a Toistorakenne

Aina kun jotain tehtävää tehtäessä alkaa tuntua siltä, että samaa tulee toistettua useita kertoja, kannattaa miettiä tehokkaampaa tapaa. Ohjelmoinnissa vastaus tähän kysymykseen on toistorakenne. Nimensä mukaisesti sillä voi toistaa halutut tehtävät niin monta kertaa kuin on tarpeen. JavaScriptistä löytyy useita toistorakenteita: for, while, do while ja for in. Tällä kurssilla keskitymme for ja while toistorakenteisiin.

for toistorakenne
Toistorakenteista for soveltuu hyvin esimerkiksi taulukoiden ja olio tietorakenteiden läpikäyntiin. Sitä voi käyttää tietysti muuhunkin. Toistorakenteista for on jonkin verran rajoittuneempi kuin while rakenne. Sillä ei pysty tekemään ihan kaikkea, mitä while toistorakenteella pystyy. for rakenteen periaate on seuraavanlainen:

```
for (toistonAloitusArvo; lopetusEhto; kierrosLaskuri) {
      // Tänne tulevat toistettava lauseet
}
```

Esimerkiksi jos haluttaisiin vaikkapa tulostaa numerot 1:stä 5:een selaimen konsoli-ikkunaan, tehtäisiin se seuraavanlaisesti. Toisto siis halutaan aloittaa 1:stä, joten toiston aloitusarvoksi asetettaan 1. Tämä onnistuu luomalla (määrittämällä) uuden muuttujan (i) ja asettamalla sen arvoksi 1 (var i = 0).

```
for (var i = 1; lopetusEhto; kierrosLaskuri) {
      // Tänne tulevat toistettava lauseet
}
```

Seuraavaksi voidaan jatkaa lopetusehdon tekemiseen. Kyseessä on ehto, joita olemme jo tehneetkin if lauseen yhteydessä. Tässä haluamme, että toisto loppuu numeroon 5. Ehto on siis i<=5, eli niin kauan kuin muuttujan i arvo on pienempi tai yhtä suuri kuin 5, jatketaan toistamista.

```
for (var i = 1; i<=5; kierrosLaskuri) {
      // Tänne tulevat toistettava lauseet
}
```

Kolmanneksi laitamme kierroslaskurin kuntoon, jotta toistorakenne tietää kuinka monta kertaa se on toistanut. Joka kierroksen jälkeen siis kierroslaskurin arvoa on kasvatettava yhdellä. Se onnistuu lauseella i++. JavaScriptistä löytyy ++ operaattori, jolla voi kasvattaa muuttujan arvoa yhdellä. i++ tarkoittaa samaa kuin i = i + 1.

```
for (var i = 1; i<=5; i++) {
      // Tänne tulevat toistettava lauseet
}
```

Nyt toistorakenne aloittaa toiston 1:stä ja toistaa aina 5:een asti. Toistoja tulee siis 5 kertaa. Mutta toistorakenteemme ei vielä tee sen enempää. Tässä tehtävässähän halusimme tulostaa luvut 1:stä 5:een Web selaimen konsoli-ikkunaan. Konsoli-ikkunaan pystyy tulostamaan JavaScriptissä console.log() metodilla. Riittää kun log() metodin arvoksi antaa halutun muuttujan: console.log(i).

```
for (var i = 1; i<=5; i++) {
      console.log(i);
}
```

