**Scope & Context**

**DEADLINE:** 19/02/2019

## FOLDER STRUCTURE

|  |  |
| --- | --- |
| FLX\_homework\_10/\*  └─ task/  └─ FLX\_homework\_10.docx  └─ homework/\*  └─ src/\*  └─ credits\_handler.js\* | \*   – required |

|  |  |
| --- | --- |
|  |  |

## TASK

**1)** **Create a** **User Card:**

You should create a function **‘userCard’** (ES6 class is restricted for current task) which takes a number (some index) and returns interface with card methods:

const card3 = userCard(3); // returns an object with methods

**User Card methods:**

**getCardOptions.** This function returns object which contain information about card:

card3.getCardOptions(); // returns *options object* with card info

*Options object* should contain the following properties:

* **balance** (default value should be 100)
* **transactionLimit** (default value should be 100. It is a limit of credits you can take from this card)
* **historyLogs** (array of objects which contains information about card`s operations and transactions)
* **key** (number in range [1; 3] depends on index you passed to userCard function. Each card should have unique key)

**putCredits.** This function receives amount of credits and fills up balance of card:

card3.putCredits(150);

**takeCredits.** This function receives amount of credits and reduces these credits from balance of card (opposite Put credits method):

card3.takeCredits(100);

You can take credits from the card if transaction limit and remaining balance are greater than credits you want to take. If not, you should log appropriate message in the console (use **console.error**).

**setTransactionLimit.** This function receives amount of credits and set them as transaction limit of the card.

card3.setTransactionLimit(5000);

**transferCredits.** This function receives two arguments – amount of credits you want to transfer and recipient card (another object created by `userCard` function – card1):

card3.transferCredits(50, card1);

Credits you want to transfer should be taxed with 0.5% taxes.

Do not forget to check remaining balance and transaction limit of credits sender (card3) before transfer.

You should track **only** **Put credits/Take credits/Transaction limit change** operations and keep information about them in history log. History log (See Figure 1) should contain objects with properties:

* **operationType** (string which describes perfomed operation)
* **credits** (amount of credits involved in operation)
* **operationTime** (e.g. "27/07/2018, 03:24:53". Time when operation was performed)

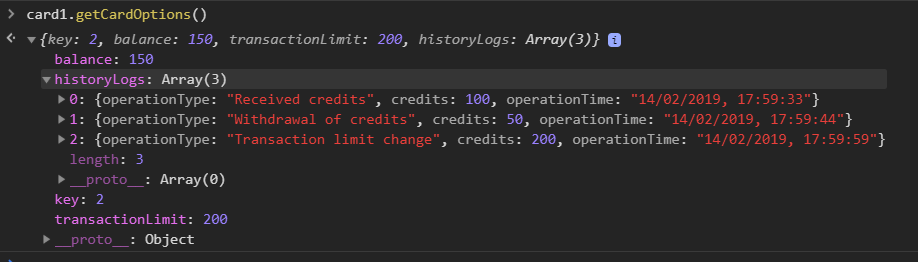


Figure 1 — Get card options function invocation example

**2)** **Create a** **UserAccount:**

Create a class `UserAccount` (for this task you can use ES6 class or regular function):

const user = new UserAccount('Bob');

Instance of this class should contain:

**User Account Properties**:

* **name** (you should pass it via constructor)
* **cards** (array of user cards. The length of array should not be longer than 3)

**User Account Methods**:

* **addCard.** should create a new card (use ‘userCard’ function) and add it to user cards:

user.addCard();

Please note, that User should have <= 3 cards.

* **getCardByKey.** Receives number in range {1; 3} and returns card object.

/\*

\* Returns an object e.g. {

\* key: 1,

\* balance: 150,

\* ...other card properties

\* }

\*/

user.getCardByKey(1);

## CODE

**Credits transfer example:**

let user = new UserAccount('Bob');

user.addCard()

user.addCard()

let card1 = user.getCardByKey(1);

let card2 = user.getCardByKey(2);

card1.putCredits(500);

card1.setTransactionLimit(800);

card1.transferCredits(300, card2);

card2.takeCredits(50);

console.log(card1.getCardOptions()); // see Figure 2

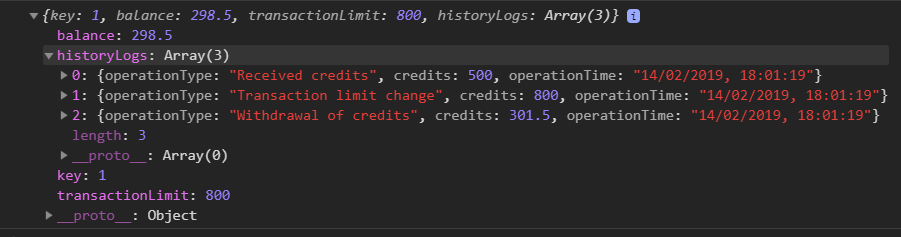


Figure 2 – The options output of the card1

console.log(card2.getCardOptions()); // see Figure 3

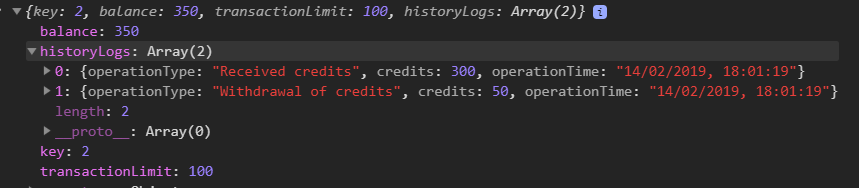


Figure 3 – The options output of the card2

## BEFORE SUBMIT

* In order to use npm package manager install nodejs (https://nodejs.org/ )
* Install eslint to check your code (npm install -g eslint)
* - open a terminal(or cmd)
* - go to src folder
* - run eslint (i.e. eslint credits\_handler.js)
* Code should be without ‘errors’
* Code should be clean, readable and well formatted
* Remove all redundant comments
* Test all your functions

## SUBMIT

* The folder should be uploaded to github repository ‘**FLX**’ into **master** branch

## USEFUL LINKS

* <https://css-tricks.com/javascript-scope-closures/>
* https://developer.mozilla.org/uk/docs/Web/JavaScript/Reference/Classes