a IF -lause

Usein on tarvetta tarkistaa esimerkiksi käyttäjän sovellutukseen antama arvo. Sehän voi olla ohjelman toiminnan kannalta huono ja voi peräti kaataa sen. Annettu arvo voi olla myös sellainen, että se ei ole sallittu. Esimerkkinä tästä voisi olla vaikkapa hinta. Jos hinnaksi antaa negatiivisen luvun (esim -16.5) niin tuotteiden yhteenlaskennassa tulisi väärä tulos. Muita vastavanlaisia tilanteita löytyy runsaasti lisää. Tällaisten tilanteiden selvittämiseen löytyy ohjelmointikielistä valintalause, joita on useita erilaisia. Tässä tutustumme if –lauseeseen. Edellä esitetyn tilanteen lisäksi if –lauseella voi rakentaa myös toimintalogiikkaa ohjelmistoonsa, joten se on tässä mielessä monipuolinen lause.

Seuraavaksi katsomme, miten if –lausetta voi käyttää erilaisissa tilanteissa. Jos kysytään vaikkapa henkilön pituutta, annettu arvo pitäisi olla tietysti positiivinen numero. Negatiivisen numeron antaminen pituudeksi ei ole oikein. Tällaisen tarkistuksen voisi tehdä JavaScriptissä lauseella:

```
var pituus = 180;
if (pituus < 0 ) {
    console.log("Anna positiivinen numero! ");
} else {
    console.log("Numero on sallittu numero pituudeksi. ");
}
console.log("Loppu");
```

Yllä olevassa ohjelmassa asetetaan eli talletetaan ajon ajaksi muuttujan pituus arvoksi 180. Seuraavaksi if lauseessa ensimmäiseksi tarkistetaan, onko pituus muuttujan arvo pienempi kuin nolla. Tämä ehto ei ole tosi tässä esimerkissä, sillä 180 ei ole pienempi kuin nolla. Sen vuoksi hypätään if –lauseen ensimmäinen haara yli ja jatketaan if lauseen seuraavaan haaraan. Seuraavana on else, joka suoritetaan aina kun edelliset haarat eivät ole tosia, kuten tässä tapauksessa on tilanne. Sen vuoksi suoritetaan siis nyt else haara. Koska lauseita ei ole kuin vain yksi eli `console.log("Numero on sallittu numero pituudeksi. ");` suoritetaan vain se. Tämän jälkeen hypätään koko if rakenteesta pois ja jatketaan if –rakennetta seuraavasta lauseesta, joka yllä olevassa esimerkissä on `console.log("Loppu");`

Seuraavaksi tarkastelemme tilannetta, jos pituus muuttujan arvo olisikin negatiivinen numero:

```
var pituus = -5.5;
if (pituus < 0 ) {
    console.log("Anna positiivinen numero! ");
} else {
    console.log("Numero on sallittu numero pituudeksi. ");
}
console.log("Loppu");
```

Nyt ohjelman suoritus etenisi siten, että heti alkuun tarkistettaisiin onko pituus muuttujan arvo pienempi kuin nolla. Koska nyt pituus muuttujan arvo on -5.5, pitää tämä ehto tällä kerralla paikkansa (true). Sen vuoksi tällä kertaa suoritetaan if –lauseen ensimmäisessä haarassa olevat kaikki lauseet. Niitä on vain yksi eli `console.log("Anna positiivinen numero!");`

joka siis suoritetaan. Tämän jälkeen hypätään koko if lauseesta pois, sitä seuraavalle riville ja jatketaan siitä eteenpäin ohjelman suorittamista. Eli suoritetaan lause

```
console.log("Loppu");
```

if –lauseeseen voi ohjelmoida myös useita eri haaroja yllä olevan kahden haaran tai vaihtoehdon sijaan. Esimerkkiimme voisi negatiivisten numeroiden testauksen lisäksi lisätä myös testaus, joka löytää nollan jos sellainen on annettu pituudeksi.

```
var pituus = 0;
if (pituus < 0 ) {
    console.log("Anna positiivinen numero! ");
} else if (pituus == 0) {
    console.log("Annoit nollan pituuden arvoksi! ");
} else {
    console.log("Numero on sallittu numero pituudeksi. ");
}
console.log("Loppu");
```

