Bài 9
Hàm

Module: BOOTCAMP PREPARATION

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài “Mảng”

Tóm tắt lại các phần đã học từ bài “Mảng”

Mục tiêu

- Trình bày được cú pháp khai báo hàm
- Trình bày được cú pháp gọi hàm
- Giải thích được tham số của hàm
- Giải thích cách sử dụng câu lệnh return trong hàm
- Trình bày được phạm vi của biến
- Khai báo và sử dụng được hàm không tham số
- Khai báo và sử dụng được hàm có tham số
- Khai báo và sử dụng được hàm có return

Thảo luận

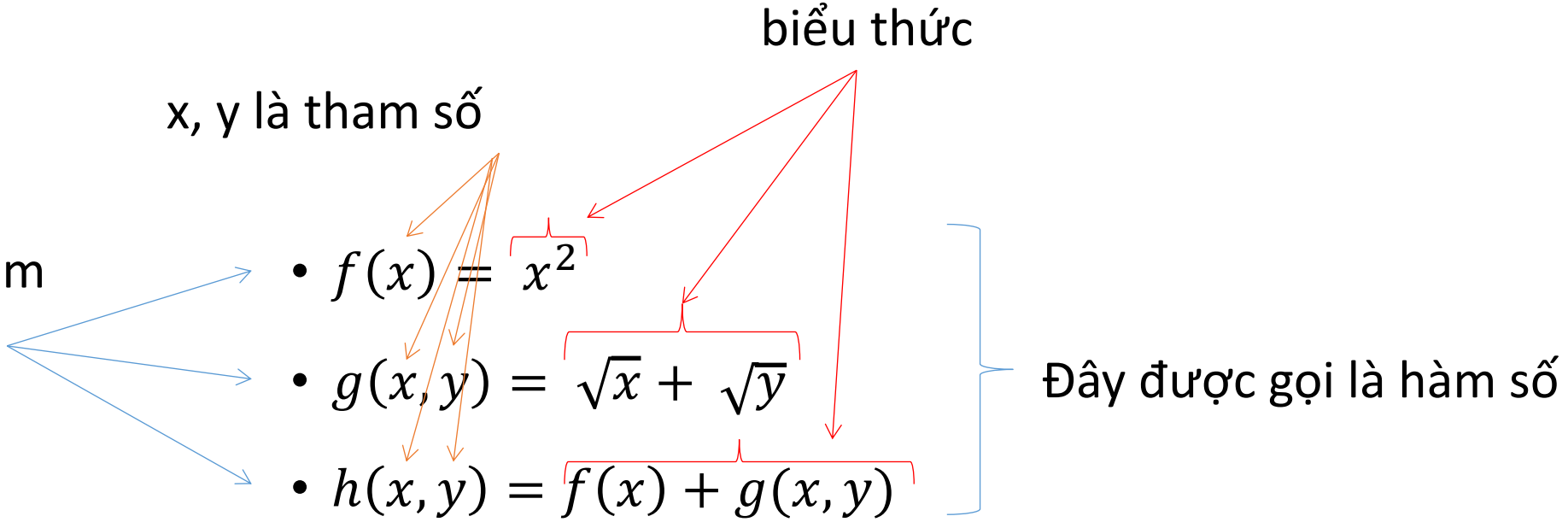
Hàm

Hàm

- Hàm (function) là một nhóm các câu lệnh thực hiện một nhiệm vụ nhất định
- *Hàm* là thuật ngữ được sử dụng phổ biến trong Lập trình hướng đối tượng. Trong nhiều trường hợp khác, các tên gọi được sử dụng là *phương thức* (method) và *thủ tục* (procedure)
- console.log(), Math.*pow*(), Math.*random*() là các hàm đã được định nghĩa sẵn cho chúng ta sử dụng

Lưu ý: Mặc dù tên gọi phương thức, hàm, procedure đôi khi có thể sử dụng thay thế cho nhau, nhưng giữa 3 khái niệm này có sự khác nhau.