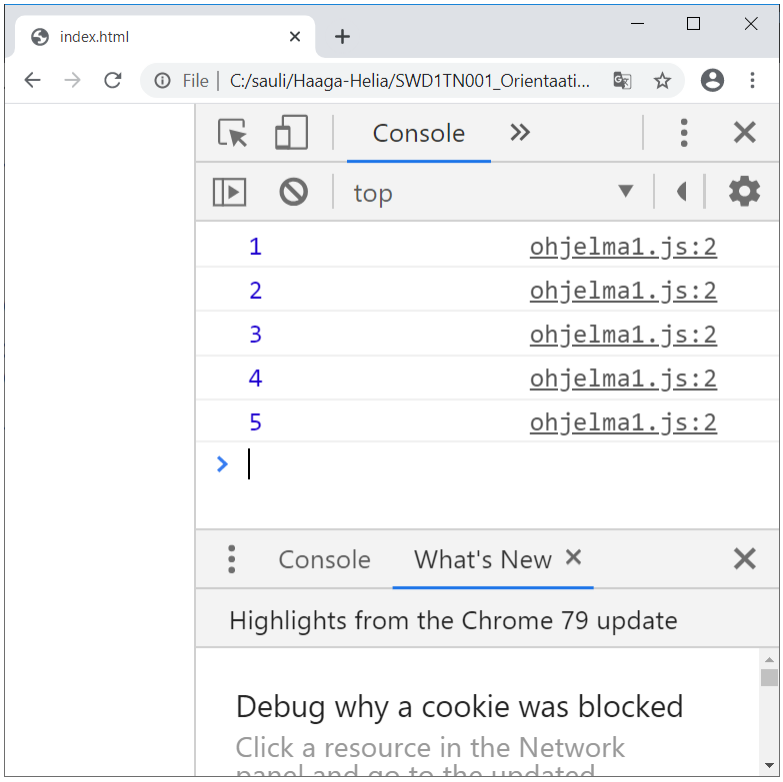
Jos tätä haluaa kokeilla Web selaimessa, tarvitaan HTML jonka sisältö on:

```
<!DOCTYPE html>
<html lang="fi">
      <head>
                <meta charset="utf-8" />
                <script src="ohjelma1.js"></script>
      </head>

   <body>

   </body>

</html>
```

<script> elementin src attribuutilla kerrotaan, mistä tiedostosta JavaScript ohjelma löytyy. Tässä tiedoston nimi on ohjelma1.js. Kyseisen tiedoston sisältö on:

```
for (var i = 1; i<=5; i++) {
      console.log(i);
}
```

Nyt tulos näyttää nyt Web selaimen konsoli-ikkunassa (Console) seuraavalta:



Esimerkki: Tulostus for lauseella Web sivulle

Jos edellisen esimerkin kaltaisen tehtävän tulostuksen haluaisikin tehdä Web sivulle, tarvitaan document.getElementById().innerHTML:ää. Sen avulla pystyy tulostaan tekstiä, muuttuja arvoja yms haluttuun HTML elementtiin. Kyseiselle HTML elementille on vain annettava yksikäsitteinen id nimi. Esimerkiksi elementille tämä onnistuisi lauseella: </result>. JavaScriptin getElementById() puolestaan voidaan laittaa etsimään kyseinen elementti sen nimen perusteella, lauseella: document.getElementById("result").innerHTML. Se hakee elementin niin sanotusta DOM puusta, jonka selain aina muodostaa näyttäessään web sivun käyttäjän selaimessa (Document Object Model, DOM). Seuraavaksi annamme innerHTML ominaisuudelle arvon, esimerkiksi: document.getElementById("result").innerHTML= " Hei";
Näillä HTML:n elementeillä ja JavaScriptin metodeilla pystymme tekemään seuraavanlaiset tiedostot:

HTML sivun tiedosto näyttäisi tällöin seuraavalta (tiedoston nimi voisi olla vaikkapa index.html tms):

```
<!DOCTYPE html>
<html lang="fi">
      <head>
                <meta charset="utf-8" />
                <script src="ohjelma1.js"></script>
      </head>

  <body>
      <button onclick="aja()">Aja</button>
      <div id="result"></div>

  </body>

</html>
```

Huomaa, että nyt ohjelma on tarkoitus käynnistää Aja painonappulalla. Se käynnistää aja() nimisen funktion. Tämä funktio täytyy olla nyt JavaScript tiedostossamme. Se sisältää toistorakenteemme ja tulostuksen elementtiin. Näin käytännössä tulostus Web sivulle saadaan onnistumaan.