Nyt esimerkki ohjelmamme ajaminen etenisi edelliseen verrattuna hiukan eri tavalla. Koska pituus muuttujan arvo on nyt nolla, ei if lauseen ensimmäinen testaus (0<0) pidä paikkaansa (false). Sen vuoksi ensimmäinen haara if –lauseessa hypätään yli ja edetään seuraavaan testiin. Tässä testissä verrataan, että onko pituus muuttujan arvo sama kuin nolla (0 == 0). Tämä pitää paikkaansa (true). Tämän vuoksi tämä haara if –lauseesta suoritetaan, eli tulostetaan konsoli –ikkunaan teksti Annoit nollan pituuden arvoksi! Seuraavaksi ohjelman suoritus etenee siten, että koko if –rakenteesta hypätään pois ja jatketaan sitä seuraavasta lauseesta eteenpäin, joka tässä tapauksessa on ”Loppu” -tekstin tulostus Web selaimen konsoli –ikkunaan.

if –lauseessa voi siis ehtojen avulla tarkistaa täyttyykö jokin ehto vai ei. Näitä ehtoja ohjelmoidaan seuraavilla operaattoreilla:

== tai ===	yhtäsuuruus
!= tai !===	eri suuruus
<	pienempi kuin
<=	pienempi tai yhtäsuuri kuin
>	suurempi kuin
>=	suurempi tai yhtäsuuri kuin

Lisäksi voi ehdon ohjelmoida useasta eri eri ehdosta yhdistelemällä. Silloin tarvitsee seuraavia operaattoreita:

&&	JA eli AND operaattori
||	TAI eli OR operaattori

Lisäksi on ns. unaari operaattori:

!	EI eli NOT operaattori (muuttaa false:n true:ksi / true:n false:ksi)

JA (and) eli && operaattori on hyödyllinen operaattori esimerkiksi tilanteessa, kun haluaa selvittää, että kuuluuko annettu luku johonkin tiettyyn lukualueen. Esimerkiksi onko annettu ikä alaikäisen ikä vai aikuisen ikä. Seuraavaksi katsomme millainen on AND operaattorin niin sanottu totuustaulukko:

ehto1	ehto2	AND operaattorin tulos
true	true	true
false	true	false
true	false	false
false	false	false

Totuustaulukosta voi todeta sen, että molemmat ehdot ehto1 ja ehto2 on oltava tosia, jotta lopputulokseksi tulee true. Kaikissa muissa tapauksissa lopputulos on aina false. Hyvä muistisääntö.

AND operaattori on JavaScriptissä &&. Seuraavaksi voisimme katsoa, miten && -operaattorin avulla voi rakentaa esimerkiksi testin, jolla pystyy tarkistamaan onko annettu luku lukujen 18….30 välillä.

```
var ika = 23;
if(ika>=18 && ika<30) {
     alert("Nuori aikuinen");
}
```

Koska ika muuttujan arvo on 23, etenee if lauseen suoritus siten, että ensimmäiseksi tarkistetaan if –lauseen ensimmäinen testi. Tässä testissä on kaksi ehtoa, joista ensimmäisessä verrataan, onko ika suurempi tai yhtäsuuri kuin 18. Toisessa ehdossa verrataan, että onko ika pienempi kuin 30. Koska ika muuttujan arvo on 23, on ensimmäinen ehto seuraavanlainen 23>=18, joka pitää paikkaansa (true). Seuraavaksi edetään toisena olevan ehdon suorittamiseen, eli 23<30, joka myös pitää paikkaansa (true). Koska molemmat ehto1 ja ehto2 ovat nyt true, saadaan AND operaattorin lopputulokseksi siis true. Tämä tarkoittaa sitä, että tämä haara if lauseesta suoritetaan. Siis kaikki sieltä löytyvät lauseet. Nyt sieltä ei löydy kuin yksi lause eli `alert("Nuori aikuinen");` joka suoritetaan. Seuraavaksi hypätään koko if rakenteesta pois ja jatketaan sitä seuraavasta lauseesta eteenpäin.