Ví dụ về hàm số



- $f(5) = 5^2 = 25$
- $g(9,25) = \sqrt{9} + \sqrt{25} = 3 + 5 = 8$
- $h(9,\ 25) = 81 + 8 = 89$

Đánh giá hàm số

Các thành phần trong Hàm

- Nhiệm vụ của hàm
- Tên hàm
- Đầu vào của hàm
- Đầu ra của hàm

Ví dụ

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

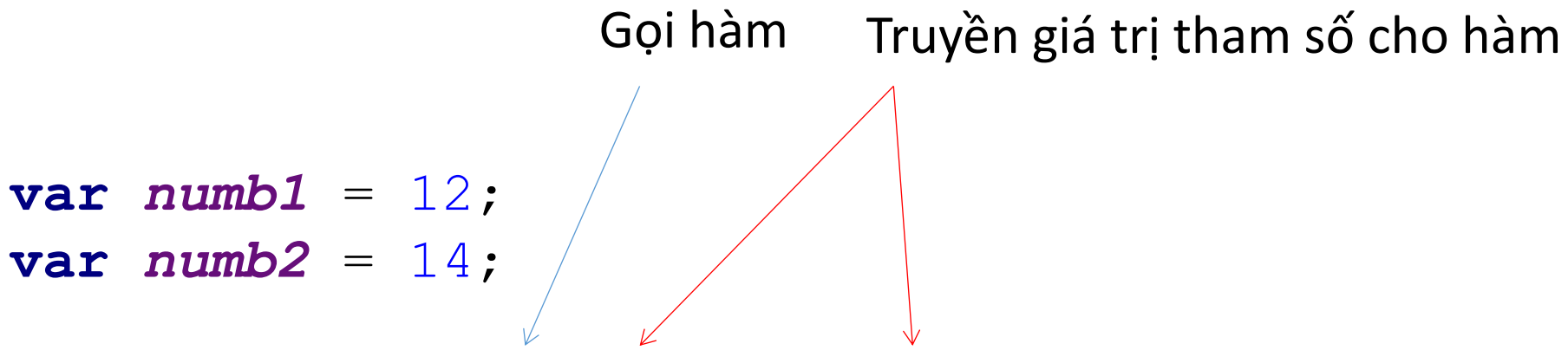
Nhiệm vụ: Tính tổng hai số

Tên hàm: sum

Đầu vào: 2 tham số

Đầu ra: tổng hai tham số

Sử dụng hàm




```
var numb1 = 12;
var numb2 = 14;

var result = sum(numb1 , numb2);
alert(result);
```

Hàm giúp tái sử dụng mã nguồn (1)

- Ví dụ, để tính tổng của các số từ 1 đến 10, tổng của các số từ 20 đến 37, tổng của các số từ 35 – 49:

```
var sum = 0;
for (var i = 1; i <= 10; i++)
sum += i;
console.log("Sum from 1 to 10 is " + sum);

sum = 0;
for (var i = 20; i <= 37; i++)
sum += i;
console.log("Sum from 20 to 37 is " + sum);

sum = 0;
for (var i = 35; i <= 49; i++)
sum += i;
console.log("Sum from 35 to 49 is " + sum);
```

Hàm giúp tái sử dụng mã nguồn (1)