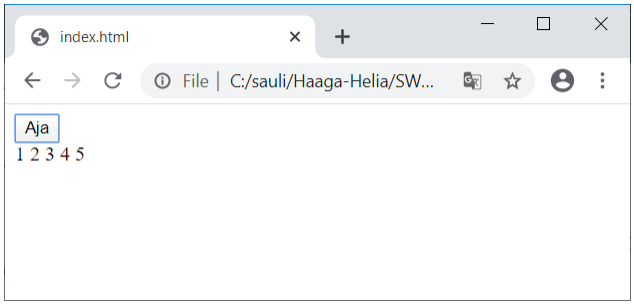
Vastaavasti JavaScipt tiedosto sisältää siis seuraavaa (ohjelmani1.js):

```
function aja() {
      var teksti = "";
      for (var i = 1; i<=5; i++) {
                teksti+=" " +i;
      }
      document.getElementById("result").innerHTML= teksti;
}
```

Nyt tosiaan kaikki JavaScript on sijoitettu aja() funktioon. Ensimmäiseksi tässä funktiossa määritellään teksti niminen muuttuja, jolle annetaan arvoksi tyhjä merkki (""). Tärkeä vaihe, jottei toistorakenteemme kaatuisi. Sehän alkaa lisäämään merkkejä (tekstiä) heti ensimmäisestä kierroksesta lähtien teksti muuttujaan. Ensimmäisellä kierroksella lisätään kyseiseen muuttujaan 1. Toisella kierroksella 2, joten teksti muuttujan arvo on nyt 1 2. Kolmannella kierroksella lisätään 3, joten teksti muuttujan arvo on 1 2 3. Näin jatketaan kunnes tulee viimeinen eli 5 toisto kierros, jonka lopuksi teksti muuttujan arvon 1 2 3 4 5. Ilman `teksti+=" " +i;` lauseessa olevaa `" "`, tulos olisi 12345. Välilyönnillä saamme numeroiden väliin välilyönnin, ja numerot näyttävät siltä kuin niiden pitää tässä näyttää. += operaattorilla puolestaan onnistuu välilyönnin ja i muuttujan arvon

lisääminen kierros kierrokselta teksti muuttujan arvoksi, siellä jo olevan tekstin (merkkijonon) perään.

Lopputulos näyttäisi selaimessa seuraavalta. Ohjelman saa käyntiin klikkaamalla Aja painonappulaa, jonka jälkeen ohjelma tulostaa merkkijonon 1 2 3 4 5 sivulle:



while toistorakenne

Ohjelmoinnissa while rakenne on kaikkein monipuolisin toistorakenne. Sillä pystyy ratkaisemaan kaikkein parhaiten erilaisia tehtäviä, joissa tarvitaan toistaa lauseita, tehdä laskentaa tms. Tämän vuoksi while:n käyttö kannatta osata. while rakenne on periaatteessa seuraavanlainen:

```
while(lopetusEhto) {
      // Toistettavat lauseet
}
```

Toiston aloitusarvoa ja kierroslaskuria ei siis välttämättä tarvitse antaa. Usein ne kuitenkin halutaan antaa. Mutta tosiaan, while rakennetta voi käyttää vaikkapa seuraavaan tapaan:

```
while(true) {
      // Toistettavat lauseet
}
```

