

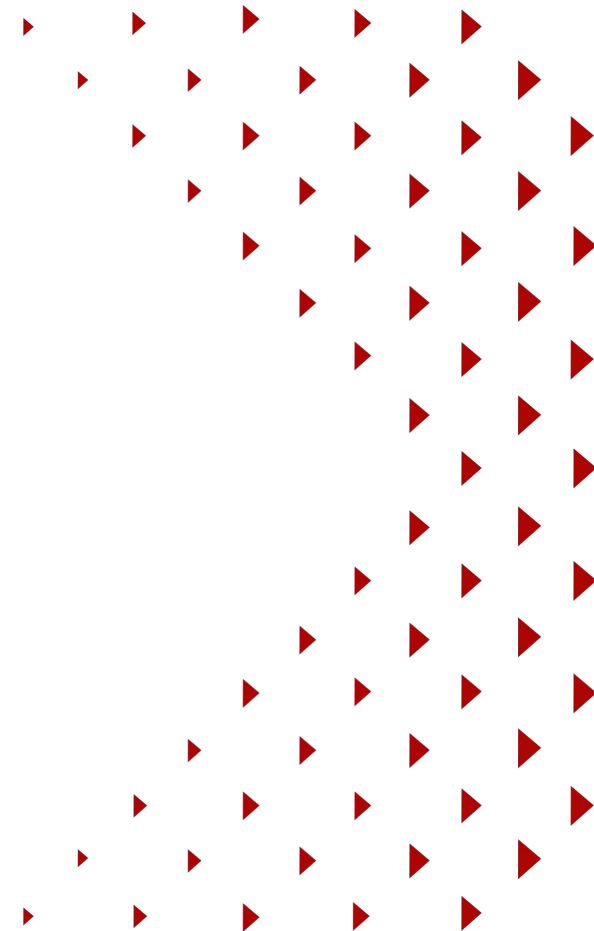


SESSION 10:

FUNCTION

Module 1: Web Front-end Fundamental

Version: 2.0



1. Tổng quan function

2. Các loại function

TỔNG QUAN FUNCTION - 1

• Định nghĩa

- Hàm trong javascript là một **chương trình con** bao gồm một **tập hợp các câu lệnh** thực hiện **một tác vụ** cụ thể
- Có thể tái sử dụng và bảo trì tốt hơn giúp nâng cao hiệu suất công việc

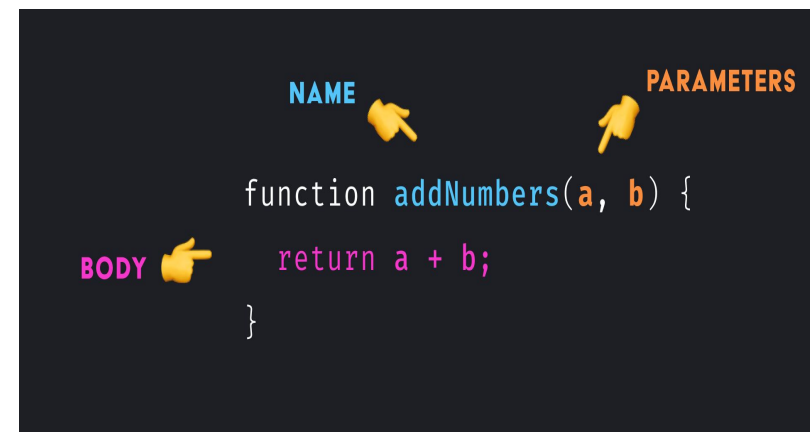
• Khai báo hàm

```
function functionName(param1,param2,...){
    Statements;
}
```

