

Front-End Development

Exercise 02

Functions and Object Literals

our course book "Eloquent JavaScript" (3rd edition) by Marjin Haverbeke. The aims of the exercise are to learn the basics of working with functions, arrays, and objects in JavaScript. Embed your theory answers, drawings, codes, and screenshots directly into this document. Always immediately after the relevant Help to complete the tasks of this exercise can be found on the chapter 3 "Functions" and chapter 4 "Data Structures: Objects and Arrays" of question. Return the document into your return box in itsLearning by the deadline. Remember to give your own assessment when returning this document. It's also recommendable to use Internet sources to supplement the information provided by the course book. The maximum number of points you can earn from this exercise is $10 + 1 = 11$.

Tasks:

1. Program a function by using a function declaration. (2 points)

Write the function `isLeapYear` according to ES5 standard. **Use a function declaration.**

The function takes a `year` to be checked as a parameter.

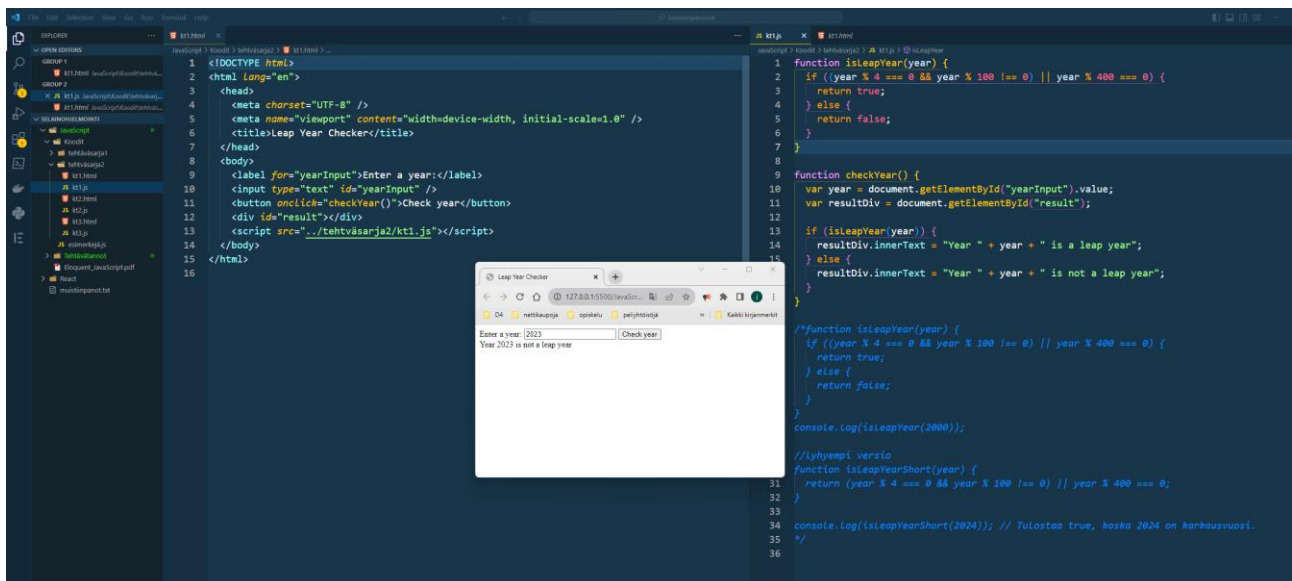
The function returns `true` if the given parameter value is a leap year and `false` if it is not a leap year.

It should be noted that leap years must meet the following two conditions at the same time: 1) when the year is divided by four, the remainder is zero, and 2) (when the year is divided by one hundred, the remainder is not zero) or (when the year is divided by four hundred, the remainder is zero).

Fill a year into a textbox on a web page. Call the function by clicking the button Check year and display (by utilizing another function) "Year xxxx is a leap year" or "Year xxxx is not a leap year" in a div below. The xxx is the filled in year.

WE WILL PROGRAM THIS TOGETHER.

Tehtävä1





2. Program a function by using a function expression. (1 point)

Implement the function `containsNumber` according to ES5 standard. However, **use this time a function expression**.