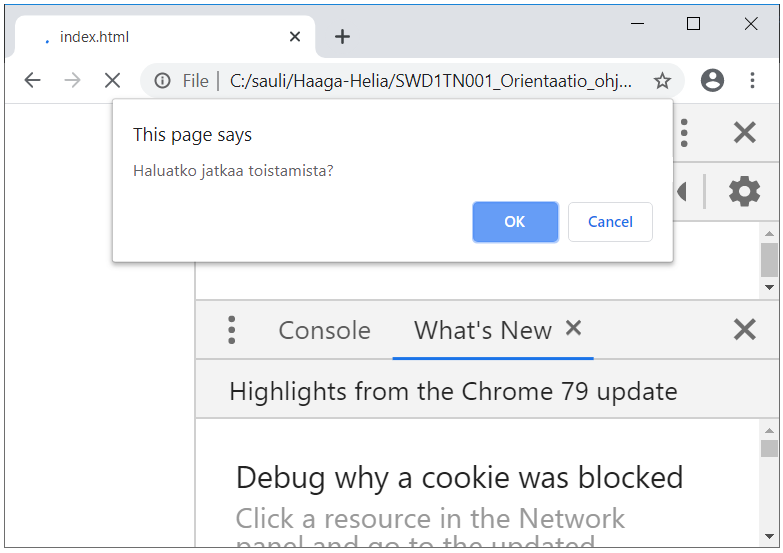
Nyt lopetusehdoksi on annettu suoraan true, eli toistorakenteemme toistaa loputtomasti. Tätä tavallisesti ei koskaan haluta tapahtuvan. Tässä mielessä for rakenne on yksinkertaisempi, sillä sen saa helpommin ohjelmoitua oikein. Yllä olevaan while rakenteeseen tarvitaan siis jokin tapa lopettaa toisto jossain vaiheessa. Se voidaan tehdä while rakenteen sisällä olevalla if –rakenteella tai vaikkapa confirm() funktiolla. JavaScriptin confirm() nimittäin palauttaa true:n jos käyttäjä klikkaa tämän ikkunan OK nappulaa. Vastaavasti Cancel nappula palauttaa false:n. Eli niin kauan kuin käyttäjä jaksaa klikata OK nappulaa jatketaan toistamista. Heti kun käyttäjä klikkaa Cancel nappulaa, toistaminen lopetetaan. Nyt ohjelmamme näyttäisi seuraavalta:

```
while(confirm("Haluatko jatkaa toistamista?")) {

}
```

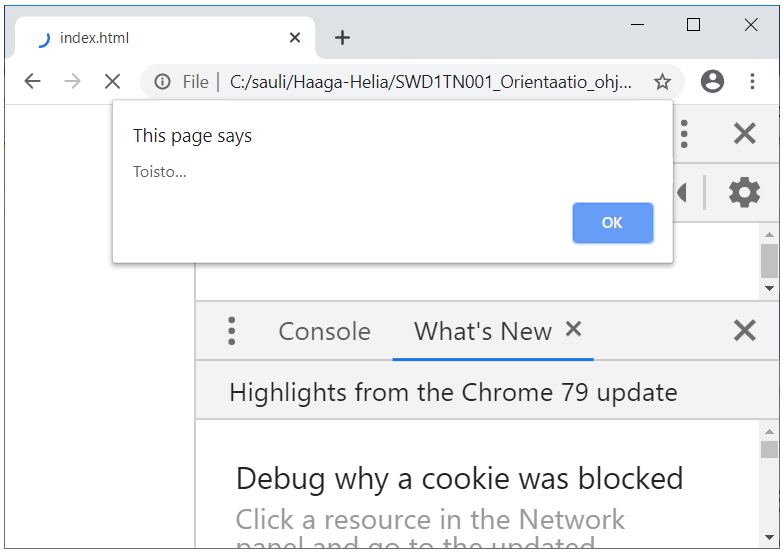
Nyt ohjelmamme saadaan haluttaessa lopettamaan toistaminen. Se ei vain tee toistaessaan vielä yhtään mitään. JavaScriptin alert() funktiolla voimme näyttää käyttäjälle vaikkapa viesti-ikkunan esimerkin vuoksi. Nyt toistorakenteemme olisi valmis ja se näyttäisi seuraavalta:

```
while(confirm("Haluatko jatkaa toistamista?")) {
      alert("Toisto…");
}
```

Ohjelmamme toimisi seuraavasti:



Ja jos OK nappulaa klikattaisiin, saataisiin:



Tästä esimerkistä käynee selville, että while on melko monipuolinen toistorakenne. Sitä voi tietysti käyttää for rakenteessa esitetyn kaltaisten tehtävien ratkaisemiseen. Jos vaikkapa haluaisimme tulostaa kaikki numerot 10:stä 20:een, onnistuisi sekin. Lopetusehto löytyy siitä, että numeroita halutaan tulostaa 20:een asti. Eli ehdoksi saadaan i<=20. Jälleen i muuttujaa käytetään kierroslaskurina. Ohjelma näyttäisi tässä vaiheessa seuraavalta:

```
while (i<=20) {

}
```

