Help to complete the tasks of this exercise can be found on the chapter 4 “Data Structures: Objects and Arrays” of our course book “Eloquent JavaScript” (3rd edition) by Marijin Haverbeke, but in this exercise you’ll need use Google too.

The aims of the exercise are to learn some skills working with objects and classes in JavaScript. However, the exercise does not elaborate on the rather intricate JavaScript prototypal inheritance. That is left for you as a subject of further studies.

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 + 3 = 13.

**Tasks:**

**1. Explain. (4 \* 0,25 = 1 point)**

a. What is the difference between having two references to the same object and having two different objects that contain the same properties?

Kun sinulla on kaksi viittausta samaan objektiin, molemmat viittaukset jakavat saman objektin muistissa, ja muutokset vaikuttavat kumpaankin viittaukseen. Kun sinulla on kaksi erillistä objektia, jotka sisältävät samat ominaisuudet, ne ovat itsenäisiä, ja muutokset toisessa eivät vaikuta toiseen.

b. The keyword new.

Kun käytät new-avainsanaa JavaScriptissä, se luo uuden instanssin tietylle objektiluokalle. Se varaa muistista tilan uudelle objektille ja viittaa siihen. Käyttäessäsi new-avainsanaa, luot uuden objektin tietyn rakenteen pohjalta, kuten konstruktorifunktion tai luokan määrittelyn perusteella. Tämä mahdollistaa useiden samanlaisten objektien luomisen, joilla on yhteisiä ominaisuuksia ja metodeja, mutta jotka voivat sisältää erilaisia arvoja. new-avainsana auttaa organisoimaan ja luomaan dynaamisia objekteja ohjelmoinnissa.

c. The keywod this.

**this** on avainsana JavaScriptissä, jota käytetään viittaamaan objektiin, jossa tietty koodi suoritetaan. **this** voi viitata eri objekteihin riippuen siitä, missä se sijaitsee koodissa ja miten funktio tai metodi on kutsuttu.

d. Is there something wrong in the examples below? Are you certain?

const friedmanBooks = [

  'The Little Schemer',

  'The Seasoned Schemer',

];

const hoyteBook = {

  name: 'Let Over Lambda',

  author: 'Doug Hoyte',

};

**2. Object destructuring. (4 \* 0,25 = 1 point)**

You have the following object.

const musician = {

name: 'Sting',

realName: 'Gordon Matthew Thomas Sumner’,

instrument: {

type: 'bass'

}

};

Use object destructuring to do the following assignments.

a. Read the attributes name and instrument into the variables name and instrument.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Multimediaohjelmisto

Kuvaus luotu automaattisesti

b. Read the attributes name and instrument into the variables nameOfArtist and instumentOfArtist.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Multimediaohjelmisto

Kuvaus luotu automaattisesti

c. Read the type of the instrument into a variable instrumentTypeOfArtist.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Multimediaohjelmisto

Kuvaus luotu automaattisesti

d. Read the make of the instrument into a variable instrumentMakeOfArtist. If the attribute is missing from the current object, give it a default value “unknown”.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Multimediaohjelmisto

Kuvaus luotu automaattisesti

**3. For..in loop, Object.entries, Object.keys, and Object.values. (4 \* 0,5 = 2 points)**

a. Explain for..in loop.

silmukka on JavaScript-ohjelmointikielen silmukkatyyppi, joka käy läpi objektin ominaisuudet, kuten avaimet tai indeksit, ja suorittaa sille määritellyn koodin kullekin ominaisuudelle. Tämä silmukka on hyödyllinen objektien tietojen käsittelyssä ja läpikäynnissä.

b. Explain static methods Object.entries, Object.keys, and Object.values.

