Help to complete the tasks of this exercise can be found on the chapter 10 ”Modules” of our course book “Eloquent JavaScript” (3rd edition) by Marijin Haverbeke and from the Net tutorials Promise (<https://javascript.info/promise-basics>), Promises Chaining (<https://javascript.info/promise-chaining>) and Error Handling with Promises (<https://javascript.info/promise-error-handling>). Please note that the Promise tutorial has even more parts, but we don’t elaborate on those is this exercise. The aims of the exercise are to learn to work with modules and to write simple asynchronous programs with promises.

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 platform by the deadline.

It’s also recommendable to use Internet sources to supplement the information provided by the course book.

Please note that modules cannot be tested in browser environment without a web server i.e. reading files directly to browser from the file system of your computer is not possible. The easiest way to test modules is to install node and execute them on the node platform.

The maximum number of points you can earn from this exercise is 10.

**Tasks:**

**1. JavaScript modules (2 \* 0.5 = 1 point)**

a. What is a module in JavaScript?

It is a file containing specific lines of code to do said specific thing. square.mjs would for example be used to create square (.mjs stands of module javascript, helps to know which has module). Modules should be independent “packages”, that way you are able to call/import them without too much of a hassle.

Modules helps you to be organized, tools in correct places rather than messy around in toolbox.

b. What kinds of benefits can modules provide? List at least 4.

Cleaner code, shorter files, reusable, name spacing, easier to manage

**2. Implementing an ES6 module. (2 points)**

THIS WILL BE DONE PARTLY TOGETHER DURING THE LESSONS

Create an ES6 module PriceList. The PriceList module publishes separately two operations: setPriceList and getPrice. Make certain that the module also publishes an object than can be used to reveal all public operations in one go.

Please be careful in naming the module file. The file name is going to be the module name.

The setPriceList operation accepts one argument: thePriceList. You can see the form of the argument in the example below.

[

{

itemid: 1,

itemname: ‘bottle’,

prices: [{ qty: 1, price: 1.25},{ qty: 100, price: 1.10},{ qty: 1000, price: 1.00}]

},

{

itemid: 2,

itemname: ‘glass’,

prices: [{ qty: 1, price: 2.50},{ qty: 50, price: 2.00},{ qty: 200, price: 1.60}]

}

]

The setPriceList operation assigns the given pricelist array into the module’s private variable priceList. Please note that the qty in the pricelist is the minimum quantity that must be bought to get the item with a related unit price.

You can use the pricelist shown above, but please add at least one more item and its prices to it.

The getPrice operation returns the total price of the quantity of the item. It accepts two arguments: itemId and quantity. For example, by using the pricelist shown above the function call getPrice(2, 150) returns 300.00.

**3. Using an ES6 module. (1 point)**

THIS WILL BE DONE PARTLY TOGETHER DURING THE LESSONS

Import the ES6 module of the previous task to another JavaScript module, set the pricelist, and call the getPrice operation several times with different products and quantities. Display the results to the console. Follow the instruction below and repeat the process twice. Please note that only modules can import modules.

1. Import the selected public features from the module one by one by listing their names. (0,25 points)



1. Import all the public features of the module at one go by utilizing the module’s public object. (0,25 points)



c. What are the pros and cons of the previous alternatives? (0,5 points)

By listing single features, you have better control. Important in languages that you compile.

By importing all the public you get all you need at once, useful when you need all functions from method.

**4. Implementing and using CommonJS modules. (2 points)**

Repeat the tasks 2 and 3 above (1 point each), but this time implement the application with CommonJS module.

**5. Understanding Promise objects and asynchronous programming. (4 \* 0,5 = 2 points)**

Answer shortly for the following questions:

1. What is the idea of using promise objects in JavaScript?

If I understood correctly, it is place holder for value that is possibly not known currently. It promises to either return value or reject value(fail).

1. How does the promise object move to state fulfilled?

If it executes without error. If error happens then state will be set to rejected. Fulfilled state gives a result value, as result. “Promise is kept or resolved”

1. How do you chain handler functions to a promise object so that they are run one after another?

You put multiple .then functions, then it goes from one .then to another.

1. How do you attach a handler function that is performed in a case of a rejected promise?

You give .then(resolved, rejected) argument and pass it to function rejected

**6. Fetching data from API asynchronously. (3 points)**

THIS WILL BE DONE PARTLY TOGETHER DURING THE LESSONS

The JavaScript’s fetch operation can be used to make asynchronous http requests to network resources. It returns a promise which results a response, which in turn has the json method which too returns a promise.

Implement an operation getComments that asynchronously gets comments that are in json form from the url <https://my-json-server.typicode.com/typicode/demo/comments>. Utilize the fetch operation.

1. Implement the operation by using promises. (1 point)
2. Implement the operation by using keywords async and await. (1 point)
3. Also handle the possible error situations in the cases above. (1 point)