



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 7-8. Unit 4: Async Programming	4
Objectives:	4
Assumptions:	4
Problem 01:	4
Problem 02:	6
Problem 03:	6



CODE:	Async-JS.M.A102 (Async 02)
TYPE:	Medium
LOC:	300
DURATION:	180

Day 2. Unit 2-5: Async Programming

Objectives:

- Understand how to abstract asynchronous task in JavaScript (Async Programming)
- Understand the core concept of Async in JavaScript: the Event Loop
- Understand the concept of Parallel Programming and how JavaScript implement that concept
- Able to working with Async API with ease (Callback, Promise, async/await)

Assumptions:

You are given a predefined project from the file **Async-JS.M.A102 (Async 02).zip**.

The project is a simple mini app which display a list of users with his posts and comments. Investigate it carefully before reading the rest.

To get data from an API, we will use \$.get from jQuery

```
1. $.get('https://jsonplaceholder.typicode.com/users', function(users) {  
2.   // if we are here, which means the data is retrieved from the API  
3.   console.log(users);  
4. });
```

In the above example, \$.get take 2 parameters:

1. The URL to the API
2. The callback function that will be called when the data is retrieved from the first parameter

Problem 01:

Your task is to do the following using **Callback style**.

1. Get 10 users from this API (<https://jsonplaceholder.typicode.com/users>)

Example output:

```
1. [  
2.   {  
3.     id: 1,  
4.     name: 'Leanne Graham',  
5.     username: 'Bret',  
6.     email: 'Sincere@april.biz',  
7.     address: {  
8.       street: 'Kulas Light',  
9.       suite: 'Apt. 556',  
10.      city: 'Gwenborough',  
11.      zipcode: '92998-3874',  
12.      geo: {  
13.        lat: '-37.3159',  
14.        lng: '81.1496'  
15.      }  
16.    },  
17.    phone: '1-770-736-8031 x56442',
```

```
18.     website: 'hildegard.org',
19.     company: {
20.       name: 'Romaguera-Crona',
21.       catchPhrase: 'Multi-layered client-server neural-net',
22.       bs: 'harness real-time e-markets'
23.     }
24.   },
25.   .....
26. ];
```

2. For each user get his posts using this API where `userId` is the id of a user
(<https://jsonplaceholder.typicode.com/posts?userId=<<userId>>>)

For example: to get the posts of `userId` 2, you need to call `$.get` from this API
(<https://jsonplaceholder.typicode.com/posts?userId=2>)

3. For each post get all its comments using this API
(<https://jsonplaceholder.typicode.com/comments?postId=1>)
4. Combine all 3 data together to get a list of Users which have a list of posts and each post have a list of comments

Your response will something like this (other field are omitted for the sake of readability)

```
1. [
2.   {
3.     id: 1,
4.     posts: [
5.       {
6.         userId: 1,
7.         id: 10,
8.         comments: {
9.           commentId: 123,
10.          postId: 123
11.          name: ...
12.        }
13.      }
14.    ]
15.    ....
16.  },
17.  ...
18. ];
```

5. Once you the data from Step 4, call the following function

```
1. $scope.$apply(function() {
2.   $scope.users = users;
3. });
```

Where the variable **users** stored a list of Users which have a list of posts and each post have a list of comments. If you are doing correctly, you should see the figure below:

- [UserID 1] Leanne Graham has 10 posts
 - [PostId 1] Title: sunt aut facere repellat provident occaecati excepturi optio reprehenderit
 - [CommentId 1] Title: id labore ex et quam laborum by Eliseo@gardner.biz
 - [CommentId 2] Title: quo vero reiciendis velit similique earum by Jayne_Kuhic@sydney.com
 - [CommentId 3] Title: odio adipisci rerum aut animi by Nikita@garfield.biz
 - [CommentId 4] Title: alias odio sit by Lew@alysha.tv
 - [CommentId 5] Title: vero eaque aliquid doloribus et culpa by Hayden@althea.biz
 - [PostId 2] Title: qui est esse
 - [CommentId 6] Title: et fugit eligendi deleniti quidem qui sint nihil autem by Presley.Mueller@myrl.com
 - [CommentId 7] Title: repellat consequatur praesentium vel minus molestias voluptatum by Dallas@ole.me
 - [CommentId 8] Title: et omnis dolorem by Mallory_Kunze@marie.org
 - [CommentId 9] Title: provident id voluptas by Meghan_Littel@rene.us
 - [CommentId 10] Title: eaque et deleniti atque tenetur ut quo ut by Carmen_Keeling@caroline.name
 - [PostId 3] Title: ea molestias quasi exercitationem repellat qui ipsa sit aut
 - [CommentId 11] Title: fugit labore quia mollitia quas deserunt nostrum sunt by Veronica_Goodwin@timothy.net
 - [CommentId 12] Title: modi ut eos dolores illum nam dolor by Oswald.Vandervort@leanne.org
 - [CommentId 13] Title: aut inventore non pariatur sit vitae voluptatem sapiente by Kariane@jadyn.tv
 - [CommentId 14] Title: et officiis id praesentium hic aut ipsa dolorem repudiandae by Nathan@solon.io
 - [CommentId 15] Title: debitis magnam hic odit aut ullam nostrum tenetur by Maynard.Hodkiewicz@roberta.com
 - [PostId 4] Title: eum et est occaecati
 - [CommentId 16] Title: preferendis temporibus delectus optio ea eum ratione dolorum by Christine@ayana.info
 - [CommentId 17] Title: eos est animi quis by Preston_Hudson@blaise.tv
 - [CommentId 18] Title: aut et tenetur ducimus illum aut nulla ab by Vincenza_Klocko@albertha.name
 - [CommentId 19] Title: sed impedit rerum quia et et inventore unde officiis by Madelynn.Gorczy@darion.biz
 - [CommentId 20] Title: molestias expedita iste aliquid voluptates by Mariana_Orn@preston.org
 - [PostId 5] Title: nesciunt quas odio
 - [CommentId 21] Title: aliquid rerum mollitia qui a consectetur eum sed by Noemie@marques.me
 - [CommentId 22] Title: porro repellendus aut tempore quis hic by Khalil@emile.co.uk

Problem 02:

You must do the same tasks like Exercise 1, but you have to use Promise API instead of Callback Style.

To do so, you must first convert the \$.get to a function that return a Promise.

Problem 03:

You must do the same tasks like Exercise 1, but you have to use async/await API.