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 our course book “Eloquent JavaScript” (3rd edition) by Marijin 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 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:

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Front-End Development
Exercise 02
Functions and Object Literals

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.

```
function isLeapYear(aYear) {

    if (aYear % 4 === 0 && (aYear % 100 !== 0 || aYear % 400 === 0)) {
        return true;
    }

    return false;

}

function displayResult() {



    let exampleYear = document.getElementById("aYear").value;



    document.getElementById("this").innerHTML =
    isLeapYear(exampleYear) ? `Year ${exampleYear}
    is a leap year`: `Year ${exampleYear} is not a leap year`;



}
```

```
<!doctype html>

<html lang="fi">


<head>

  <title>Javascript 1</title>

  <meta charset="utf-8" />

  <script src="Lesson 2/EX02T1.js"></script>

</head>



<body>
  Enter year: <input type="String" id="aYear" size="2" name="year" maxlength="4"

  <input type="button" id="button" onclick="displayResult()"
  value="Check Leap Year">
  <p id="this"></p>

</body>



</html>
```

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 functions 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 as in JS-FUNCTION-P-LEAPYEAR-1. This time the result text should be like “Array contains the number x” or “Array doesn’t contain the number x”.

```
const containsNumber = function(numbers, aNumber) {

    for (let i = 0; i < numbers.length; i++){
        if(numbers[i]== aNumber) {
            return true;
        }
    }
    return false;
};




function displayResult() {

    debugger;

    let someNumbers = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16];
    let aNumberToSearch = document.getElementById("aNumber").value;



    document.getElementById("this").innerHTML =
    containsNumber(someNumbers, aNumberToSearch) ?
     `Array contains the number ${aNumberToSearch} `: `Array doesn't contain the number `+ aNumberToSearch;
```

```
<!doctype html>

<html lang="fi">


<head>

  <title>Javascript 1</title>

  <meta charset="utf-8" />

  <script src="Lesson 2/EX02T2.js"></script>

</head>



<body>
  Enter number to search: <input type="number" id="aNumber" size="2" name="number" maxlength="4"><br>

  <input type="button" id="button" onclick="displayResult()"
  value="Check Number">
  <p id="this"></p>
</body>



</html>
```

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 able to handle a parameter value that is given in one of the following formats: x.xx, xx.xx, x,xx or xx,xx.

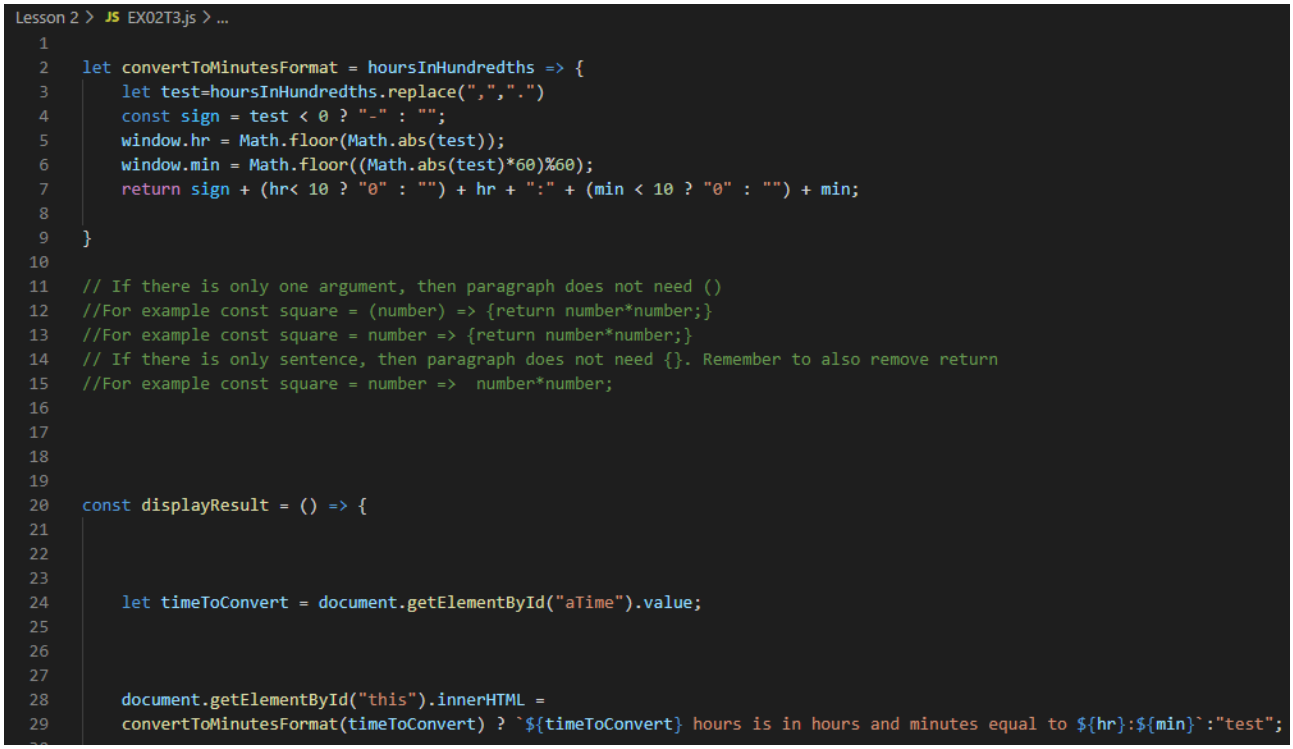
Example parameter values: 3,40, 03.20, 0.15, 14.80.