The function takes two arguments. The first argument is `numbers`, which is an array of numbers. The second argument is `aNumber`, which is the number to search.

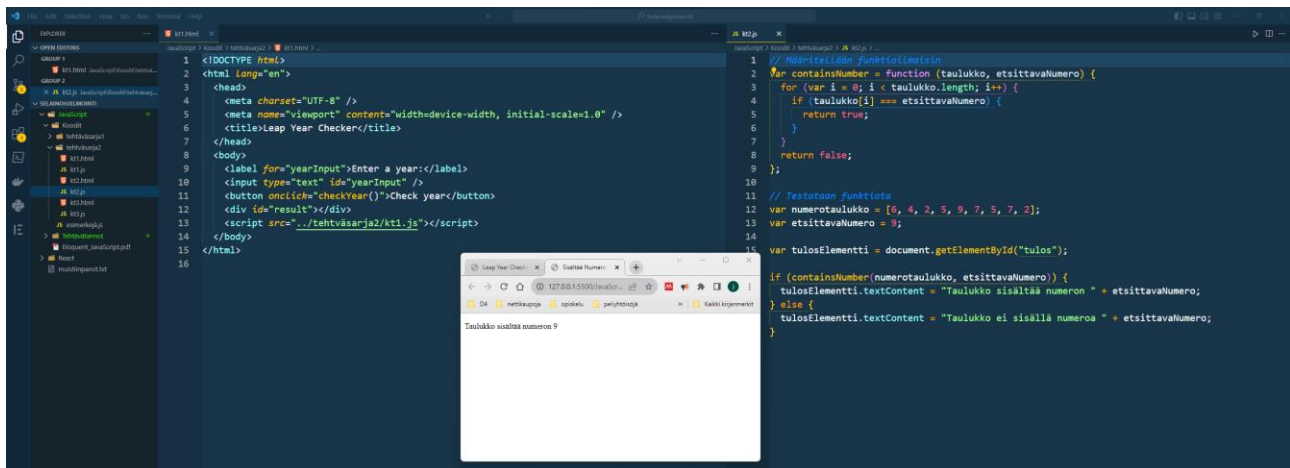
An example of one possible `elements` array:

`[6, 4, 2, 5, 9, 7, 5, 7, 2]`

The function returns `true`, if the array contains the number given in `aNumber`. Otherwise, the function returns `false`.

Display the result on the web page like in the task 1. However, this time the result text should be like “Array contains the number x” or “Array doesn’t contain the number x”.

Tehtävä2





3. Program a function by using an arrow function. (2 points)

Write the function `convertToMinutesFormat`. Use this time an arrow function introduced in ES6.

The function takes a `hoursInHundredths` as a parameter. The function should be able to handle a parameter value that is given in one of the following formats: `x.xx`, `xx.xx`, `x,xx` or `xx,xx`.

Example parameter values: 3.40, 03.20, 0.15, 14.80.

The function returns hours and minutes in one of the following formats: `h:mm` or `hh:mm`.

From the above example parameter values the function returns: `3:24`, `3:12`, `0:09`, `14:48`.

Fill an hour time to convert into a textbox on a web page. Call the function by clicking the button "Convert to Minutes" and display the result like (by utilizing another function) "3,20 hours are in hours and minutes equal to 3:12"

Take care of the necessary rounding of minutes.

