**Báo cáo**

*Số 15*

**Họ Tên**: Nguyễn Thái Sơn

**Ngày**: 05-12-2019

Task name

* Nội dung tìm hiểu : Function và class Object trong Javascript
* Tiến độ (%) 60%
* Vấn đề gặp phải

**JavaScript Classes**

* Với ES6, còn được gọi là ECMAScript2015 đã giới thiệu classes
* Class là một hàm, nhưng thay vì sử dụng hàm với từ khóa function, chúng ta sử dụng từ khóa class và các thuộc tính được gán trong một phương thức constructor

**Định nghĩa Class**

* Sử dụng từ khóa class để khởi tạo một lớp, và luôn luôn sử dụng phương thức contructor
* Phương thức contructor sẽ được gọi mỗi lần đối tượng lớp được khởi tạo

Vd :

class Car {  
  constructor(brand) {  
    this.carname = brand;  
  }  
}

* Bây giờ có thể khởi tạo đối tượng sử dụng lớp Car :
* class Car {  
    constructor(brand) {  
      this.carname = brand;  
    }  
  }  
  mycar = new Car("Ford");
* Phương thức contructor được gọi tự động khi đối tượng lớp được khởi tạo

**Phương thức**

* Phương thức contructor rất đặc biệt, đó là nơi bạn khởi tạo các thuộc tính, nó được gọi tự động khi một lớp được khởi tạo và nó phải có tên xác định là Contructor. Thực tế nếu không có phương thức contructor Javascript sẽ thêm một phương thức contructor vô hình và trống
* Cũng có thể tạo ra những phương thức riêng.
* Vd :
* class Car {  
    constructor(brand) {  
      this.carname = brand;  
    }  
    present() {  
      return "I have a " + this.carname;  
    }  
  }  
    
  mycar = new Car("Ford");  
  document.getElementById("demo").innerHTML = mycar.present();

**Phương thức tĩnh**

* Static methods được định nghĩa trên chính lớp đó và không phải trên nguyên mẫu
* Điều đó có nghĩa là không thể gọi một phương thức tĩnh trên đối tượng nhưng lại có thể gọi trên lớp.
* Vd :

class Car {  
  constructor(brand) {  
    this.carname = brand;  
  }  
  static hello() {  
    return "Hello!!";  
  }  
}  
  
mycar = new Car("Ford");  
  
//Call 'hello()' on the class Car:  
document.getElementById("demo").innerHTML = Car.hello();

**Kế thừa**

* Có thể tạo ra class kế thừa bằng cách sử dụng từ khóa extends
* Một lớp tạo ra kế thừa một lớp kế thừa thì lớp đó sẽ có các phương thức của lớp cho kế thừa
* Vd :

class Car {  
  constructor(brand) {  
    this.carname = brand;  
  }  
  present() {  
    return 'I have a ' + this.carname;  
  }  
}  
  
class Model extends Car {  
  constructor(brand, mod) {  
    super(brand);  
    this.model = mod;  
  }  
  show() {  
    return this.present() + ', it is a ' + this.model;  
  }  
}  
  
mycar = new Model("Ford", "Mustang");  
document.getElementById("demo").innerHTML = mycar.show();

* Phương thức super() dùng để chỉ lớp cha
* Bằng cách gọi phương thức super() trong phương thức contructor, chúng ta gọi phương thức contructor của cha và có quyền truy cập vào các thuộc tính và phương thức của lớp cha.

**Getter và Setter**

* Các lớp cũng cho phép sử dụng setter và getter
* Rất ưu việt khi sử dụng setter và getter cho các hàm thuộc tính
* Để sử dụng getter và setter trong lớp sử dụng các từ khóa get và set
* Vd :

class Car {  
  constructor(brand) {  
    this.carname = brand;  
  }  
  get cnam() {  
    return this.carname;  
  }  
  set cnam(x) {  
    this.carname = x;  
  }  
}  
  
mycar = new Car("Ford");  
  
document.getElementById("demo").innerHTML = mycar.cnam;

