6)` laskennan tulos olisi 5.5 joka tallennettaisiin muuttujan keskiarvo arvoksi. Funktion kolmannella eli viimeisellä rivillä palautetaan return lauseen avulla keskiarvo muuttujan arvo sitä kutsuneelle ohjelmalle, joka on yllä olevassa ohjelmassa heti funktiota seuraava rivi eli `var karvo = keskiarvo(5,6);` Funktio palauttaa siis arvon 5.5 ja tallentaa eli sijoittaa sen muuttujan karvo arvoksi.

Huomaa, että määrittelemäämme keskiarvo funktiota ei suoriteta, jollei sitä käynnistetä. Keskiarvo funktiomme saa ajettua lauseella:
`keskiarvo(10,20)`

Koska funktiomme palauttaa return lauseella tuloksen on tulos saatava talteen, esimerkiksi

muuttuun. Tämä onnistuu muuttamalla lausetta seuraavanlaiseksi:
var karvo = keskiarvo(10,20);

JavaScriptissä voi siis funktion saada palauttamaan kutsuvalle ohjelmalle tuloksen return avainsanan avulla. Lisäksi funktion ajettava ohjelma laitetaan aaltosulkujen {} väliin, kuten yllä olevissa esimerkeissä on tehty.

```
function nimi1() {  
    // Funktion sisältä ohjelma tulee tähän  
    // return lause tulee yleensä loppuun ja sillä voi palauttaa tuloksen  
    // tätä funktiota kutsuneelle ohjelmalle.  
}
```

Kun funktio on saatu tehtyä, niin sen voi käynnistää eli ajaa joko painonappulasta vaikkapa onclick tapahtuman käsittelijän avulla. Toinen vaihtoehto on se mitä käytettiin keskiarvo funktiomme yhteydessä eli kutsumalla funktiota sen nimellä.

Seuraavaksi voisimme katsoa, miten funktion saa käynnistettyä Web sivulla olevan painonappulan avulla. Tässä käytämme html kielen <button> -elementtiä. Siitä nimittäin löytyy onclick, joka käynnistyy kun painonappulaa osoittaa ja klikkaa hiiren vasenta korvaa. Tämä onnistuu kun button elementti on seuraavanlainen: <button onclick="funktionNimi()" ></button>

Alla olevassa esimerkissä on painonappula käynnistää klikattaessa laskeAleHinta() funktion. Kyseinen funktio löytyy omasta JavaScript tiedostosta, joka puolestaan on kerrottu script -elementissä src attribuutille:

```
<script type="text/javascript" src="ohjelma1.js"></script>
```

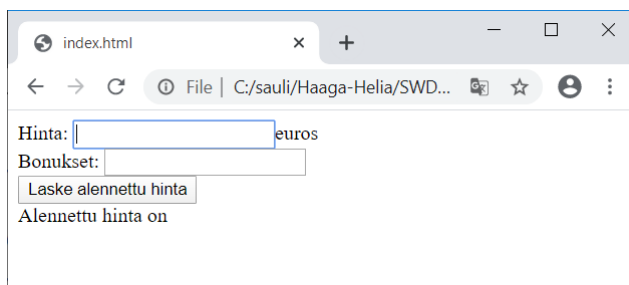
Seuraavaksi alla on esitetty HTML sivu, joka löytyy index.html tiedostosta sivuston omasta kansioista:

```
<!DOCTYPE html>  
<html lang="fi">  
  <head>  
    <meta charset="utf-8" />  
  </head>  
  <body>  
    <script type="text/javascript" src="ohjelma1.js">  
    </script>  
    Hinta: <input id="hinta" type="text" />euros<br />  
    Bonukset: <input id="bonus" type="text" /><br />  
    <button onclick="laskeAleHinta()" >  
      Laske alennettu hinta</button><br />  
    Alennettu hinta on <div id="result"></div>  
  </body>  
</html>
```