Seuraavaksi kierroslaskuri on muuttujana määriteltävä ja annettava sille aloitusarvo, josta toisto

aloitetaan. Tässä toisto halutaan aloittaa arvosta 10, joten lause on var i = 10. Nyt ohjelmamme näyttää seuraavalta:

```
var i = 10;
while (i<=20) {

}
```

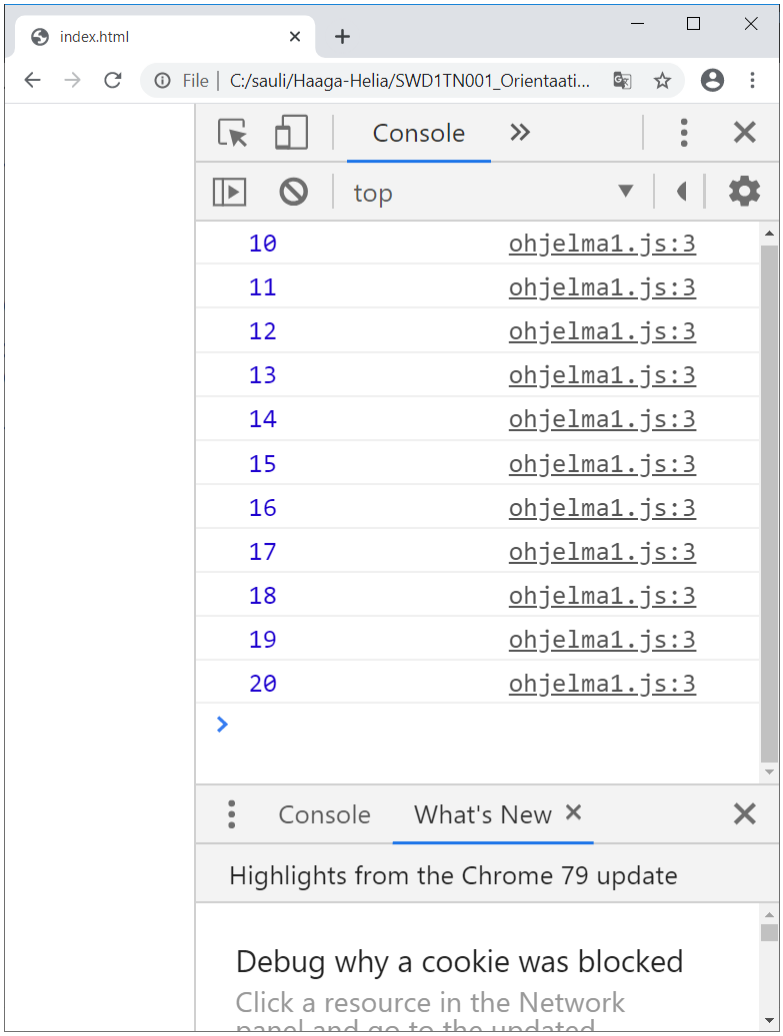
Kolmanneksi tarvitaan vielä kasvattaa kierroslaskurin arvoa, jokaisen kierroksen lopussa. Se onnistuu samaan tapaan kuin vastaavassa for esimerkissämme, eli i++. Tässä vaiheessa ohjelmamme on seuraavanlainen:

```
var i = 10;
while (i<=20) {

      i++;
}
```

Lopuksi voisimme näyttää kierroslaskurin arvon selaimen konsoli-ikkunassa console.log() metodin avulla:

```
var i = 10;
while (i<=20) {
      console.log(i);
      i++;
}
```

Kun ohjelma ajetaan selaimessa, löytyy tulos konsoli-ikkunasta:



b Materiaalia kiinnostuneille

- `for ... in`

  You select `for...in` loop, when you need to go through an array. The easiest way.
  For example:

```
for (i in playlist) {
    document.write("<br/>Now playing: " + playlist[i]);
}
```

For example:
(JavaScript: The Definitive Guide, s. 100)

```
var o = {x:1, y:2, z:3};
var a = [], i = 0;
for(a[i++] in o) /* empty */;

for (i in a) {
   document.write("<br/>a["+i+"]= " + a[i]);
}
```