**Báo cáo**

*Số 14*

**Họ Tên**: Nguyễn Thái Sơn

**Ngày**: 04-12-2019

Task name

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

**Javascript Functions**

* Hàm trong Javascript là một khối code được thiết kế để thực hiện một tác vụ cụ thể
* Hàm trong Javascript được thực thi khi ta gọi hàm.

vd :

function myFunction(p1, p2) {  
  return p1 \* p2;   // The function returns the product of p1 and p2  
}

* Hàm trong Javascript được xác định với từ khóa function theo sau là tên hàm, tiếp theo là dấu ()
* Tên hàm có thể chứa các chữ cái, chữ số, dấu gạch dưới và ký hiệu $.
* Bên trong các dấu ngoặc đơn có thể bao gồm tên các tham số được phân tách thành dấu phẩy (param1, param2)
* Nội dung của hàm được đặt bên trong cặp ngoặc {}

Vd :

function name(parameter1, parameter2, parameter3) {  
  // code to be executed  
}

* Các tham số hàm được liệt kê bên trong dấu ngoặc () trong định nghĩa hàm
* Các đối số hàm là các giá trị mà hàm nhận được khi nó được gọi
* Bên trong hàm, các đối số hoạt động như các biến cục bộ
* Hàm giống như các chương trình con trong các ngô ngữ lập trình khác

**Function Invocation**

* Code bên trong hàm sẽ chạy khi được gọi hàm khi :
  + Khi một sự kiện xảy ra (khi người dùng nhấn vào nút)
  + Khi nó được gọi từ code Javascript
  + Tự động (gọi lại)

**Function Return**

* Hàm trong javascript chạy đến câu lệnh return hàm sẽ dừng thực thi
* Nếu hàm được gọi ra từ một câu lệnh. Javascript sẽ “return” để thực thi mã lệnh sau câu lệnh gọi
* Các hàm thường tính một giá trị trả về. Giá trị trả về được trả lại cho người gọi

Vd :

var x = myFunction(4, 3);   // Function is called, return value will end up in x  
  
function myFunction(a, b) {  
  return a \* b;             // Function returns the product of a and b  
}

**Tại sao lại sử dụng hàm?**

* Bạn có thể xử dụng lại mã: Xác định mã một lần và sử dụng nhiều lần
* Bạn cũng có thể sử dụng cùng một mã nhiều lần với các đối số khác nhau, để tạo ra các kết quả khác nhau

**Vd:**

function toCelsius(fahrenheit) {  
  return (5/9) \* (fahrenheit-32);  
}  
document.getElementById("demo").innerHTML = toCelsius(77);

**Toán tử () gọi hàm**

* Sử dụng ví dụ trên toCelsius đề cập đến đối tượng hàm và toCelsius() đề cập đến kết quả hàm
* Truy cập hàm không có () sẽ trả về định nghĩa hàm thay vì kết quả hàm

Vd :   
function toCelsius(fahrenheit) {  
  return (5/9) \* (fahrenheit-32);  
}  
document.getElementById("demo").innerHTML = toCelsius;

**Hàm được sử dụng làm giá trị biến**

* Các hàm có thể được sử dụng giống như các bạn sử dụng các biến, trong tất cả các loại công thức bài tập và tính toán
* Vd :

var x = toCelsius(77);  
var text = "The temperature is " + x + " Celsius";

* Bạn có thể sử dụng hàm trực tiếp làm giá trị biến :

var text = "The temperature is " + toCelsius(77) + " Celsius";

**Biến cục bộ**

* Các biến khai báo trong hàm sẽ trở thành biến cục bộ của hàm đó.
* Các biến cục bộ chỉ có thể được truy cập từ bên trong hàm

Vd :

// code here can NOT use carName  
  
function myFunction() {  
  var carName = "Volvo";  
  // code here CAN use carName  
}  
  
// code here can NOT use carName

* Vì các biến cục bộ chỉ được nhận dạng bên trong các hàm của chúng, các biến có cùng tên có thể được sử dụng trong các hàm khác nhau
* Biến cục bộ được tạo khi hàm bắt đầu và bị xóa khi hàm hoàn thành

**Arrow Function**

* Arrow Function đã được giới thiệu trong ES6
* Arrow Function cho phép chúng ta viết hàm ngắn gọn hơn
* Trước khi sử dụng Arrow Function

hello = function() {  
  return "Hello World!";  
}

* Khi sử dụng Arrow Function

hello = () => {  
  return "Hello World!";  
}

* Arrow Function được viết ngắn gọn hơn nữa. Nếu hàm chỉ có một câu lệnh và câu lệnh trả về giá trị, Bạn có thể xóa dấu ngoặc và từ khóa return.

hello = () => "Hello World!";

* Điều này chỉ hoạt động nếu chức năng chỉ có một câu lệnh
* Nếu có tham số hãy cho chúng vào trong ngoặc đơn

hello = (val) => "Hello " + val;

* Trong thực tế nếu bạn chỉ có một tham số bạn cũng có thể bỏ qua dấu ngoặc đơn

hello = val => "Hello " + val;