JavaScript ohjelma tosiaan on script elementissä kerrottu löytyvän niin sanotusta ulkoisesta tiedostosta, joten seuraavaksi tässä on esitetty tämä on ohjelma1.js tiedoston sisältö. Sieltä löytyy tarvitsemamme laskeAleHinta() funktion määrittely, jotta sen suorittaminen onnistuu:

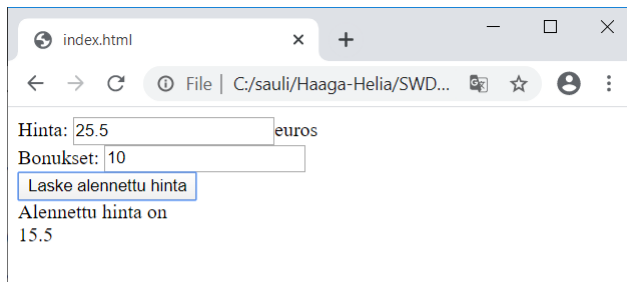
```
function laskeAleHinta() {  
    var hinta = document.getElementById('hinta').value;  
    var bonus = document.getElementById('bonus').value;  
    var aleHinta = hinta - bonus;  
    document.getElementById('result').innerHTML=aleHinta;  
}
```

Web-sivu näyttäisi seuraavanlaiselta:



Tämän ohjelman ajaminen etenisi seuraavalla tavalla web selaimessa. Käyttäjä kirjoittaa ensin hinta tekstilaatikkoon haluamansa hinnan (esim. 25.5), jonka jälkeen voi antaa bonukset (esim. 10). Tämän jälkeen käyttäjä klikkaa Laske alennettu hinta painonappulaa, joka käynnistää laskeAleHinta() nimisen funktion. Tämän jälkeen ohjelman ajo siirtyy laskeAleHinta() funktioon. Sieltä suoritetaan ensimmäiseksi rivi `var hinta = document.getElementById('hinta').value;` joka hakee samaisen web –sivun hinta id-nimisestä tekstielementistä käyttäjän syöttämän arvon. SE talletetaan eli sijoitetaan (assign) muuttujan hinta arvoksi. Samoin tehdään bonus muuttujan määrittelyn jälkeen heti seuraavaksi. Tähän muuttujaan talletetaan ajon ajaksi käyttäjän syöttämä bonus. Seuraavalla eli funktiomme kolmannella rivillä määritellään uusi aleHinta niminen muuttuja, jonka arvoksi tulee hinta – bonus vähennyslaskun arvo. Koska tässä esimerkissä käyttäjä syötti hinnan arvoksi 25.5 ja bonuksen arvoksi 10, tulee aleHinta muuttujan arvoksi 15.5. Funktion viimeisellä rivillä eli lauseessa `document.getElementById('result').innerHTML=aleHinta;` etsitään JavaScriptin built-in olion eli document olion getElementById() funktion avulla html elementti, johon voimme tulostaa saadun laskenta tuloksen web –sivulle. Sivultamme löytyy div elementti nimeltä (id nimeltään) result. Jos annamme tämän nimen getElementById() funktion lähtöarvoksi, löydämme kyseisen div elementin sivultamme, ja voimme tulostaa tai tarkemmin sanottuna sijoittaa = operaattorin eli sijoitus operaattorin avulla sinne haluamamme tuloksen. Tässä tapauksessa siis aleHinta muuttujan sisältämän arvon, joka on 15.5. Alla on esitetty miltä lopputulos näyttää Web

selaimessa:



b JavaScript built-in funktiot

JavaScriptistä löytyy myös siihen valmiiksi tehtyjä (built-in) funktioita. Olemme ehtineet jo tutustua joihinkin niistä, kuten document:in getElementById() –metodiin. Oliossa olevia funktioita kutsutaan metodeiksi. Myös console() metodi on samassa mielessä tullut tutuksi. JavaScriptissä valmiit siinä olevat funktiot eivät ole missään erityisasemassa edellisessä kappaleessa tehtyjen omien funktioiden kanssa. JavaScriptin tulkki kohtelee niitä tasa-arvoisesti.

