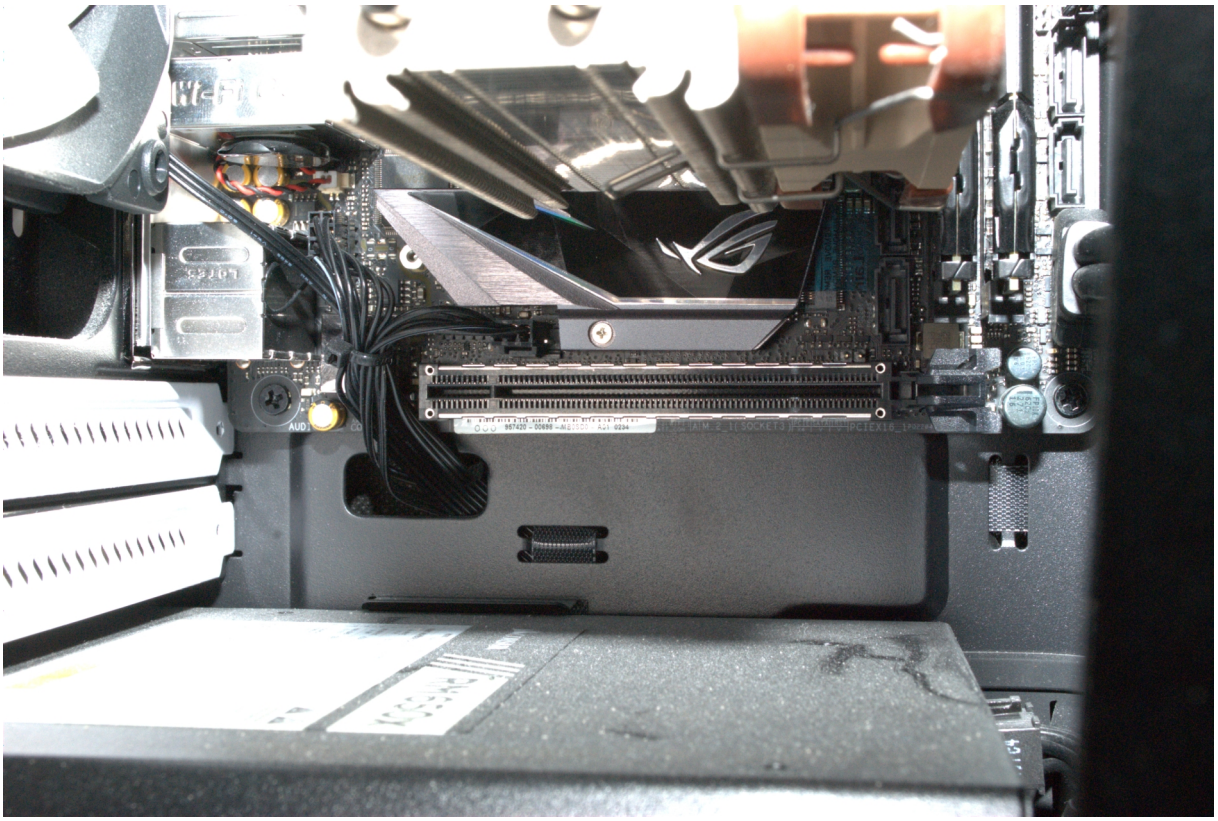
a) JavaScript muuttujat, operaattorit
JavaScript Variables, Types, Operators, and Expressions

Ohjelman ajon aikana tarvitaan tallentaa tietoa muistiin. Tällainen tilanne syntyy, jos käyttäjältä kysytään esimerkiksi nimeä ja osoitetietoja tai muuta sellaista. Tietoa voidaan tallentaa tietokoneessa keskusmuistiin (Random Access Memory, RAM) tai pitkä aikaiseen tallennukseen massamuistiin kiintolevylle tai SSD -levyyn (Solid-state drive). Tässä keskitymme keskusmuistiin tehtävään tallentamiseen. Tärkein ero keskusmuistiin tallentamisen ja massamuistin välillä on, että keskusmuistissa oleva tieto säilyy vain ohjelman ajon ajan. Tieto siis häviää sen jälkeen. Massamuistissa tieto sen sijaan säilyy pitkiä aikoja.



