Cập nhật tháng 8 năm 2024

[Bài đọc] Giá trị trả về và Gọi hàm

1. Giá trị trả về

- Khi hàm kết thúc quá trình thực thi, nó có thể trả về giá trị cho nơi gọi hàm bằng cách sử dụng từ khóa return.

Ví dụ về giá trị trả về đơn giản:

```
<!doctype html>
<html>
<head>
   <title>Return Value</title>
</head>
<body>
<script type="text/javascript">
   function multiplyNums(x) {
       return x * 2;
   }

   let theNumber = 10;
   let result = multiplyNums(theNumber);
   alert(result);
</script>
</body>
</html>
```

- Ví dụ này tạo hàm tên là multiplyNums với giá trị đầu vào được gán cho biến x. Hàm có chức năng trả về giá trị gấp đôi của đối số.

```
function multiplyNums(x) {
   return x * 2;
}
```

- Sau đó, biến theNumber được khởi tạo:

```
let theNumber = 10;
```

- Tiếp theo, một biến nữa có tên result được tạo. Biến này lưu kết quả của lời gọi hàm multiplyNums. Hàm multiplyNums sử dụng biến theNumber với vai trò đối số.

```
let result = multiplyNums(theNumber);
```

- Bạn có thể trả về giá trị từ bất cứ vị trí nào trong hàm chứ không chỉ ở cuối hàm. Sử dụng lệnh return trong khối lệnh điều kiện hoặc sau vòng lặp là cách làm phổ biến. Ví dụ như sau:

```
function myFunction(x) {
   if (x == 1) {
       return true;
   } else {
       return false;
   }
}
```

- Tuy vậy, bạn cần chú ý vị trí đặt lệnh return, vì khi gặp lệnh return, hàm sẽ trả lại giá trị và không thực thi những dòng mã sau đó. Ví dụ, đoạn mã như sau sẽ không thực thi như ý định của bạn:

```
function myFunction(){
   let count = 0;
   let firstNum = 48;
   return;
   var secondNum = 109;
}
```

- Đoạn mã này sẽ không bao giờ khởi tạo biến secondNum.

2. Gọi hàm

- Bạn sẽ luôn gọi hàm với một vài đối số hoặc chỉ với cặp ngoặc trống như sau:

```
let result = orderFruit();
```

- Nếu hàm đòi hỏi phải có đối số, lời gọi hàm sẽ như sau:

```
let result = orderFruit(type, quantity);
```

- Bỏ qua cặp ngoặc đơn khi gọi hàm có thể gây ra những kết quả hoàn toàn khác so với ý định của bạn. Việc gọi hàm không có cặp ngoặc đơn sẽ trả lại tên của hàm, thay vì giá trị trả về của hàm. Hơn nữa, hàm

đó sẽ không được thực thi.


```
<html>
<head>
   <title>Order Fruit</title>
   <script type="text/javascript">
       function orderFruit() {
           let total = 0;
           // Call another function to place order
           return total;
       }
   </script>
</head>
<body>
<script type="text/javascript">
   let result = orderFruit();
   alert("The total is " + result);
</script>
</body>
</html>
```

- Khi chạy đoạn mã này gọi đến hàm orderFruit(). Hàm orderFruit() gọi một hàm khác để tạo đơn đặt hàm. Sau đó, tổng chi phí được tính và trả lại cho hàm gọi.

- Chỉnh sửa một chút – cụ thể là xóa cặp dấu ngoặc đơn khỏi lời gọi hàm sẽ làm thay đổi toàn bộ kết quả:

```
let result = orderFruit;
```

- Bất kể hàm có giá trị hoặc nhận đối số hay không, việc gọi hàm với cặp ngoặc đơn là thao tác quan trọng.

3. Hàm vô danh hoặc không được đặt tên

- Các hàm bạn đã thấy ở trên đều được định nghĩa theo chuẩn. Tuy nhiên, JavaScript không đòi hỏi hàm phải được định nghĩa theo cách đó. Ví dụ, với cách tạo hàm không tên hay còn gọi là hàm vô danh, hàm có thể được định nghĩa và gắn với biến như sau:

```
let divNums = function (firstNum, secondNum) {
   return firstNum / secondNum;
}
```

- Các hàm vô danh thường được dùng trong JavaScript hướng đối tượng và dùng làm hàm xử lý sự kiện.

4. Bao đóng – Closure

- Trong JavaScript, các hàm con bên trong có quyền truy cập đến các biến của hàm cha bên ngoài. Closure chỉ sự tồn tại của các biến bên ngoài ngữ cảnh thực thi thông thường của hàm. Closure thường được vô ý tạo ra và có thể gây rò rỉ bộ nhớ trong Windows Internet Explorer nếu không được xử lý đúng cách. Tuy nhiên, closure lại là một trong những tính năng mạnh trong javascript.

Xem xét ví dụ sau:

```
function myFunction() {
   let myNum = 10;
   function showNum() {
       alert(myNum);
   }
   return showNum();
}

let callFunc = myFunction();
myFunction();
```

- Trong ví dụ này, hàm showNum có quyền truy cập tới biến myNum được tạo ở hàm bên ngoài, hàm myFunction. Biến callFunc được tạo trong ngữ cảnh toàn cục và chứa một tham chiếu tới hàm showNum. Khi biến callFunc được tạo, nó lập tức có quyền truy cập tới biến myNum.

- Closure có thể được sử dụng để giả lập phương thức private bên trong đối tượng, và có thể áp dụng trong các hàm xử lý sự kiện.

Danh sách các bài học