JavaScriptissä **Object.entries**, **Object.keys** ja **Object.values** ovat staattisia metodeja, jotka liittyvät JavaScript-objekteihin. Nämä metodit mahdollistavat objektin ominaisuuksien, kuten avainten ja arvojen, käsittelyn ja käsittelytoimenpiteiden suorittamisen helposti ja tehokkaasti. **Object.entries** palauttaa objektin avaimet ja niihin liittyvät arvot, **Object.keys** palauttaa objektin avaimet ja **Object.values** palauttaa objektin arvot. Näitä metodeja voidaan käyttää monipuolisesti objektien sisällön tutkimiseen ja manipulointiin JavaScript-ohjelmoinnissa.Lomakkeen yläreuna

c. Use for..in with the object musician above. Log the attribute names and attribute values on the console. For example, when it is the turn of the attribute realName, the following text should be printed:

realName = Gordon Matthew Thomas Sumner

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Multimediaohjelmisto

Kuvaus luotu automaattisesti

d. Use the musician object from the task 2 again. Implement also another musician object. Put them both into an array. Give examples of the use of the static methods Object.entries, Object.keys, and Object.values with the object array you just created.

**Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Käyttöjärjestelmä

Kuvaus luotu automaattisesti**

**4. Getters and setters. (2 subtask answered gives 0,5 points, 3 subtasks answered gives 1 point)**

a. Create an object song. It has one attribute called name. It has a getter (a virtual attribute) called duration, and a setter that is also called duration. The getter returns the duration of the in minutes and seconds, and the setter can be used to set it.

b. Invoke the setter and the getter.

c. Explain the differences between normal object methods and these getters and setters.

**5. Working with JSON. (4 \* 0,5 = 2 points)**

a. What are the purposes of JSON?

b. There are few differences between JavaScript objects and JSON. List and explain them.

c. Serialize the object person below to a string containing a JSON object literal.

let person = {name: "Pentti", age: 22, country: "Finland"};

d. Deserialize the JSON object literal back to another JavaScript object.

**6. Working with some common JavaScript library objects. (2 \* 0,5 = 1 point)**

a. Create a function getRandomIntegerFromRange. It accepts two arguments. The argument startRange should be an integer and it sets the start of the Range. The argument endRange should also be an integer and it sets the end of the Range. The function returns a random integer that is greater or equal to the startRange abd less or equal to the endRange.

b. Create a function getTimeDifferenceInFullDays that returns the number of full days between to dates. It accepts two arguments. The argument startDate is the start date of the period. The function endDate is the end date of the period. Use Date and Math objects.

**7. Initializing an object with a JavaScript class. (2 \* 0,5 = 1 point)**

a. Create a class called Person. The class has a constructor that accepts to arguments: name and age. There should be two functions tacked on the class: getName, getAge and sayGreeting. Create 2 objects of the class. Call some methods.

b. What is the idea of a constructor?

**8. Creating a utility with a JavaScript class. (2 \* 0,5 = 1 point)**

a. Create a class ZipValidator. It has two static methods: isValidZip and fixZip. The first static methods accept a zipCode and checks that it contains only numbers, and that it contains exactly five numbers. It returns true or false. The second static method accepts an argument zipCode. If the argument has a length less than five characters, the method prefixes it with leading zeros. The method returns a valid zipCode. Use the class and call the static methods.

b. What is the difference between static methods and normal (instance) methods?

**9. Extending JavaScript classes. (2 subtask answered gives 0,5 points, 3 subtasks answered gives 1 point)**

a. Create a class SuperHero. Inherit it from the class Person. The constructor accepts an additional argument: superpower. There is also a function tacked on the class: useSuperPower. (In our case it is enough to just log the superpower on the console as a string)

b. How do you make certain that also the initializations defined in the constructor of the inherited class Person are done?

c. Create 2 objects of the class. Call some methods.

**10. Revealing Module pattern and IEFE. (4 \* 0,5 = 2 points)**

const greeter = (function () {

  let greeting = 'Hello';

  const exclaim = msg => `${msg}!`;

  const greet = name => exclaim(`${greeting} ${name}`);

  const salutation = (newGreeting) => {

    greeting = newGreeting;

  };

  return {

    greet: greet,

    salutation: salutation,

  };

}());

Look at the code above.

a. What is the idea of the code? What is the extra value it produces to JavaScript?

b. What is IEFE?

c. Use object greeter. Call its functions. Try to read and set the greeting attribute without calling a method. What do you notice?

d. Look at the object the function returns. Use property value shorthands to make it a bit less verbose. Do you lose anything when using the shorthands?