* Ngay cả khi getter là một phương thức bạn không sử dụng dấu ngoặc đơn khi bạn muốn lấy giá trị thuộc tính
* Tên của phương thức getter / setter không thể giống với tên của thuộc tính
* Vd :
* class Car {  
    constructor(brand) {  
      this.\_carname = brand;  
    }  
    get carname() {  
      return this.\_carname;  
    }  
    set carname(x) {  
      this.\_carname = x;  
    }  
  }  
    
  mycar = new Car("Ford");  
    
  document.getElementById("demo").innerHTML = mycar.carname;

**JavaScript Objects**

* Trong Javascript gần như mọi thứ là một đối tượng
  + Booleans có thể là đối tượng (nếu được xác định bằng từ khóa new)
  + Number có thể là đối tượng (nếu được xác định bằng từ khóa new)
  + Strings có thể là đối tượng (nếu được xác định bằng từ khóa new)
  + Dates luôn luôn là đối tượng
  + Maths luôn luôn là đối tượng
  + Biểu thức chính quy luôn luôn là đối tượng
  + Array luôn luôn là đối tượng
  + Hàm luôn luôn là đối tượng
  + Objects luôn luôn là đối tượng
* Tất cả các giá trị Javascript, ngoại trừ nguyên thủy là các đối tượng

**Object là biến**

* Đối tượng cũng là biến, nhưng các đối tượng có thể chứa nhiều giá trị
* Các giá trị được viết dưới dạng cặp tên: giá trị

Vd :

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

**Tạo một đối tượng trong javascript**

* Với Javascript bạn có thể xác định và tạo các đối tượng của riêng mình
* Có nhiều cách để tạo ra đối tượng mới:
  + Xác định và tạo một đối tượng sử dụng một đối tượng bằng chữ
  + Xác định và tạo một đối tượng sử dụng từ khóa new
  + Xác định một hàm tạo đối tượng và sau đó tạo các đối tượng của kiểu được xây dựng
* Trong ECMAScript 5, cũng có thể tạo ra một đối tượng bằng hàm Object.create()

**Sử dụng một đối tượng**

* Đây là cách dễ nhất để tạo một đối tượng trong Javascript
* Sử dụng một đối tượng bằng chữ, cả hai bạn xác định và tạo một đối tượng trong một câu lệnh
* Một đối tượng bằng chữ là một danh sách các cặp tên : giá trị
* Vd :

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

**Sử dụng từ khóa new trong Javascript**

Vd :

var person = new Object();  
person.firstName = "John";  
person.lastName = "Doe";  
person.age = 50;  
person.eyeColor = "blue";

**Các đối tượng trong Javascript có thể thay đổi**

* Các đối tượng có thể thay đổi, được giải quyết bằng tham chiếu, không phải theo giá trị

**Thuộc tính Javascript**

* Thuộc tính là các giá trị được liên kết với một đối tượng trong Javascript
* Một đối tượng Javascript là một tập hợp các thuộc tính không có thứ tự
* Các thuộc tính thường có thể được thay đổi, thêm và xóa, nhưng một số chỉ có thể được đọc

**Truy cập thuộc tính trong Javascript**

* Cú pháp để truy cập vào thuộc tính trong Javascript

*objectName.property*// person.age

Hoặc

*objectName*["*property*"]      // person["age"]

Hoặc

*objectName*[*expression*]      // x = "age"; person[x]

## **JavaScript for...in Loop**

* Câu lệnh for … in câu lệnh lặp qua các thuộc tính của một đối tượng
* Cú pháp

for (*variable* in *object*) {  
*// code to be executed*  
}

* Khối mã bên trong vòng lặp for in sẽ được thực thi một lần cho mỗi thuộc tính
* Vòng lặp thông qua các thuộc tính của một đối tượng
* Vd :

var person = {fname:"John", lname:"Doe", age:25};  
  
for (x in person) {  
  txt += person[x];  
}

**Thêm thuộc tính mới**

* Có thể thêm các thuộc tính mới vào một đối tượng hiện có bằng cách chỉ cần cho nó một giá trị
* Vd :

person.nationality = "English";

**Xóa bỏ thuộc tính của đối tượng**

* Từ khóa new sẽ xóa một thuộc tính khỏi một đối tượng
* Vd :

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};  
delete person.age;   // or delete person["age"];