

***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

[Day 11-12. Unit 6: ES6 Collections 4](#_Toc43804570)

[Objectives: 4](#_Toc43804571)

[Problem 01 4](#_Toc43804572)

[Problem 02 4](#_Toc43804573)

[Problem 03 6](#_Toc43804574)

|  |  |
| --- | --- |
|  | **CODE: FEA.M.A502 (ES6 02)**  **TYPE: Medium**  **LOC: 300**  **DURATION: 180** |

# Day 11-12. Unit 6: ES6 Collections

Objectives:

* Understand the History of JavaScript and ES6 (the most populator JavaScript version)
* Understand ES6 features: Arrow function, Classes, Block scope, Rest/Spread, Destructuring, Template string, Map/Set
* Able to use ES6 features to create more readable and cleaner code

Problem 01

Your task is to use ES6 Generators to implement Object Spread

1. var numbers = {
2. // ..
3. };
4. // should print 0..100 by step 1
5. // 0 1 2 … 100
6. for (let num of numbers) {
7. console.log(num);
8. }
9. // should print 6..30 by step 4
10. // 6 10 14 … 30
11. for (let num of /\*..\*/) {
12. console.log(num);
13. }

Problem 02

Given the following code which calculate the fibonaci number of n:

1. function fibonaci(n) {
2. if (n <= 1) {
3. return 1;
4. }
5. return fibonaci(n - 1) + fibonaci(n - 2);
6. }

The problem with this code is we make too many duplicate computation:

|  |
| --- |
|  |

In the figure above, we can see that f(2) is computed 3 times, f(3) is computed 2 times

Your task is to use Map to memorize the Fibonacci of lower value so we don’t have to recompute it again.

|  |
| --- |
|  |

For example: if we have computed Fibonacci for n – 2 and n – 3 (left branch) then we don’t need to compute anymore for the right branch.

Problem 03

You are given a list of object represent a Person.

Each person have following properties: **id**, **name**, **age**.

Your task is to remove duplicates from that lists. We define that a person is duplicate of another person if they have the same **id** and we will keep the previous Person.

Example:

1. [
2. {
3. id: 1,
4. name: 'Dung',
5. age: 20
6. },
7. {
8. id: 2,
9. name: 'Diu',
10. age: 20
11. },
12. {
13. id: 3,
14. name: 'Ky',
15. age: 20
16. },
17. {
18. id: 1,
19. name: 'Hai',
20. age: 22
21. }
22. ]

In the above example, id = 1 is duplicated, and we will keep the person with id = 1 and name = ‘Dung’

Expected Output:

1. [
2. {
3. id: 1,
4. name: 'Dung',
5. age: 20
6. },
7. {
8. id: 2,
9. name: 'Diu',
10. age: 20
11. },
12. {
13. id: 3,
14. name: 'Ky',
15. age: 20
16. }
17. ]