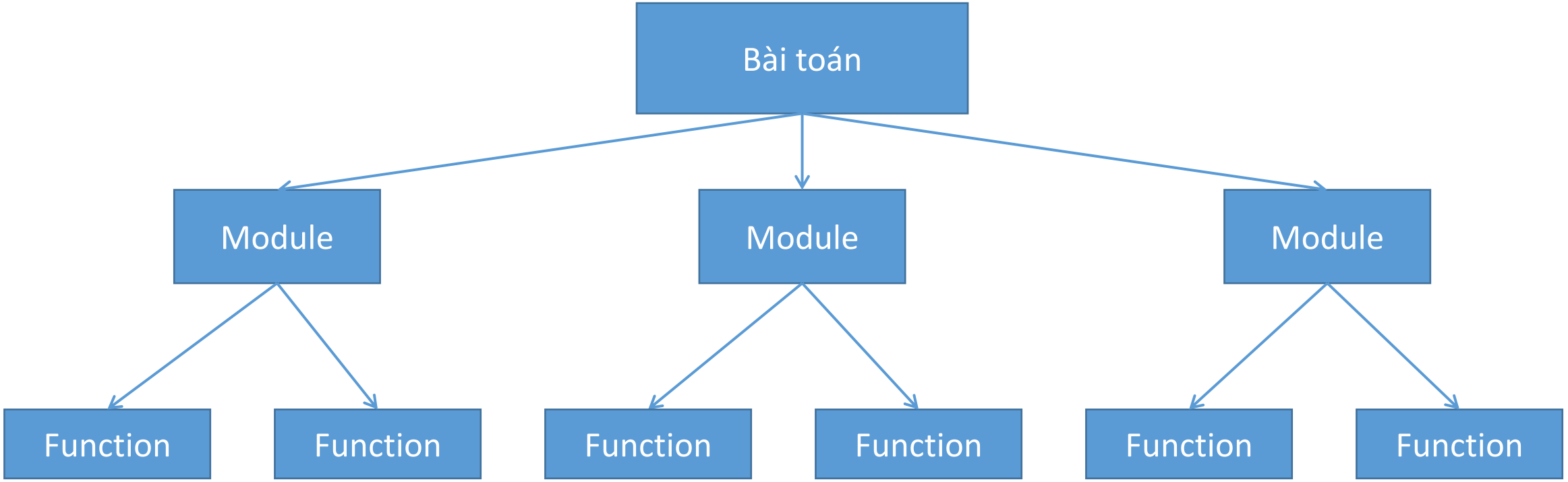
- Nếu sử dụng hàm, mã nguồn sẽ trở nên ngắn gọn hơn

```
function sum(start, end){
    var total = 0;
    for (var i = start; i <= end; i++){
        total += i;
    }
    return total;
}

function main() {
    console.log("Sum from 1 to 10 is " + sum(1,
10));
    console.log("Sum from 20 to 37 is " + sum(20,
37));
    console.log("Sum from 35 to 49 is " + sum(35,
49));
}
```

Ý nghĩa của việc sử dụng Hàm

- Thiết kế giải pháp theo hướng “chia để trị”



Khai báo hàm

- Cú pháp của hàm

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

- Các thành phần quan trọng của hàm
  - name: Tên hàm
  - parameter1, parameter2, parameter 3: Tham số
  - code to be executed: phần thân hàm (các lệnh thực thi hàm)

Tên của hàm

- Tên của hàm tuân thủ quy tắc đặt tên của Java
- Tên của hàm nên bắt đầu bằng một động từ (vì hàm thực hiện một hành động)

Ví dụ tên không tốt	Tên tốt	Lý do
myInterest	calculateInterest	Bắt đầu bằng động từ
FindMaxValue	findMaxValue	Viết thường chữ đầu tiên
get_payment	getPayment	Sử dụng Camel Case

Lưu ý: Đặt tên cho hàm là một việc rất quan trọng và cần được thực hiện với sự cẩn trọng để đảm bảo mã nguồn được tốt nhất.