Tehtävä 3

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8" />
5 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6 <title>Document</title>
7 <script src="../../tehtava3/kt3.js"></script>
8 </head>
9 <body>
10 <p id="result"></p>
11 <script>
12 document.addEventListener("DOMContentLoaded", function () {
13 document.getElementById("result").innerHTML =
14 convertToMinutesFormat("3.40");
15 });
16 </script>
17 </body>
18 </html>
19
1 const convertToMinutesFormat = (hoursInHundredths) => {
2 // Tarkista, onko syöte numero, ja jos ei ole, palauta virheviesti
3 if (isNaN(hoursInHundredths)) {
4 return "Invalid input";
5 }
6
7 // Muunna syöte desimaaliluvuksi
8 const hoursFloat = parseFloat(hoursInHundredths);
9
10 // Tarkista, onko syöte negatiivinen
11 const isNegative = hoursFloat < 0;
12
13 // Otetaan luku ilman desimaaleja
14 const hours = Math.floor(Math.abs(hoursFloat));
15
16 // Lasketaan desimaaliosan minuutit ja pyöristetään ne
17 const minutes = Math.round((Math.abs(hoursFloat) - hours) * 60);
18
19 // Muotoillaan tulos
20 const formattedHours = hours.toString().padStart(2, "0");
21 const formattedMinutes = minutes.toString().padStart(2, "0");
22
23 // Palautetaan tulos
24 if (isNegative) {
25 return `-${formattedHours}:${formattedMinutes}`;
26 } else {
27 return `${formattedHours}:${formattedMinutes}`;
28 }
29 };
30
31 // Testausta varten voit kutsua funktiota eri syötteillä
32 console.log(convertToMinutesFormat(3.4)); // Tulostaa "03:24"
33 console.log(convertToMinutesFormat(-3.4)); // Tulostaa "-03:24"
34 console.log(convertToMinutesFormat("invalid")); // Tulostaa "Invalid input"
35

```



4. Use Net to learn JavaScript API. (1 point)

Search 3 good JavaScript references from the Net. Answer the following questions.

a. What are the names and the url of the references you found? (0,5 points)

1. Mozilla Developer Network (MDN) - WebExtensions API
 - <https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API>
2. W3Schools - JavaScript Reference - Web APIs
 - https://www.w3schools.com/jsref/api_web.asp
3. GeeksforGeeks - Working with APIs in JavaScript
 - <https://www.geeksforgeeks.org/working-with-apis-in-javascript/>

b. What kind of information the references give to you? (0,5 points)

1. Mozilla Developer Network (MDN API):

Tämä viite tarjoaa kattavat tiedot Mozilla WebExtensions) - **WebExtensions** API:sta, joka liittyy selainlaajennusten kehittämiseen. Se sisältää yksityiskohtaiset selitykset API-metodeista ja ominaisuuksista.

2. W3Schools - JavaScript Reference - Web APIs:

W3Schools tarjoaa selkeät ja käyttäjäystävälliset esimerkit ja selitykset erilaisista JavaScriptin Web API -osista. Näitä ovat esimerkiksi DOM (Document Object Model) -API ja muita selainliittymiä koskevat tiedot.

3. GeeksforGeeks - Working with APIs in JavaScript:

GeeksforGeeks tarjoaa opetusmateriaalia JavaScriptin API:n käytöstä, mukaan lukien ohjeet ja esimerkit erilaisten API-palveluiden käyttämisestä JavaScriptissä.



5. Give short code examples. (1 point)

a. How do you give default values for the function parameters?

Funktioparametrien oletusarvot voidaan määrittää seuraavalla tavalla:

```
1 function tervehdi(nimi = "Matti Meikäläinen") {
2   console.log("hei," + nimi + "!");
3 }
4
5 tervehdi();
6 tervehdi("Pekka Puupää");
7
8 /*var taulukko1 = [1, 2];
9 var taulukko2 = [3, 4];
10 var yhdistetty = taulukko1.concat(taulukko2); // Luo uuden taulukon 'yhdistetty' ilman että 'ta
11 console.log(yhdistetty); // Tuloste: [1, 2, 3, 4]
12 console.log(taulukko1); // Tuloste: [1, 2] (alkuperäinen 'taulukko1' säilyy muuttumattomana)
13 console.log(taulukko2); // Tuloste: [3, 4] (alkuperäinen 'taulukko2' säilyy muuttumattomana)
14
15 /*
16 // Luodaan taulukko 'autot' ja alustetaan se kahdella merkkijonolla
17 var autot = ["opel", "mersu"];
18 // Käytetään 'push'-metodia lisäämään "mazda" taulukon loppuun
19 autot.push("mazda");
20 // Tulostetaan muokattu 'autot'-taulukko
```

PORTS PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS C:\Users\jouni\OneDrive\Desktop\Selainohjelointi> node "c:\Users\jouni\OneDrive\Desktop\Selainohjelointi\JavaScript\Koodit\esimerkeja.js"
hei,Matti Meikäläinen!
hei,Pekka Puupää!
```



b. How do you use rest parameters?