Seuraavassa katsomme muutamia valmis funktioita ja mitä niillä pystyy tekemään.

Desimaalinumeron pyöristäminen kokonaisnumeroksi (Math.round())

Math:in round() funktiolla voi pyöristää desimaalinumeron lähimpään kokonaisnumeroon.

Esimerkiksi:

```
var numero1 = 5.123;  
var numero2 = Math.round(numero1);
```

pyöristää numero1 muuttujan sisältämän 5.123 numeron kokonaisnumeroksi, joka on 5 sillä se on lähin kokonaisnumero. Tämä arvo sijoitetaan eli talletetaan ajon ajaksi numero2 muuttujaan.

Jos ohjelma olisikin seuraavanlainen:

```
var numero1 = 5.767;  
var numero2 = Math.round(numero1);
```

5.767 pyöristettäisiin 6:een sillä se on nyt lähin kokonaisnumero.

Satunnaisluvun generointi (Math.random())

Math:in random() funktiosta löytyy muiden matematiikka funktioiden (metodien) ohella random() –metodi, jolla saa generoitu satunnaista desimaalinumeron 0:n ja 1:n väliltä. Esimerkiksi:

```
var satunnaisluku1 = Math.random();
```

toimisi siten, että Math.random() generoisi minkä tahansa desimaaliluvun 0:n ja 1:n väliltä ja sijoittaisi sen satunnaisluku1 muuttujan arvoksi. Jos haluaa generoida esimerkiksi satunnaisia numeroita 5:n ja 10:n väliltä, täytyy tietysti lisätä lausekkeeseen 5:n ja kertoa satunnaisluku generaattorin antama satunnaisluku 10:llä:

```
var satunnaisluku1 = Math.random()*10 + 5;
```

nyt lauseemme generoi satunnaisia desimaalinumeroita 5:n ja 10:n väliltä. Jos haluaa, että ne ovatkin kokonaisnumeroita, voi satunnaisluvun tietysti pyöristää lähimpään kokonaisnumeroon:

```
var satunnaisluku1 = Math.round(Math.random()*10 + 5);
```

Desimaaliluvun pyöristäminen haluttuun desimaali tarkkuuteen (.toFixed())

JavaScriptistä löytyy jokaisesta muuttujasta aina joukko erilaisia funktioita valmiina. Yksi esimerkki tällaisesta valmis funktiosta on .toFixed(). Sen avulla voi antaa mihin desimaali tarkkuuteen desimaalinumeron haluaa pyöristää. Esimerkiksi:

```
var n1 = 75.4544;  
console.log(n1.toFixed(2));
```

pyöristää desimaali numeron 75.4544 numeroksi 75.45 ja tulostaa sen web –selaimen konsoli-ikkunaan.

JavaScriptissä on muitakin vastaavanlaisia valmis funktioita kuin `toFixed()`. Niistä joitain seuraavassa.

Tekstin muuttaminen isoiksi kirjaimiksi (`.toUpperCase()`)

Muuttujassa olevan tekstin voi muuttaa isoilla kirjaimilla olevaksi `.toUpperCase()` funktiolla.

Esimerkiksi:

```
var nimi1 = "Matti";  
console.log(nimi1.toUpperCase());
```

`nimi1` muuttujan sijoitettu merkkijono eli teksti `Matti` muutetaan seuraavalla rivillä isoilla kirjaimilla kirjoitetuksi, eli `MATTI`. Tämä tulos tulostetaan web-selaimen konsoli-ikkunaan.

Tekstin muuttaminen pieniksi kirjaimiksi (`.toLowerCase()`)

Tekstin saa tietysti muutettua niin halutessaan myös pieniksi kirjaimiksi. Funktio `.toLowerCase()` tekee tämän. Esimerkiksi:

```
var nimi1 = "Matti";  
console.log(nimi1.toLowerCase());
```

muuttaa tekstin `Matti` kokonaan pienille kirjaimille, eli `matti`.