Demo

Hàm

Thảo luận

Kiểu dữ liệu trả về

Kiểu dữ liệu trả về

- Một hàm có thể trả về một giá trị
- Ví dụ, hàm kiểm tra số chẵn có kiểu dữ liệu trả về là boolean:

```
function isEven( number){
    return number % 2 == 0;
}
```

Tham số (parameter) và đối số (argument)

- Tham số (còn được gọi đầy đủ là tham số hình thức – formal parameter) là các biến được khai báo trong phần header
- Khi gọi hàm thì giá trị của các biến này sẽ được truyền vào. Các giá trị này được gọi là tham số thực (actual parameter) hoặc đối số (argument)
- Ví dụ:

parameter

```
function isEven( number){
    return number % 2 == 0;
}
```

argument

```
isEven(5);
```

Hàm là chiếc hộp đen

- Có thể hình dung hàm như là những chiếc hộp đen có công dụng thực hiện các nhiệm vụ nhất định
- Tham số là đầu vào của chiếc hộp
- Giá trị trả về là đầu ra của chiếc hộp
- Người sử dụng hàm không quan tâm đến bên trong chiếc hộp, chỉ quan tâm đến đầu vào và đầu ra

Gọi hàm

- Gọi (*call* hoặc *invoke*) phương thức là cách để thực thi một hàm đã được định nghĩa trước đó
- Khi gọi hàm thì cần truyền đối số vào
- Ví dụ, gọi hàm không có giá trị trả về:

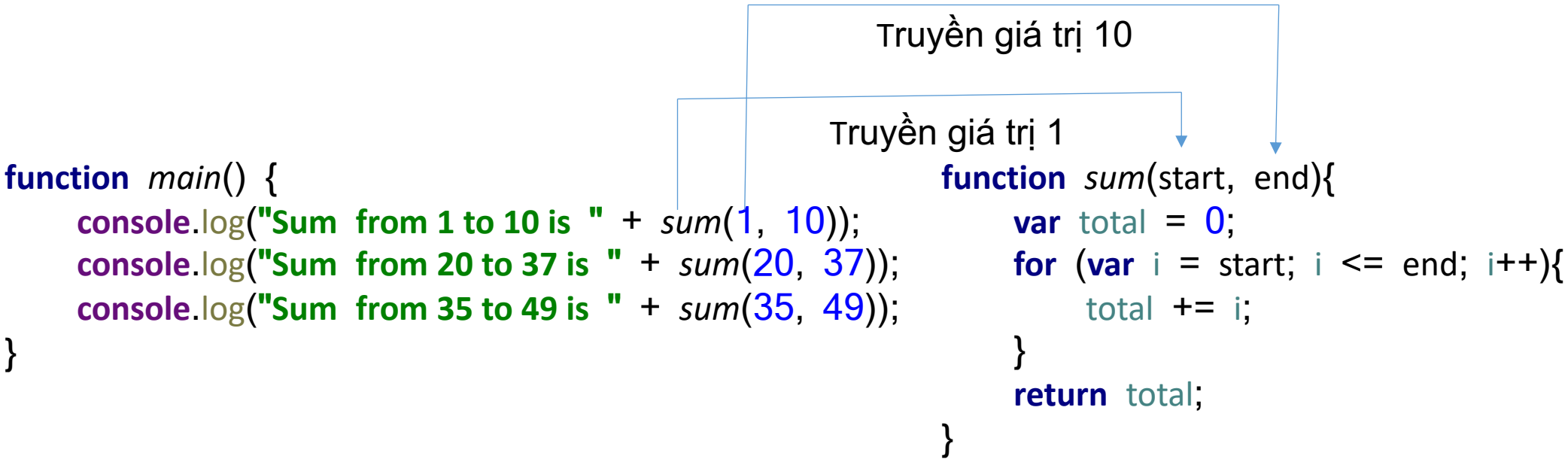
```
console.log("Welcome to Java!");
```

- Ví dụ, gọi hàm có giá trị trả về:

```
var larger = max(3, 4);
```

Luồng điều khiển (control flow)