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 is in hours and minutes equal to 3:12"

Take care of the necessary rounding of minutes.



```
const containsNumber = function(numbers, aNumber) {

    for (let i = 0; i < numbers.length; i++){
        if(numbers[i]== aNumber) {
            return true;
        }
    }
    return false;
};




function displayResult() {

    debugger;

    let someNumbers = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16];
    let aNumberToSearch = document.getElementById("aNumber").value;



    document.getElementById("this").innerHTML =
    containsNumber(someNumbers, aNumberToSearch) ? `Array contains the number ${aNumberToSearch} `: `Array doesn't contain the number `+ aNumberToSearch;
```

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)

   MDN Web Docs.

   https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API

   W3 School is always good source for simple understanding.

   https://www.w3schools.com/js/js_api_intro.asp

   MDN Web Docs

   https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

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

   List of JavaScript API's and information for every API. I also get information what are APIs and how to use them.

5. Give short code examples. (1 point)

```
sayHello = name => console.log('Hello', name);

function sayHello(name) {
    console.log('Hello', name);
  }

sayHello("Steve")

volume=(l,w=3,h=4) => (l*w*h);

function volume(l,w,h){
    if (w=== undefined))
    w=3;
    if (h===undefined)
    h=4;
    return l*w*h;
}

volume(2)
```

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

Example in volume, I can give them straight to parameters, or I can also manually give them with volume(2,3,4)

b. How do you use rest parameters?

```
volume(2)

function sum(...theArgs) {
    return theArgs.reduce((previous, current) => {
      return previous * current;
    });
  }

console.log(sum(2,3,4))
```

Here I used “sum” but with volume example, I can give unlimited of parameters

6. Basics of object literals. (2 point)

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

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

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

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) Yes, they model same book, since it is made from same object aka. same blueprint.

d. Create two book objects with exactly the same values in all the features. Do they have the same identity? (0,5 points) Even if they have exactly same values, they are identified as different objects.

```
const book={
  isbn:"978-0-13-601970-1",
  name:"Book of Nico",
  authors:"Nico M. Kranni",
  publicationDate:"8.2.2022",
```

```
getAuthors : function() {
  return this.authors;
},
setAuthors : function() {
  book.authors = "Steve"
},
getIsbn:function(){
  return this.isbn;
},
setIsbn:function(){
  book.isbn="978-0-13-601970-9"
},
```

```
const book={
  isbn:"978-0-13-601970-2",
  name:"Book of Nico",
  authors:"Nico M. Kranni",
  publicationDate:"8.2.2022",
  getAuthors : function() {
    return this.authors;
  },
  setAuthors : function() {
    book.authors = "Steve"
  },
  getIsbn:function(){
    return this.isbn;
  },
  setIsbn:function(){
    book.isbn="978-0-13-601970-9"
  },
}

const book1=Object.create(book);
const book2=Object.create(book);
console.log(book1.name)

console.log(book1 == book2);
console.log(book1 === book2);
```

7. Handling 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.

```
function convertOuncesToGrams(measurements) {
  var arrayLength= measurements.length
  for (var i = 0; i < arrayLength; i++){ //Goes trough the array
    console.log(measurements[i].unit) // Trouble shoot prints
    if (measurements[i].unit=="ounce"){ // If the unit is "ounce" it runs these convert systems
      measurements[i].weight=measurements[i].weight*28.3495231; // Ounce to Gram ratio
      measurements[i].weight=Math.round((measurements[i].weight+ Number.EPSILON)*100)/100 // Rounding to two digits
      measurements[i].unit="grams"; // Changes unit after convert
    }



  }


}

const batch1={
  batchid:434,
  unit:"ounce",
  weight:12.21,
}

const batch2={
  batchid:414,
  unit:"gram",
  weight:199.54,
}

const batch3={
  batchid:522,
  unit:"ounce",
  weight:18.88,
}

const batches=[batch1, batch2, batch3] // We add objects to array
console.log(batches.length) // My troubleshoot lines

convertOuncesToGrams(batches)

console.log(batches)
```

```
▼ (3) [{…}, {…}, {…}]
  ▶ 0: {batchid: 434, unit: 'grams', weight: 346.15}
  ▶ 1: {batchid: 414, unit: 'gram', weight: 199.54}
  ▶ 2: {batchid: 522, unit: 'grams', weight: 535.24}
    length: 3
```

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Front-End Development
Exercise 02
Functions and Object Literals