Rest-parametrit mahdollistavat muuttuvan määrän argumentteja funktiolle. Ne keräävät ylimääräiset argumentit taulukkoon. Tässä on esimerkki:

```
1 function summa(...numerot) {
2   let tulos = 0;
3   for (let numero of numerot) {
4     tulos += numero;
5   }
6   return tulos;
7 }
8
9 console.log(summa(9, 8, 7, 6));
10 console.log(summa(51, 10));
11 console.log(summa());
12
13 /*function tervehdi(nimi = "Matti Meikäläinen") {
14   console.log("hei," + nimi + "!");
15 }
16 tervehdi();
17 tervehdi("Pekka Puupää");
18
19 var taulukko1 = [1, 2];
20 var taulukko2 = [3, 4];
```

PORTS PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS C:\Users\jouni\OneDrive\Desktop\Selainohjelointi> node "c:\Users\jouni\OneDrive\Desktop\Selainohjelointi\JavaScript\Koodit\esimerkeja.js"
30
61
0
PS C:\Users\jouni\OneDrive\Desktop\Selainohjelointi>
```



6. Basics of object literals. (2 point)

This time we concentrate on object literals. Write code clips to

- Create an object book containing the following properties: isbn, name, authors, publicationDate. (0,5 points)

```

1 var book = {
2   isbn: "548-0123456789",
3   name: "Tämä on kirja",
4   authors: ["matti meikäläinen", "Maisa meikäläinen"],
5   publicationDate: "2023-09-30",
6 };
7 console.log(book.name);
8 console.log(book.authors[0]);
9

```

Terminal output:

```

PS C:\Users\jouni\OneDrive\Desktop\Selainohjelointi> node "c:\Users\jouni\OneDrive\Desktop\Selainohjelointi\JavaScript\Koodit\tehtvasarja2\kt6.js"
Tämä on kirja
matti meikäläinen
PS C:\Users\jouni\OneDrive\Desktop\Selainohjelointi>

```

- Add the following methods with the following names to the book object: getAuthors, setAuthors, getIsbn, setIsbn. (0,5 points)

```

1 var book = {
2   isbn: "548-0123456789",
3   name: "Tämä on kirja",
4   authors: ["Matti Meikäläinen", "Maisa Meikäläinen"],
5   publicationDate: "2023-09-30",
6
7   // Metodi saataksesi kirjailijat
8   getAuthors: function () {
9     return this.authors;
10  },
11
12  // Metodi asettaaksesi kirjailijat
13  setAuthors: function (newAuthors) {
14    this.authors = newAuthors;
15  },
16
17  // Metodi saataksesi ISBN
18  getIsbn: function () {
19    return this.isbn;
20  },
21
22  // Metodi asettaaksesi ISBN
23  setIsbn: function (newIsbn) {
24    this.isbn = newIsbn;
25  },
26 };
27

```

Terminal output:

```

PS C:\Users\jouni\OneDrive\Desktop\Selainohjelointi> node "c:\Users\jouni\OneDrive\Desktop\Selainohjelointi\JavaScript\Koodit\tehtvasarja2\kt6.js"
Tämä on kirja
Matti Meikäläinen
PS C:\Users\jouni\OneDrive\Desktop\Selainohjelointi>

```



- c. Create two book objects. Compare if they model the same book. You can use the value of the field isbn as comparison criteria in defining equality: Same isbn value, same book. (0,5 points)