TAI eli OR operaattorilla voi selvittää esimerkiksi täyttyykö jokin vaihtoehdoista vai ei. Sitä voi tietysti käyttää muuhunkin, mutta ainakin tällaisissa tilanteissa OR operaattori on käyttökelpoinen. JavaScriptissä OR –operaattori on ||. OR operaattorin totuustaulukko on puolestaan:

ehto1	ehto2	OR operaattorin tulos
true	true	true
false	true	true
true	false	true
false	false	false

Tästä totuustaulukosta huomaa helposti, että jos yksikin ehdoista on tosi on lopputulos aina tosi. Jos kaikki ehdot ovat epätosia (false) on lopputulos ainostaan silloin false. Tämäkin käy muistisäännöstä, joka kannattaa muistaa.

Seuraavaksi voisimme tarkastella, miten || -operaattoria voi käyttää vaikkapa sukupuolen tarkistamisessa. Sukupuolia on kolme vaihtoehtoa, eli nainen, mies, ei sukupuolta. Ne eivät voi olla voimassa yhtäaikaa. Ne ovat siis vaihtoehtoja. || -operaattori on kuin tehty tällaisen tilanteen testaamiseen. Seuraavassa esimerkissä katsomme, miten tämä tehdään käytännössä:

```
var sukupuoli = "mies";

if(sukupuoli===" nainen" || sukupuoli==="mies" || sukupuoli===
  "eisukupuolta") {
  alert("Sukupuoli on annettu oikein!");
}
```

Tutustumme nyt samalla myös === operaattoriin JavaScriptissä. Aikaisemminhan käytimme yhtäsuuruuden vartailuun == -operaattoria. JavaScriptistä löytyy myös toinen yhtäsuuruutta vertaileva operaattori eli ===. Se tekee vertailun niin sanotusti tiukasti, eli vertailtavat arvot on oltava täsmälleen samaa tietotyyppiä vertailtavan arvon lisäksi. JavaScriptin automaattista tyyppi muunnosta ei tehdä, kuten == -operaattorissa tehdään. === -operaattori toimii siis hyvin samankaltaisesti kuin muissa ohjelmointikielissä yhtäsuuruus operaattori (esim. Java, C, C++).

Nyt pääsemme katsomaan, miten esimerkki ohjelmamme yllä toimii. Esimmäiseksi luodaan muuttuja sukupuoli, jonka arvoksi talletetaan mies. Seuraavaksi ohjelman suoritus etenee seuraavalle riville, josta löytyy if –rakenne. Sen suoritus alkaa ensimmäisen haaran tai testin suorittamisella. Siitä löytyy peräti kolme ehtoa. Ensimmäinen ehto testaa onko sukupuoli yhtäsuuri kuin nainen. Koska sukupuoli muuttujan arvo on mies, on testi siis "mies" === "nainen", koska nämä merkkijonot eivät ole samoja on tuloksena false. Seuraavaksi edetään toiseen ehtoon, jossa testataan, onko sukupuoli muuttujan arvo mies. Koska sukupuoli muuttujassa on arvona mies, on testi siis "mies" === "mies". Tämä pitää paikkaansa eli tulokseksi tulee true. Vielä olisi kolmaskin ehto, jonka tulokseksi voi jo nähdä false:n. Sitä ei kuitenkaan suoriteta, sillä JavaScriptin tulkki osaa jo päätellä, että OR tulee päätymään tulokseen true. If –lauseemme ensimmäisen haaraan tulos on siis tosi, jonka vuoksi seuraavaksi suoritetaan tämä haara if –lauseesta. Sieltä löytyy vain yksi lause, jossa käyttäjälle

näytetään: Sukupuoli on annettu oikein! Seuraavaksi hypätään if –lauseesta pois ja jatketaan eteenpäin sitä seuraavasta rivistä eteenpäin.