- Khi gọi hàmức, luồng điều khiển được chuyển cho hàm đó



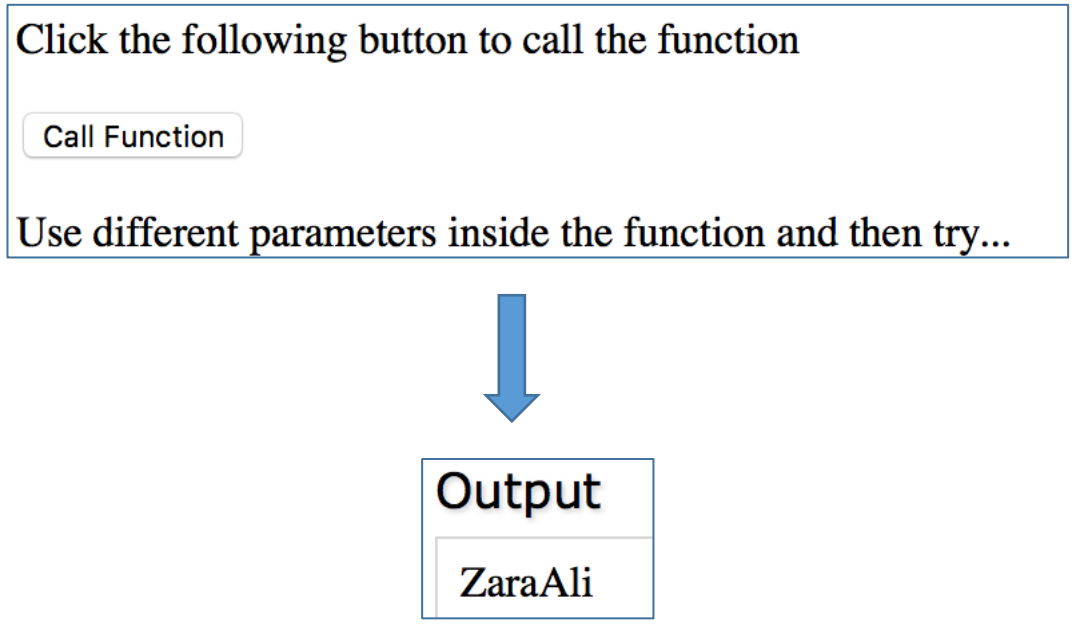
Giá trị trả về

- Giá trị trả về là kết quả sẽ được trả về tại vị trí hàm được gọi
- Mỗi một hàm chỉ có một giá trị trả về. Các giá trị này có thể là một biến, một mảng hay một đối tượng, danh sách đối tượng.
- Cú pháp sử dụng câu lệnh return

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

Giá trị trả về

```
<script type="text/javascript">
      function concatenate(first, last) {
         var full;
         full = first + last;
         return full;
      }
      function secondFunction() {
         var result;
         result = concatenate('Zara', 'Ali');
         document.write (result );
      }
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
   <input type="button" onclick="secondFunction()" value="Call Function">
</form>
<p>Use different parameters inside the function and then try...</p>
```



Thảo luận

Truyền giá trị vào hàm

Truyền giá trị vào hàm

- Khi gọi hàm thì cần truyền giá trị vào nếu hàm đó có khai báo tham số
- Trật tự các giá trị truyền vào phải tương ứng với trật tự khai báo các tham số
- Ví dụ:

```
function printMultiple(message, n) {
    for (var i = 0; i < n; i++)
    console.log(message);
}
```

Gọi đúng: `printMultiple("Hello", 4);`

Gọi sai: `printMultiple(4, "Hello");`

Pass-by-value: Ví dụ 1

```
function increment(n) {
    n++;
    console.log("n inside the method is " + n);
}

function main() {
    var x = 1;
    console.log("Before the call, x is " + x);
    increment(x);
    console.log("After the call, x is " + x);
}

main();
```

```
Before the call, x is 1
n inside the method is 2
After the call, x is 1
```