Keskusmuistiin voi tallentaa tietoa muuttujiin. Muuttuja täytyy ensin määritellä (luoda), ja se tapahtuu JavaScriptissä var tai uudemmalla let avainsanalla. Jos haluttaisiin vaikkapa tallentaa sukunimi, onnistuisi se lauseella:

```
var lastName;
```

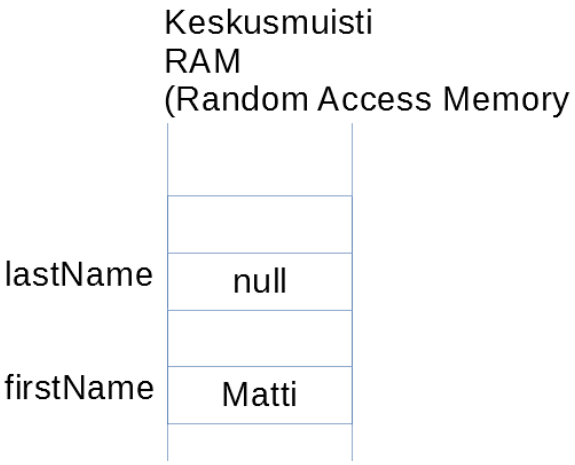
tai vaihtoehtoisesti uudemmalla tavalla:

```
let lastName;
```

Tässä määriteltiin muuttuja lastName, mutta sille ei annettu mitään arvoa. Sen vuoksi sen arvo on tällä hetkeälä null.

Seuraavassa lauseessa määritellään ajonaikaisesti tietokoneen keskusmuistiin muuttuja firstName, jonka arvo on annettu:

```
var firstName = "Matti";
```



Käytämme tässä Camel Case nimeämistapaa (https://en.wikipedia.org/wiki/CamelCase). Nimensä mukaisesti siinä on ”kamelin kyttyrä”, kuten nimessä firstName on. Tämä tapa helpottaa lukemista, koska muuttuja nimeä ei voi antaa esimerkiksi muodossa: var first name = "Matti"; Tämä aiheuttaisi syntaksi virheen, koska siinä on välilyönti first sanan ja name sanan välissä. Muuttujien nimissä on siis noudatettava JavaScriptin sääntöjä. Muuttuja nimessä ei saa olla erikoismerkkejä, eikä muuttuja saa alkaa numerolla, mutta sen jälkeen nimessä saa olla numeroita.

Jos muuttujaan haluaa tallentaa tekstiä, annetaan teksti lainausmerkkien sisällä, kuten yllä olevassa esimerkissä Matti annettiin. Lainausmerkkinä voi JavaScriptissä käyttää " –merkkiä tai ' –merkkiä. Numerot annetaan puolestaan alla olevaan tapaan:

```
var pisteet = 25;
var paino = 80.5;
```

Huomaa, että desimaalierotin JavaScriptissä on piste eikä pilkku.

JavaScript muuttujiin voi siis tallentaa tekstiä (merkkijonoja) ja numeroita (kokonaisnumeroita ja desimaalinumeroita). Muuttujan arvon voi peräti muuttaa ohjelman ajon aikana tekstistä numeroksi alla olevaan tapaan:

```
var id = "DX101";         // of type string
id = 1601;                // of type number
```

JavaScriptissä voit selvittää muuttujan sisältämän arvon tietotyypin avainsanalla `typeof`. Tämän tiedon voi tulostaa vaikkapa viesti-ikkunaan `alert` functiolla, joka kuuluu JavaScriptiin:
`alert(typeof id);`

Kaiken kaikkiaan JavaScriptissä ovat siis seuraavat tietotyypit: merkkijonot (String), numerot (Number), Boolean, taulukot (Arrays) ja oliot. Niistä on hyvä olla tietoinen, vaikka niitä ei tarvitsekaan murehtia uutta muuttujaa määritellessään eli luodessaan.

Seuraavaksi voisimme katsoa millaisen ohjelman voisi tehdä kahden numeron yhteenlaskemiseksi. Koska numeroita on kaksi, voisi olla hyvä idea esimerkiksi tehdä kaksi muuttujaa. Kolmanteen muuttujaan voitaisiin tallentaa yhteenlaskun tulos. Tulos voitaisiin lopuksi näyttää console –ikkunassa tai Web sivulla.

Esimerkki 1: Kahden numeron laskeminen yhteen JavaScriptissä (huomaa tuloksen tarkkuus).

```
var num1;

num1 = 3.14159;
var num2 = 3;
var sum = num1 + num2;

document.write(sum);
```

Ensimmäisessä lauseessa var avainsanan avulla otetaan käyttöön ajonajaksi muuttuja num1. Sille ei anneta mitään arvoa, joten sen arvo on undefined.

Kakkos rivillä num1 muuttujalle annetaan arvoksi 3.14159 (huomaa että desimaalierotin on piste, ei pilkku).