Tekstin pituus merkkeinä (`.length`)

Tekstissä olevien kirjainten tai merkkien lukumäärän saa selville `.length()` –funktiolla. Sitä voi käyttää esimerkiksi, muuttujassa olevan tekstin pituuden selvittämiseen:

```
var nimi1 = "Matti";  
console.log(nimi1.length);
```

yllä oleva ohjelma tulostaa konsoli-ikkunaan 5, sillä muuttujassa `nimi1` olevassa tekstissä eli `Matti:ssä` on 5 merkkiä.

Halutun merkin tulostaminen tekstistä (`.charAt()`)

Tekstissä olevia merkkejä voi myös tulostaa haluamastaan paikasta tekstistä. JavaScriptissä jokaisella tekstin tai merkkijonon merkillä on järjestysnumero (indeksi numero). Jos teksti olisi vaikkapa `Kotikatu`, niin sen järjestysnumero merkeille olisivat:

```
Kotikatu  
01234567
```

Merkkijonossa ensimmäisenä oleva `K` on siis järjestysnumeroltaan 0, eikä 1. Jos siis haluaisimme tulostaa merkkijonon ensimmäisen merkin, saamme sen selville lauseella:

```
var katuosoite = "Kotikatu";  
console.log(katuosoite.charAt(0));
```

jossa tulostetaan `katuosoite` muuttujaan sijoitetun arvon ensimmäinen merkki eli `K` seuraavalla rivillä konsoli-ikkunaan.

Halutun merkkijonon selvittäminen, että löytyykö jostain toisesta merkkijonosta (.indexOf())

Jos esimerkiksi muuttujassa osoite on tallennettu Kotikatu:

```
let osoite = "Kotikatu";
```

Voi selvittää, onko tässä merkkijonossa Kotikatu merkkijono katu, seuraavalla tavalla:

```
osoite.indexOf("katu")
```

Lause palauttaa vastauksenaan indeksi kohdan, josta katu alkaa merkkijonossa Kotikatu. Merkkijono katu alkaa indeksi kohdasta 4, joten vastauksena tulee 4.

```
      Kotikatu
indeksi kohta 01234567
```

Sen voisi tallentaa esimerkiksi vaikkapa muuttujaan talteen. Seuraavalla tavalla:

```
let indeksiKohta = osoite.indexOf("katu");
```

Muuttujan indeksiKohta data-arvoksi tulisi siis 4.

Merkkijonon osan tulostaminen (.substr())

Merkkijonon eli tekstin osan voi myös tulostaa. Tähän tarkoitukseen on JavaScriptissä olemassa .substr() (substring) –metodi. Jälleen käytetään edellisessä kappaleessa esitettyä järjestysnumerointia eli indeksointia merkeille:

```
var katuosoite = "Kotikatu";
console.log(katuosoite.substr(4,3));
```

tulostaa konsoli-ikkunaan kat, sillä k kirjain on merkkijonossa 4:ssä indeksi numerossa. Funktion seuraavaksi lähtötiedoksi annettu 3:n puolestaan kertoo, että halutaan ottaa seuraavat 3 kirjaista eli kirjaimet kat. Lopuksi ne tulostetaan konsoli-ikkunaan.

Tämän hetken päivämäärä (Date())

Date():llä voi luoda ohjelman ajohetken päivämäärä talteen ajon ajaksi. Esimerkiksi:

```
var currentDate = new Date();
```

luo currentDate nimisen muuttujan (tarkemmin sanoen olion) ja tallentaa siihen ajo hetken päivämäärän. Jos lause ajettaisiin esimerkiksi 5.3.2019 kello 9:20 sijoitettaisiin currentDate muuttujaan 5.3.2019 9:20

Päivämäärän sisältämän vuoden tulostaminen (getFullYear())

Päivämäärän sisältämän vuoden saa selville getFullYear() –funktion avulla. Esimerkiksi:

```
var currentDate = new Date();
console.log(currentDate.getFullYear());
```

jos ohjelman ajo hetki olisi jälleen 5.3.2019 9:20, niin getFullYear() –funktio palauttaisi arvonaan 2019, joka tulostettaisiin seuraavaksi konsoli-ikkunaan. Vastaavanlaiset metodit löytyvät päivämäärälle ja kuukausille (kuukausissa yllättävästi JavaScriptissä kuukausi numerointi alkaa nollasta, eli tammikuuta esimerkiksi vastaa järjestysnumero 0).

