Buổi 9
Hàm

I. Khái niệm

Hàm (*function*) là 1 đoạn chương trình độc lập, thực hiện một công việc cụ thể.

Hàm được sử dụng trong lập trình nhằm mục đích:

- Tránh việc lặp đi lặp lại các đoạn chương trình giống nhau.
- Giúp cho chương trình gọn gàng, rõ ràng, dễ hiểu.
- Giúp cho việc kiểm tra lỗi, sửa lỗi dễ dàng hơn.

Trong ngôn ngữ lập trình có 2 loại hàm:

- Hàm chuẩn: là những hàm đã được xây dựng sẵn trong các thư viện như hàm `alert()`, `confirm()`, `prompt()`, `Number()`, `isNaN()`…
- Hàm tự định nghĩa: là những hàm do lập trình viên tự viết.

Một chương trình có thể có nhiều hàm, và các hàm có thể gọi lẫn nhau.

II. Cách cài đặt và sử dụng hàm

Cú pháp để cài đặt 1 hàm trong JS như sau:

```
function <Tên hàm>() {
      // Các câu lệnh cần thực hiện
}
```

Trong đó: `<Tên hàm>` do lập trình viên tự đặt, tuân theo quy tắc đặt tên của C/C++.

Ví dụ: Hàm hiển thị 1 lời chào ra màn hình:

```
function HienThiLoiChao() {
      alert("Xin chào");
}
```

Để gọi thực hiện lại hàm đã được cài đặt chỉ cần gọi thông qua tên hàm.

Ví dụ:

```
<button onclick="HienThiLoiChao()">Click me</button>
```

Hàm có thể trả về giá trị để tiếp tục tính toán dựa trên giá trị này. Hàm được gọi ở đâu sẽ trả giá trị về tại đó.

Câu lệnh `return` dùng để trả về 1 giá trị, sau đó dừng và thoát khỏi hàm ngay lập tức. 1 hàm chỉ có thể trả về 1 giá trị nhưng có thể có nhiều câu lệnh `return`.

Ví dụ: Hàm tính tổng $S = 1 + 2 + 3 + \ldots + 10$:

```
function TinhTong() {
      var s = 0;
      for (i = 1; i <= 10; i++) {
            s += i;
      }
      return s;
}
```

Hàm có trả về giá trị có thể được gán cho 1 biến, có thể xuất ra màn hình, hoặc dùng trong các biểu thức tính toán…

Ví dụ:

```
function HienThiKQ() {
	// Gán cho 1 biến
	var tong = TinhTong();

	// Xuất ra màn hình
	alert("Tổng các số từ 1 đến 10 = " + TinhTong());
}
```

III. Tham số của hàm

Tham số (*parameter*[1]) là các giá trị được gửi vào hàm để thực hiện công việc. Việc gửi tham số vào hàm được gọi là truyền tham số.

Danh sách tham số được liệt kê trong cặp dấu ngoặc `()` của hàm theo cú pháp sau:

```
<Tên hàm>(<tên tham số 1>, <tên tham số 2>,…)
```

Khi gọi hàm, cần truyền tham số theo đúng thứ tự, đúng số lượng.

Ví dụ: Hàm tính tổng 2 số nguyên:

- Cài đặt hàm:

```
function TinhTong(a, b) {
	return a + b;
}
```

- Gọi hàm:

```
function F() {
	var x = 5, y = 8;
	alert("x + y = " + TinhTong(x, y));
}
```

Nếu truyền tham số không đủ số lượng, các tham số còn thiếu sẽ mang giá trị `undefined` và đôi khi sẽ gây ra lỗi. Do đó, có thể khai báo giá trị mặc định cho tham số khi cài đặt hàm như sau:

```
function TinhTong(a = 0, b = 0) {
	return a + b;
}
```

IV. Phạm vi sử dụng của biến

Mỗi biến chỉ được phép sử dụng trong đoạn chương trình mà nó được khai báo. Ở ngoài đoạn chương trình đó, biến sẽ bị hủy và không còn có thể sử dụng được. Đoạn chương trình được bao bởi cặp ngoặc `{ }`.

Phạm vi sử dụng của biến được gọi là tầm vực (*scope*).

Có 2 loại biến:

- Biến toàn cục (*global variable*): Khai báo ngang hàng với các hàm, có thể sử dụng ở bất kì đâu trong chương trình.
- Biến cục bộ (*local variable*): Khai báo trong hàm hoặc trong 1 đoạn lệnh (cấu trúc rẽ nhánh, vòng lặp…), chỉ có thể sử dụng trong đoạn lệnh đó.

Ví dụ:

```
var a;          // a là biến toàn cục

function F() {
```

[1] Còn có tên khác là *argument*, tuy nhiên giữa parameter và argument có sự khác nhau về mặt ý nghĩa.

```
        var c;          // c là biến cục bộ của hàm F()
        for (i = 0; i < n; i++) {   // i là biến cục bộ của vòng lặp for
            …
        }
    }
```

V. Bài tập

Thực hiện các bài tập sau, mỗi bài tập nằm trong 1 trang web khác nhau.

Yêu cầu: Sử dụng hàm.

1. Nhập 2 số nguyên. Tính tổng, hiệu, tích và thương của 2 số nguyên đó.
2. Nhập vào 1 trong 3 lựa chọn, sau đó thực hiện lựa chọn đó:
   - Lựa chọn 1: Nhập bán kính, tính chu vi và diện tích hình tròn.
   - Lựa chọn 2: Nhập độ dài 2 cạnh, tính chu vi và diện tích hình chữ nhật.
   - Lựa chọn 3: Nhập độ dài cạnh, tính chu vi và diện tích hình vuông.
3. Nhập số nguyên dương n. Cho biết n có phải số nguyên tố hay không.
4. Nhập số nguyên dương n. Xuất ra màn hình các số nguyên tố không vượt quá n.
5. Nhập số nguyên dương n. Xuất ra màn hình n số nguyên tố đầu tiên.
6. Nhập 2 số nguyên dương a và b. Xuất ra màn hình ước chung lớn nhất và bội chung nhỏ nhất của a và b.