Kolmannella rivillä määritellään uusi muuttuja num2, jolle annetaan heti arvo.

Neljännellä rivillä luodaan var avainsanalla uusi muuttuja sum (huomaa, että let avainsanaakin voisi käyttää). Muuttujalle sum lasketaan arvo = operaattorin oikealla puolella. Muuttujien num1 ja num2 arvot lasketaan yhteen ja talletetaan sum muutujan arvoksi. sum muuttujan arvoksi tulee 6.14159. Tämä tulos näytetään lopuksi document.write() metodilla käyttäjän web –selaimessa.

Laskutoimituksia voi tehdä seuraavilla operaattoreilla:

+ Yhteenlasku ja merkkijonojen yhdistäminen yhteen.
  Esim. var kokoNimi = "Milla" + "Magia";
  tallentaa muuttujaan kokoNimi arvon MillaMagia
- Vähennyslasku
* Kertolasku
/ Jakolasku
% Jakojäännos
  Esim. var jakoj = 5%2;
  tallentaa jakoj muuttujaan arvon 1.

Seuraavassa esimerkissä tutustumme desimaalinumeroiden tarkkuuden pyöristämiseen haluttuun tarkkuuteen. Esimerkiksi jakolaskun yhteydessä usein vastauksena tulee numero, jonka desimaaliosa on pitkä. Valuuttojen yhteydessä esimerkiksi tämä halutaan pyöristää kahteen desimaaliin. Tämä onnistuu toFixed() funktiolla, joka löytyy JavaScriptissä kaikista muuttujista.

Esimerkki 2: `toFixed` function JavaScript numeroiden tarkkuuden formatoinnissa haluttuun desimaalitarkkuuteen.
http://www.w3schools.com/jsref/jsref_toprecision.asp
http://www.w3schools.com/jsref/jsref_obj_global.asp
(JavaScript: The Definitive Guide, s. 48)

```
var kustannus;
kustannus = 55;
var lkm = 3;
var maksu = kustannus / lkm;
var maksu_formated = maksu.toFixed(2);

console.log(maksu_formated);
```

Jakolaksun kustannus/lkm tuloksena saadaan 18.333333333. Jos tulos halutaan esittää pyöristettynä kahden desimaalintarkkuuteen, onnistuu se .toFixed() metodin avulla. Lauseessa
`var maksu_formated = maksu.toFixed(2);`
suoritetaan tämä pyöristys muuttujan maksu arvolle. Tulos talletetaan maksu_formated muuttujan arvoksi, jonka jälkeen se tulostetaan käyttäjän web selaimen konsoli-ikkunaan.

Saman voi tehdä myös toPrecision() funktiolla:

Esimerkki 3: Myös `toPrecision` funktion avulla voi muuttaa numeroiden tarkkuutta.
Tässä esimerkissä esitetään kaikki tiedostot, jotka tarvitaan ohjelman ajamiseksi web –selaimessa. Ensimmäiseksi tarvitaan html tiedosto, jonka sisältö on esitetty alla:

```
<!DOCTYPE html>
<html>
<head>
  <title>5. JavaScript Variables, Types, Operators and
Expressions</title>
  <script type="text/javascript" src="ohjelma1.js">
  </script>
</head>
<body>
  <h1>5. JavaScript Variables, Types, Operators and Expressions</h1>
  Anna numero 1:<input type="text" id="numero1Text"><br />
  Anna numero 2: <input type="text" id="numero2Text">
  <input type="submit" name="addButton" value="Add"
onclick="add()"></input>
</body>
</html>
```

HTML tiedostossa on <script> elementti, jossa on kerrottu src attribuutissa, mistä tiedostosta löytyy JavaScript tiedosto. Tämä tiedosto on nimeltään ohjelma1.js ja löytyy samasta kansiosta kuin html tiedostokin. Alla on esitetty JavaScript tiedoston sisältö:

```
function add() {
     var numero1;
     numero1 = document.getElementById('numero1Text').value;
     var numero2 = document.getElementById('numero2Text').value;
     var tulo = numero1 * numero2;
     var tulo_formated = tulo.toPrecision(3);
     document.write(sum_formated);
}
```

JavaScript ohjelmaa ajettaessa haetaan muuttujaan numero1 arvo html sivun tekstilaatikosta. Kyseisen tekstilaatikon id nimi on numero1Text. Tätä id nimeä voidaan käyttää tekstilaatikon etsimiseen html sivulta. Tämä onnistuu

```
document.getElementById('numero1Text').value
```

lauseella. Metodille getElementById() annetaan html elementin id nimi haettavaksi. Kun se on löytynyt html sivu DOM puusta, otetaan löydetyn elementin value arvo ja talletetaan se muuttujan numero1 arvoksi. Seuraavalla rivillä tehdään sama asia numero2 muuttujalle.