Giá trị của biến x không thay đổi sau khi gọi phương thức increment

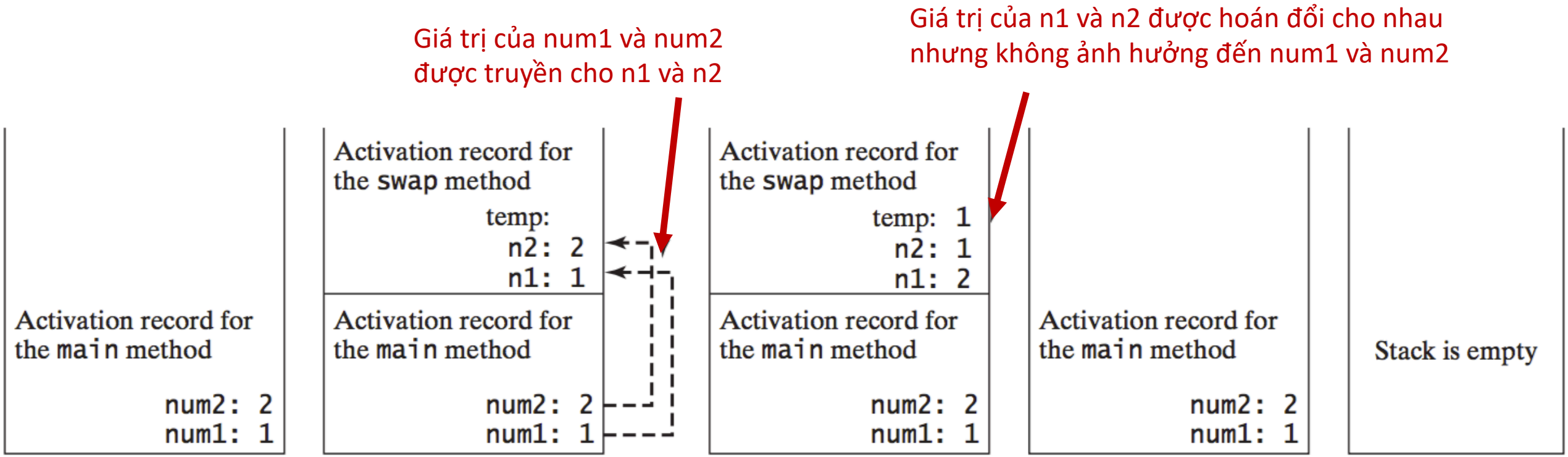
Pass-by-value: Ví dụ 2

```
function swap(n1, n2) {
    console.log("\tInside the swap method");
    console.log("\t\tBefore swapping, n1 is " + n1 + " and n2 is " + n2);
    var temp = n1;
    n1 = n2;
    n2 = temp;
    console.log("\t\tAfter swapping, n1 is " + n1 + " and n2 is " + n2);
}

function main() {
    var num1 = 1;
    var num2 = 2;
    console.log("Before invoking the swap method, num1 is " + num1 + " and num2 is " + num2);
    swap(num1, num2);
    console.log("After invoking the swap method, num1 is " + num1 + " and num2 is " + num2);
}
```

```
Before invoking the swap method, num1 is 1 and num2 is 2
  Inside the swap method
    Before swapping, n1 is 1 and n2 is 2
    After swapping, n1 is 2 and n2 is 1
After invoking the swap method, num1 is 1 and num2 is 2
```

Mô phỏng pass-by-value



1) Hàm main được gọi

2) Hàm swap được gọi → 3Hàm swap đang thực thi → 4) Hàm swap được thực thi xong → 5)Hàm main được thực thi xong

