

***Front-end Advanced***

**Training Assignment**

|  |  |
| --- | --- |
| Document Code | 25e-BM/HR/HDCV/FSOFT |
| Version | 1.1 |
| Effective Date | 7/1/2019 |

**Hanoi, mm/yyyy**

RECORD OF CHANGES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Effective Date | Change Description | Reason | Reviewer | Approver |
| 1 | 30/May/2019 | Create a new assignment | Create new | DieuNT1 | VinhNV |
| 2 | 07/Jun/2019 | Update Fsoft Template | Update | DieuNT1 | VinhNV |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

[Unit 4: Higher-order Functions 4](#_Toc44070112)

[Objectives: 4](#_Toc44070113)

[Assumptions: 4](#_Toc44070114)

[Problem Descriptions: 4](#_Toc44070115)

[Exercise 01 4](#_Toc44070116)

[Exercise 02 4](#_Toc44070117)

[Exercise 03 4](#_Toc44070118)

[Exercise 04 4](#_Toc44070119)

[Exercise 05 5](#_Toc44070120)

[Exercise 06 5](#_Toc44070121)

[Exercise 07 5](#_Toc44070122)

[Exercise 08 5](#_Toc44070123)

[Exercise 09 5](#_Toc44070124)

[Exercise 10 5](#_Toc44070125)

[Exercise 11 5](#_Toc44070126)

[Exercise 12 5](#_Toc44070127)

|  |  |
| --- | --- |
|  | **CODE: JS-A.M.A401 (map, filter, reduce)**  **TYPE: Medium**  **LOC: 200**  **DURATION: 90** |

# Unit 4: Higher-order Functions

Objectives:

* Understand Advanced JavaScript concept: Scope, Higher-Order Function
* Able to recognize variable's scope to determine its value at runtime with ease (including nested scope)
* Able to recognize and use closure to solve common problems
* Understand how to use Function as value to create abstraction
* Able to use Higher-Order function to remove duplicate code

Assumptions:

You are given the **JS-A.M.A401-problem.js** file which contains the data. You have to fullfile the requirement in each function

Problem Descriptions:

Problem 01

Use **Array.prototype.forEach** to get the full name (first\_name and last\_name) of all user and put it an array then return the array. The order in new array must be the same order as the user appear in the **users** array.

Expected output:

1. [
2. 'Eamon Harhoff',
3. 'Laney Whittam',
4. 'Lynett Twinberrow',
5. …

Problem 02

Use **Array.prototype**.**filter** to return an array of user which is male and age is under 40 and

Problem 03

Use **Array.prototype.map** to return an array of full name of each user

Problem 04

Use map to transform **users** array where the key of each record in new array is **camelCase** return that new array

Example:

1. [
2. { "id": 1, "firstName": "Eamon", "lastName": "Harhoff", "email": "eharhoff0@imageshack.us", "gender": "Male", "age": 76, "salary": 18888 },
3. { "id": 2, "firstName": "Laney", "lastName": "Whittam", "email": "lwhittam1@issuu.com", "gender": "Female", "age": 42, "salary": 15018 },
4. { "id": 3, "firstName": "Lynett", "lastName": "Twinberrow", "email": "ltwinberrow2@gov.uk", "gender": "Female", "age": 99, "salary": 13343 }
5. …
6. ]

Problem 05

Use **Array.prototye.reduce** to calculate the average age in **users** and return the result.

Problem 06

Use **Array.prototye.reduce** to implement Problem 02 – 04

Problem 07

Use sort function of Array.prototype.sort to sort the users array by field first\_name in **ascending** **order**

Problem 08

Write a function named **faMap** that takes an array, and a transformation function.

Map function have the same functionality like **Array.prototype.map**

Problem 09

Write a function named **faFilter** that takes an array, and a predicate function.

Map function have the same functionality like **Array.prototype.filter**

Problem 10

Write a function named **faReduce** that take an array, a function and a default value.

Map function have the same functionality like **Array.prototype.reduce**

Problem 11

Reuse function **faReduce** of Problem 10 to write function **problem1101** (works like **faMap**) and function **problem1102** (works like **faFilter**) without using loops (for, while, do-while).

Problem 12

1. Use reduce to create function **problem1201** which will calculate the sum of every item in array.
2. Use reduce to create function **problem1202** which will calculate the product of every item in array.
3. Use reduce to create function **problem1203** which will reverse the position of every item in array.

Problem 13

Use **faReduce** to create function **getProp** with 2 parameters an object and the path to the property inside object. getProp will extract the property inside object in a safe manner than access the property directly. Nested property is support by join them together with ‘.’. Example path: ‘clazz.frontend’ means access the **clazz** property then **frontend** property. Array is supported through index, ‘addresses.0’ means element at index 0 of array **addrresses**

Check example below:

1. var student = {
2. name: 'Nguyen Van A',
3. addresses: [
4. {
5. type: 'personal',
6. location: 'Hanoi'
7. },
8. {
9. type: 'work',
10. location: 'Hoa Lac'
11. }
12. ],
13. clazz: {
14. frontend: {
15. name: 'Angular'
16. }
17. }
18. }
19. getProp(student, 'name'); // Nguyen Van A same as student.name or student['name']
20. getProp(student, 'addresses.0.location') // Hanoi same as student.addresses[0].location
21. getProp(student, 'clazz.frontend.name') // Angular same as student.clazz.frontend.name
22. getProp(student, 'hobbbies.name') // undefined no field hobbies in student if we do student.hobbies.name we will get Error