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

CODE:	JS-A.M.A501 (OOP)
TYPE:	Medium
LOC:	300
DURATION:	120 minutes

Unit 5: OOP and Prototypes

Objectives

- Understand the core concepts, step to analysis, design and program to OOP
- Able to recognize class, field, method, interactions between classes
- Understand the 4 attributes of OOP: Abstraction, Inheritance, Encapsulation and Polymorphism
- Undertand OOP Principles to create Abstract class, Interface

Specification 1

Write a class **Vec** that represents a vector in two-dimensional space. It takes x and y parameters (numbers), which it should save to properties of the same name.
Give the **Vec** prototype two methods, **plus** and **minus**, that take another vector as a parameter and return a new vector that has the sum or difference of the two vectors' (this and the parameter) *x* and *y* values.
Add a getter property length to the prototype that computes the length of the vector—that is, the distance of the point (*x*, *y*) from the origin (0, 0).

Expected Output:

```
// Your code here.
console.log(new Vec(1, 2).plus(new Vec(2, 3)));
// => Vec{x: 3, y: 5}
console.log(new Vec(1, 2).minus(new Vec(2, 3)));
// => Vec{x: -1, y: -1}
console.log(new Vec(3, 4).length);
// => 5
```

Specification 2

The standard JavaScript environment provides another data structure called Set. Like an instance of Map, a set holds a collection of values. Unlike Map, it does not associate other values with those—it just tracks which values are part of the set. A value can be part of a set only once—adding it again doesn't have any effect.
Write a class called Group (since Set is already taken). Like Set, it has **add**, **delete**, and **has** methods. Its constructor creates an empty group, add adds a value to the group (but only if it isn't already a member), delete removes its argument from the group (if it was a member), and has returns a Boolean value indicating whether its argument is a member of the group.
Use the **==** operator, or something equivalent such as **indexOf**, to determine whether two values are the same.
Give the class a static **from** method that takes an array as argument and creates a group that contains all the values produced by iterating over it.

Expected Output:

```
class Group {
  // Your code here.
}

let group = Group.from([10, 20]);
console.log(group.has(10));
// → true
console.log(group.has(30));
// → false
group.add(10);
group.delete(10);
console.log(group.has(10));
// → false
```

Specification 3

Earlier in the chapter I mentioned that an object's hasOwnProperty can be used as a more robust alternative to the in operator when you want to ignore the prototype's properties. But what if your map needs to include the word "hasOwnProperty"? You won't be able to call that method anymore because the object's own property hides the method value.
Can you think of a way to call hasOwnProperty on an object that has its own property by that name?

Expected Output:

```
1.  let map = { one: true, two: true, hasOwnProperty: true };
2.
3.  // Fix this call
4.  console.log(map.hasOwnProperty.call('one'));
5.  // → true
```