Thảo luận

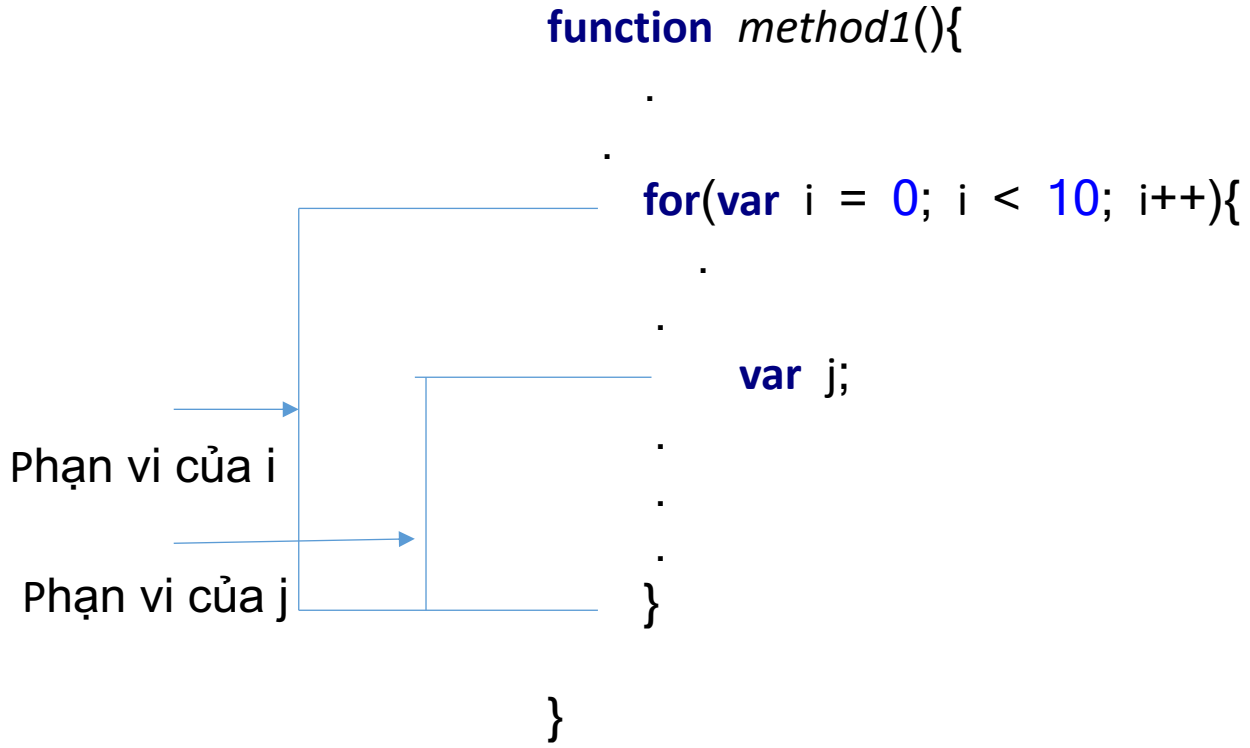
Phạm vi của biến

Phạm vi của biến

- Phạm vi (scope) của biến là các vị trí trong chương trình mà một biến có thể được sử dụng
- Một biến được khai báo trong một phương thức thì được gọi là *biến địa phương* (local variable)
- Phạm vi của biến địa phương bắt đầu từ vị trí nó được khai báo cho đến điểm kết thúc của khối lệnh chứa nó
- Một biến địa phương cần được khai báo và gán giá trị trước khi sử dụng
- Tham số của hàm cũng là các biến địa phương
- Phạm vi của các tham số là trong toàn bộ hàm đó

Phạm vi biến trong vòng lặp for

- Biến được khai báo trong phần header của vòng lặp thì có phạm vi trong toàn bộ vòng lặp
- Biến được khai báo trong phần thân của vòng lặp thì chỉ có phạm vi bên trong thân vòng lặp (tính từ vị trí được khai báo cho đến hết khối lệnh chứa nó)



Tóm tắt bài học

- Phương thức giúp tái sử dụng mã nguồn
- Các thành phần của một phương thức: modifier, kiểu dữ liệu trả về, tên phương thức, danh sách tham số
- Đặt tên cho phương thức là một thao tác rất quan trọng để đảm bảo clean code
- Tham số của phương thức (parameter) là các biến hình thức
- Đối số là các giá trị được truyền vào khi gọi phương thức
- Câu lệnh return được sử dụng để trả về giá trị của một phương thức
- Phạm vi của biến là các vị trí trong chương trình mà một biến có thể được sử dụng

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: *Lập trình hướng đối tượng*