```

1  var book1 = {
2      isbn: "548-0123456789",
3      name: "Kirja",
4      authors: ["Maisa Meikäläinen"],
5      publicationDate: "2023-09-30",
6  };
7
8  var book2 = {
9      isbn: "858-9876543210",
10     name: "Kirja 2",
11     authors: ["Matti Meikäläinen"],
12     publicationDate: "2023-10-1",
13 };
14
15 if (book1.isbn === book2.isbn) {
16     console.log("Samat kirjan kuvaajat");
17 } else {
18     console.log("Eri kirjat kuvaajat");
19 }
20
21 /*
22 var book = {
23     isbn: "548-0123456789",

```

PORTS PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

PS C:\Users\jouni\OneDrive\Desktop\Selainohjelmointi> node "c:\Users\jouni\OneDrive\Desktop\Selainohjelmointi\JavaScript\Koodit\tehtävsarja2\kt6.js"
Eri kirjat kuvaajat
PS C:\Users\jouni\OneDrive\Desktop\Selainohjelmointi>

```

- d. Create two book objects with exactly the same values in all the features. Do they have the same identity? (0,5 points)

```

3  name: "Tämä on kirja",
4  authors: ["Matti Meikäläinen", "Maisa Meikäläinen"],
5  publicationDate: "2023-09-30",
6  };
7
8  var book2 = {
9      isbn: "548-0123456789",
10     name: "Tämä on kirja",
11     authors: ["Matti Meikäläinen", "Maisa Meikäläinen"],
12     publicationDate: "2023-09-30",
13 };
14
15 if (book1 === book2) {
16     console.log("Nämä kaksi kirjaoliota ovat identtisiä");
17 } else {
18     console.log("Nämä kaksi kirjaoliota eivät ole identtisiä");
19 }
20
21 /*
22 var book1 = {
23     isbn: "548-0123456789",

```

PORTS PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

PS C:\Users\jouni\OneDrive\Desktop\Selainohjelmointi> node "c:\Users\jouni\OneDrive\Desktop\Selainohjelmointi\JavaScript\Koodit\tehtävsarja2\kt6.js"
Nämä kaksi kirjaoliota eivät ole identtisiä
PS C:\Users\jouni\OneDrive\Desktop\Selainohjelmointi>

```




7. Working with objects. (2 points)

Write the function `convertOuncesToGrams`. You can use ES6 standard features.

The function takes `measurements` as a parameter.

The `measurements` parameter is an array containing objects.

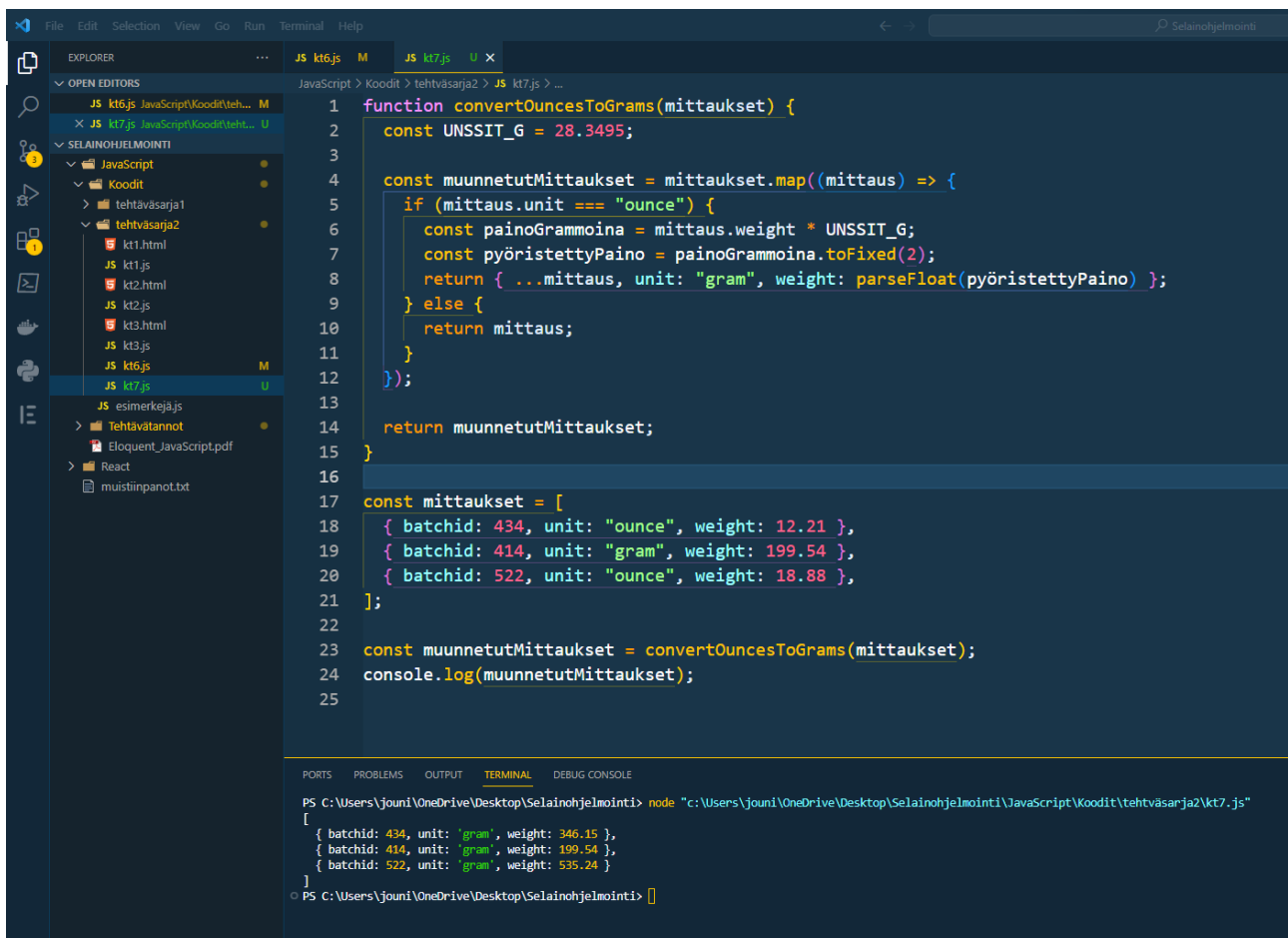
An example of the value of the `measurements` parameter:

```
[{ batchid: 434, unit: "ounce", weight: 12.21 }, {batchid: 414, unit: "gram", weight: 199.54 }, { batchid: 522, unit: "ounce", weight: 18.88 }]
```

The function returns an array where all the measurements are in grams. Like the following:

```
[{ batchid: 434, unit: "gram", weight: 346.15 }, {batchid: 414, unit: "gram", weight: 199.54 }, { batchid: 522, unit: "gram", weight: 535.24 }]
```

Please, give the results with two digits.



```
1 function convertOuncesToGrams(mittaukset) {
2   const UNSSIT_G = 28.3495;
3
4   const muunnetutMittaukset = mittaukset.map((mittaus) => {
5     if (mittaus.unit === "ounce") {
6       const painoGrammoina = mittaus.weight * UNSSIT_G;
7       const pyöristettyPaino = painoGrammoina.toFixed(2);
8       return { ...mittaus, unit: "gram", weight: parseFloat(pyöristettyPaino) };
9     } else {
10      return mittaus;
11    }
12  });
13
14  return muunnetutMittaukset;
15 }
16
17 const mittaukset = [
18   { batchid: 434, unit: "ounce", weight: 12.21 },
19   { batchid: 414, unit: "gram", weight: 199.54 },
20   { batchid: 522, unit: "ounce", weight: 18.88 },
21 ];
22
23 const muunnetutMittaukset = convertOuncesToGrams(mittaukset);
24 console.log(muunnetutMittaukset);
25
```

```
PS C:\Users\jouni\OneDrive\Desktop\Selainohjelointi> node "c:\Users\jouni\OneDrive\Desktop\Selainohjelointi\JavaScript\Koodit\tehtvasarja2\kt7.js"
[
  { batchid: 434, unit: 'gram', weight: 346.15 },
  { batchid: 414, unit: 'gram', weight: 199.54 },
  { batchid: 522, unit: 'gram', weight: 535.24 }
]
```