If –lauseessa voi tietysti olla useita haaroja. Niitä voi rakentaa if –lauseeseensa tarvitsemansa määrän else if avainsanojen avulla. Seuraavaksi voisimme katsoa if –rakennetta, jossa on kolme eri haaraa tai vaihtoehtoa. Ensimmäisessä haarassa tarkistetaan, että annettu ikä ei ole negatiivinen numero. Seuraavassa haarassa tarkistettaan, että onko ika lukujen 0…150 välissä. Huomaa, että tähän lukualueeseen kuuluu nolla mukaan muttei 150. Viimeisenä tarkistetaan, että on ikä yhtäsuuri tai suurempi kuin 150:

```
var ika = 20;
if (ika<0) {
    alert("Negatiivinen ikä!");
}
else if (ika>=0 && ika<150) {
    alert("Ikä "+ika + " on hyväksyttävä ikä.");
}
else if (ika>=150) {
    alert("Ikä on liian suuri numero!");
}
```

Tämän esimerkin suoritus alkaa ika muuttujan määrittelyllä ja numeron 20 tallentamisella (sijoittamisella) sen arvoksi. Seuraavaksi edetään if lauseen suorittamiseen. Siitä suoritetaan ensimmäinen haara, eli tehdään vertailu 23<0. Tämä ehto ei pidä paikkaansa (false), joten hypätään if –rakenteen seuraavaan haaraan. Sieltä löytyy ehto, jossa testataan kuuluuko ika muuttujan arvo 0…150 väliin. Tässä on käytetty AND –operaattoria, joten ensimmäiseksi tehdään vertailu 23>=0, joka pitää paikkaansa (true). Seuraavaksi tehdään vertailu 23<150, joka sekin pitää paikkaansa (true). AND –operaattorin tulokseksi tulee siis true, jonka vuoksi if –lauseen tämä haara suoritetaan seuraavaksi. Sieltä löytyy alert viesti-ikkunan käynnistävä funktio, joka näyttää tekstin ”Ikä 23 on hyväksyttävä ikä.” Seuraavaksi hypätään koko if –lauseesta pois ja jatketaan eteenpäin sitä seuraavasta rivistä eteenpäin.

b Lisä aiheita kiinnostuneille

- Do not use == operator in JavaScript, instead use === operator (known as strict equality operator). This applies also to the !== operator. So, do not use != operator. (JavaScript: The Definitive Guide, s. 71)

- In JavaScript there is this unique idea of "truthy" and "falsy" values. Falsy value are the value false and also undefined, 0 and NaN (not a number, for example some number divided by zero is NaN). `null`, `0`, `-0`, `NaN` and `""` convert in JavaScript to `false`. Otherwise the value is considered "truthy". (JavaScript: The Definitive Guide, s. 40).

Example 5: Conditional operator/Ternary operator. Testing for "`truthy`" or "`falsey`".

```
var someValue = 24;
var myResult = "";
myResult=(someValue === 24) ? "equal" : "not equal";
alert(myResult);
```

-> "`truthy`" and "`falsey`" values are in JavaScript: `false`, `undefined`, `0` and `NaN` (Not a Number), otherwise the value is considered "`truthy`".

-> Try also for example in Firebug Console:

```
myResult = false ? "truthy" : "falsey";
falsey
```


```
myResult = typeof 24/"some text";
NaN

myResult = (24/"some text") ? "truthy" : "falsey";
falsey

myResult = undefined ? "truthy" : "falsey";
falsey
myResult = typeof someUndefinedVariable ? "truthy" : "falsey";
truthy
```

(Note this example! Why it returns truthy, even though the variable has not even been declared? The reason is that JavaScript considers `someUndefinedVariable` as a string, and therefore returns `truthy`.

- Now, try === and ==:

```
myResult = (3 == "3") ? "true" : "false";
true
```

This should not be the case. This is the reason, why you never should use == operator. JavaScript tries to coersion 3 and "3" together. It should not be doing this implicit type coersion. Luckly === does not do this, so use it instead of == -operator in JavaScript.

```
myResult = (3 === "3") ? "true" : "false";
false
```

Example 6: Switch statement.

```
var userName = "Rimmer";
switch (userName)
{
     case "Lister":
          document.write("This is Lister!");
          break;
     case "Rimmer":
          document.write("This is Rimmer!");
          break;
     default:
          document.write("This is default");
          break;
}
```

-> You should never depend on `document.write()` in a real application.