Xây dựng hàm trong JavaScript hàm ẩn danh và biểu thức hàm

Xây dựng một hàm trong JavaScript, sử dụng tham số hàm và các giá trị trả về khi gọi hàm, biểu thức hàm và hàm ẩn danh

Hàm trong JavaScript

Hàm là một khối mã lệnh được viết nhằm một múc đích nào đó. Xây dựng hàm mang lại một số lợi ích như sử dụng lại mã đã viết, một khối mã lệnh với các tham số khác nhau mang lại các kết quả khác nhau. Một hàm thi hành khi hàm đó được gọi.

Định nghĩa hàm

Hàm trong JavaScript sử dụng từ khóa `function` tiếp theo là `tên hàm` và các tham số nếu có trong ngoặc `()`. Khối mã của hàm nằm trong khối ngoặc nhọn `{}`

```
function name_funtion() {
  //các mã của hàm
}
```

Tên hàm có thể chứa các ký tự, số, gạch dưới ... (tương tự quy tắc tên biến).

Gọi hàm

Để thi hành hàm bạn cần gọi nó. Để gọi hàm viết lại tên hàm và các tham số truyền vào hàm, nhớ kết thúc bằng dấu `;` Một hàm có thể được gọi bao nhiêu lần là tùy bạn. Ví dụ:

```
function myFunction() {
  alert("Calling a Function!");
}

myFunction();
//Hiện thông báo:  "Calling a Function!"
```

Tham số hàm

Các tham số của hàm bạn cần liệt kê sau tên hàm, mỗi tham số cách nhau bởi dấu `,`

```
functionName(param1, param2, param3 = default) {
   // các dòng code trong hàm
}
```

Tham số nào muốn gán giá trị mặc định (khi gọi hàm mà không chỉ ra tham số đó thì tự động lấy mặc định) thì gán ngay ở khái báo hàm, ở trên `param3 = default`

Khi gọi hàm có tham số, bạn cần truyền tham số là giá trị thực tế để hàm thi hành

```
function sayHello(name) {
   alert("Hi, " + name);
}

sayHello("David"); //Hiện thị thông báo "Hi, David"

sayHello("Sarah"); //Hiện thị thông báo "Hi, Sarah"
```

Hàm có nhiều tham số được sử dụng một cách tương tự, nhớ là khi gọi hàm các tham số cần truyền đúng theo tứ tự qua tên hàm

```
function sayHello(name, age) {
  document.write( name + " is " + age + " years old.");
}

sayHello("John", 20)
//Hiện thị "John is 20 years old."
```

Nếu hàm được gọi mà truyền thiếu tham số, thì tham giá trị thiếu đó của tham số được thiết lập là `undefined` nếu không có khai báo giá trị mặc định, còn ngược lại sẽ lấy giá trị mặc định

Giá trị trả về của hàm

Hàm có thể tùy chọn có giá trị trả về hay không. Nếu có giá trị trả về hàm sử dụng lệnh `return`

Khối lệnh của hàm mà gặp đoạn lệnh `return` hàm sẽ dừng thi hành và trả về biểu thức của giá trị của `return`

```
function myFunction(a, b) {
   return a * b;
}

var x = myFunction(5, 6);
// Giá trị trả về được gán vào x
// x bằng 30
```

Hàm không có `return` một giá trị cụ thể nào, thì giá trị trả về của hàm là `undefined`

```
function addNumbers(a, b) {
   var c = a+b;
   return c;
}
document.write(addNumbers(40, 2));
//In ra : 42
```

Biểu thức hàm

Đây là khái niệm quan trọng, nắm vững ngay giúp cho việc học JS dễ hơn rất nhiều. Cú pháp tạo ra một biểu thức hàm giống với khai báo hàm thông thường ở trên chỉ có điều

nó khai báo bằng biểu thức (*nó là một số hạng của biểu thức*). Biểu thức hàm hàm này có thể có tên hoặc là không có tên, nếu không có tên thì biểu thức hàm này gọi là hàm ẩn danh (anonymous).

Để tường minh ta xem xét ví dụ chi tiết như sau, giả sử có một biểu thức quen thuộc

```
var a = b;
```

Biểu thức trên có hai số hạng, số hạng b có thể là một biến, một giá trị ... Giờ nếu thay số hạng b bằng một khai báo hàm đã biết ở trên, thì lúc này ta gọi nó là một biểu thức hàm.

```
var a = function xinchao(guestname) {
    alert('Xin chào ' + guestname);
}
```

Có biểu thức hàm rồi (biểu thức hàm này có tên xinchao), giá trị của biểu thức này đã gán vào biến a nếu muốn gọi biểu thức ta dùng biến a tương tự như tên hàm để thực hiện biểu thức.

```
a("Expression");            //Xuất hiện hộp thoại có dòng chữ: Xin chào Expression
```

Bạn để ý gọi biểu thức hàm ta dùng đến số hạng biến a chứ không thể dùng tên xinchao. Nếu bạn gọi sẽ có lỗi

```
xinchao("Expression");      //Lỗi vì tên hàm xinchao không có
```

Như vậy, tên này khai nhưng không thể dùng nó để gọi biểu thức, tên này xuất hiện trong bộ nhớ call stack giúp cho việc debug, giám sát ...Ví dụ bạn xem a là gì?

```
console.log(a);             //Cho biết a là một biểu thức hàm, có tên để bạn dễ
quản lý
```

Hàm ẩn danh anonymous

Chính là biểu thức hàm ở trên, nhưng trong phần khai báo bỏ tên đi (không tên). Ở ví dụ trên biểu thức hàm bỏ đi tên xinchao thì lúc này nó trở thành hàm ẩn danh, cách gọi biểu thức vẫn tương tự

```
//sau từ khóa function không có tên => thành ẩn danh

var a = function (guestname) {
    alert('Xin chào ' + guestname);
}
a("Expression");
```

Biểu thức hàm chạy ngay lập tức IIFE

Trong nhiều trường hợp, bạn khai báo biểu thức hàm rồi dùng chỉ một lần. Trong những trường hợp như vậy, bạn có thể áp dụng mô hình lập trình có tên IIFE (biểu thức hàm chạy luôn - Immediately Invokable Function Expression), để code sáng sửa và ngắn gọn hơn (nếu không thích thì cứ việc khai báo rồi gọi biểu thức cũng không sao). Ví dụ biểu thức sau sẽ chạy luôn!

```
(function (a,b,c) {
   var tong = a + b + c;
   //...
   alert(tong);
})(5,6,7);
```

Trên đây tìm hiểu cơ bản về hàm, sau khi tìm hiểu về các đối tượng trong JS, chúng ta tiếp tục với một số loại hàm đặc biệt như closures, biểu thức hàm mũi tên.

Hàm rút gọn =>

ECMAScript 6 đưa ra thêm cách định nghĩa biểu thức hàm một cách ngắn gọn, dùng kỹ hiệu mũi tên =>

Cú pháp cơ bản như sau

```
//(1) Không có tham số
() => { ... }

//(2) Có 1 tham số
param => { ... }

//(3) Dạng tổng quát
(param1, param2) => { ... }
```

Biểu thức hàm thông thường	Sử dụng dạng =>

`let x = function() {`
`  console.log("Xin Chào");`
`}`
	`let x = () => {`
	`  console.log("Xin Chào");`
	`}`

`let x = function(a) {`
`  console.log(a);`
`}`
	`let x = a => {`
	`  console.log(a);`
	`}`

`let x = function(a, b) {`
`  return a + b;`
`}`
	`let x = (a,b) => {`
	`  return a + b;`
	`}`