Desimaalinumeron muuttaminen kokonaisnumeroksi (parseInt())

Desimaalinumeron voi muuttaa halutessaan kokonaisnumeroksi. JavaScriptin parseInt() on tehty tähän tarkoitukseen. Sitä voi käyttää esimerkiksi seuraavalla tavalla:

```
var arvosana = 4.211;  
console.log(parseInt(arvosana));
```

muuttujassa arvosana oleva 4.211 muutetaan kokonaisnumeroksi seuraavalla rivillä. Koska 4 on lähin kokonaisnumero, tulostetaan sen vuoksi konsoli-ikkunaan numero 4.

Merkkijonon näkymättömien merkkien poistaminen merkkijonosta (.trim())

Merkkijonossa olevia näkymättömiä merkkejä voi poistaa .trim() –metodilla merkkijonon alusta ja lopusta. Näkymättömiä merkkejä ovat esimerkiksi välilyönnit, joita käyttäjät helposti vahingossa kirjoittavat esimerkiksi käyttöliittymien tekstilaatikoihin. Tässä mielessä .trim() on tärkeä metodi sovellutuksissasi. Sitä voi käyttää esimerkiksi:

```
var sukunimi = "  Mattila ";  
console.log(sukunimi.trim());
```

sukunimi muuttuukaan ajon ajaksi talletettu teksti Mattila sisältää kaksi välilyöntiä merkkijonon alussa ja yhden välilyönnin merkkijonon lopussa. Seuraavalla rivillä .trim() metodi poistaa ne ja lopputuloksen tulostetaan konsoli-ikkunaan teksti Mattila.

c Lisämateriaalia aiheesta kiinnostuneilla

Keskiarvo funktion voisi muuttaa laskemaan kahden numeron sijaan niin monta numeroa kuin haluaa myös seuraavan esimerkin mukaisella tavalla. Vaihtoehtoja siis riittää. Seuraavassa esimerkissä on tehty eli määritelty uusi funktio, jonka nimi on add.

- Example 1: Laske yhteen mikä tahansa määrä numeroita käyttäen ns. built-in arguments objektia JavaScriptissä:

```
function add() {  
    var sum = 0;  
    for (i=0; i<arguments.length; i++) {  
        sum += arguments[i];  
    }  
    return sum;  
}
```

Voit ajaa tämän add nimisen function alla olevalla tavalla:

```
var sum1 = add(1,2,3,4,5,6);  
alert(sum1);
```

Katsomme tätä tarkemmin myöhemmin kun opimme käyttämään toistorakenne, kuten for –lausetta.

- Example 3: Function literals:

Here is variable, which is of type function:

```
var addMe = function() {  
    return arguments[0] + arguments[1];  
}
```

```
var mySum2 = addMe(1,2);  
alert(mySum2);
```

Now the mySum2 variable is of type Function objects and invokes function literal addMe.

This is very handy, because you can pass function literals as arguments to other function. So, this is used a lot in JavaScript.

- Example 4: Anonymous functions

Anonymous functions are used a lot for example in jQuery.

```
var mySum3 = (function() {  
    return arguments[0] + arguments[1];  
})(1,2);  
alert(mySum3);
```

- Example 5: Anonymous functions

This is also possible.

```
alert( (function() {  
    return arguments[0] + arguments[1];  
}) (1,2) );
```

-> So, function do not necessary need a name to be able to pass their value around.

-> Functions are essential to JavaScript like classes are essential to Java or C#.