Tätä seuraavalla rivillä kerrotaan numero1 numero2:lla, jonka jälkeen tulos voidaan tallentaa tulo muuttujan arvoksi.

Lopuksi pyöritetään tulos kolmen desimaalin tarkkuuteen ja näytetään tulos käyttäjän web selaimessa. Tämä onnistuu document.write() metodin avulla.

Numeroiden laskemisen ohella voimme myös yhdistää tekstiä yhteen (concatenate). Esimerkiksi voisimme yhdistää sanat Matti ja Virtanen yhteen MattiVirtanen tekstiksi. Se onnistuu lauseella:

```
var nimi = "Matti" + "Virtanen";
```

Merkkijonoja (eli tekstiä) voi siis yhdistää yhteen samalla + operaattorilla kuin edellä suoritimme yhteenlaskua. Tämä ei ole ongelma, sillä 3+5 on selkeästi numeroita, joten tulos on 8. Jos mukana olisi tekstiä, esimerkiksi "kosini"+ 5, olisi tilanne eri. Nyt yksi arvoista on tekstiä (kosini), joten kaikki yhdistetään yhteen ja tulokseksi saadaan kosini5

Esimerkki 4: Merkkijonojen yhdistäminen (Concatenation of strings).

```
var pii;
pii = 3.14;
var arvo1 = "3";
var sum = pii + arvo1;

alert(sum);
```

Yllä olevassa ohjelmassa muuttujalle arvo1 annetaan arvoksi 3 mutta koska se annetaan " merkkien sisällä, on arvo tekstiä ei siis numero JavaScriptin kannalta.

Kun lauseessa `sum = pii + arvo1;` + operaattorilla lasketaan muuttujat, ei kyseessä olekaan enää yhteenlaskenta, vaan merkkijonojen yhdistäminen. + -operaattorilla on siis kaksi eri tapaa toimia JavaScriptissä. Jos kaikki arvot ovat numeroita + operaattori suorittaa yhteenlaskennan. Jos yksikin arvoista on tekstiä niin + operaattori yhdistää merkit yhteen tekstiksi (merkkijonoksi).

Esimerkki 5: Totuusarvo (Boolean eli true tai false)

```
var onkoTotta = true;
alert(typeof onkoTotta);
```

Muuttujalle voi antaa arvoksi myös totuusarvon eli boolean arvon. Boolean arvo voi olla vain true tai false.

Yllä olevassa esimerkissä otetaan käyttöön muuttuja onkoTotta, jonka arvoksi annetaan true.

Kakkos rivillä muuttujan onkoTotta sisältämän arvon tietotyyppi selvitetään typeof avainsanalla. Koska onkoTotta muuttuja sisältää arvon true, antaa typeof tulokseksi boolean. Tämä tulos näytetään sitten viesti-ikkunassa alert -funktion avulla. Funktio alert on JavaScriptin sisään rakennettuja funktioita, eli se löytyy aina JavaScriptistä.

Esimerkki 6: Undefined data value

```
var someValue;
alert(typeof someValue);
```

Yllä oleva esimerkki on muutoin sama kuin esimerkki 5, paitsi että muuttujalle onkoTotta ei anneta mitään arvoa eli se on undefined. Tämän vuoksi alert funktio tulosta nyt viesti-ikkunaan tekstin undefined.

Esimerkki 7: JavaScriptissä voi muuttuja arvon muuttaa peräti tekstistä numeroksi.

```
var someValue = "This is string";
someValue = 5;
alert(typeof someValue);
```

Esimerkki 8: Escape character:llä voi ns. eskeipata JavaScriptin merkkejä niin halutessaan

```
var someValue = "My so called \"life\"";
alert(someValue);
```

Ensimmäisen rivin \" eskeippaa eli välttää JavaScriptin " merkin käytön. Tässä nimittäin haluamme tallentaa someValue muuttujaan arvon My so called "life" eli haluamme " että lainausmerkit tulevat life sanan ympärille. Emme halua käyttää niitä JavaScriptin merkkijonojen antamiseen. Escape merkki \ siis pystyy välttämään tämän. Escape merkillä \ voi eskeipata mitä tahansa merkkejä, joilla on JavaScriptissä merkitys (esimerkiksi tällaisia ovat ' ja tietysti itse \ merkki myös).