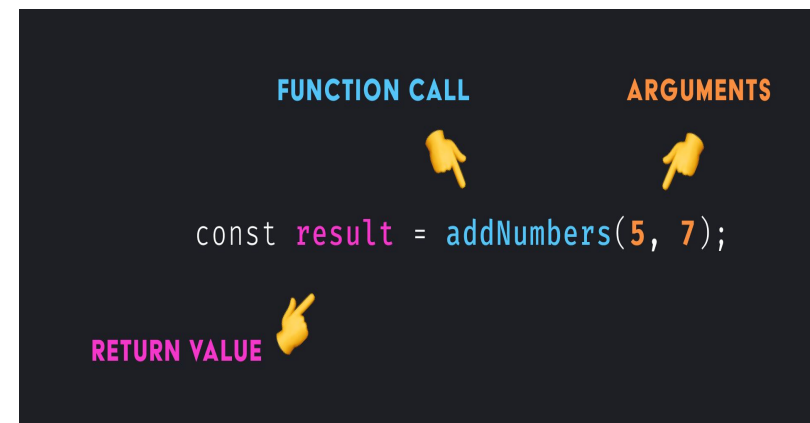
○ Trong đó:

- **function**: từ khóa khai báo hàm
- **functionName**: tên hàm, quy tắc đặt tên giống đặt tên biến (nên là động từ)
- **param1, param2...**: các tham số của hàm (0...N tham số)
- **Statements**: tập hợp các câu lệnh nằm trong thân hàm được bao quanh bởi {}
- Hàm được hoisting

© 2022 By Rikkei Academy - Rikkei Education - All rights reserved.



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



```
const result = addNumbers(5, 7);
```

TỔNG QUAN FUNCTION - 2

- **Tham số và đối số**

- **Tham số**

- Biến cục bộ của hàm được khai báo khi khai báo hàm
 - Hàm có thể có 0, 1 hay nhiều tham số

- **Đối số**

- Giá trị truyền vào khi gọi hàm
 - Tham trị
 - Truyền giá trị có kiểu dữ liệu là primitive
 - Sao chép (copy) giá trị của biến truyền vào sang tham số của hàm)
 - Tham chiếu
 - Truyền giá trị có kiểu dữ liệu là non_primitive như Array, Object, Function.
 - Truyền địa chỉ ô nhớ vào tham số của hàm

- **Gọi function**

- Gọi tên và truyền các đối số tương ứng với tham số của hàm
 - Hàm kết thúc khi gặp **return**

```
//Khai báo hàm không có tham số
function wellcome() {
    console.log("Wellcome to Rikkei Academy");
}
//Khai báo hàm có một tham số
function helloUser(userName) {
    console.log("Hello " + userName);
}
//Khai báo hàm có nhiều tham số
//Trả về một giá trị cụ thể
function addNumber(number1, number2) {
    let sum = num1 + num2;
    return sum;
}
//Gọi hàm
wellcome();
helloUser("Nguyễn Văn A");
let sumNumber = addNumber(3, 5);
```

CÁC LOẠI FUNCTION - 1

• Function Expression – biểu thức hàm

- Function được định nghĩa và lưu trữ trong một biến như kiểu giá trị
- Cú pháp

```
var | let | const variableName = function(parameters){
    Statements;
}
```
- Gọi hàm như thông thường

```
// function expression
let fnc_expr = function (name) {
    console.log("Wellcome to " + name);
};
//call function expression
fnc_expr("Rikkei Academy");
//-->output: Wellcome to Rikkei Academy
```

• Immediately invoked function expression (IIFE)

- Là Function Expression được thực thi ngay sau khi định nghĩa
- Cú pháp

```
(function(parameters){
    Statements;})(arguments)
```
- Sử dụng khi xử lý logic mà chỉ chạy đúng 1 lần

```
//immediately invoked function expression
(function (firstName, lastName) {
    console.log("Hello " + firstName + " " + lastName);
})("Rikkei", "Academy");
//-->output: Hello Rikkei Academy
```

CÁC LOẠI FUNCTION - 2

- **Arrow function**

- Cách viết function mới ra mắt trong ES6
- Giúp tiết kiệm thời gian code cũng như đơn giản hóa phạm vi function
- Không phụ thuộc vào từ khóa this
- Cú pháp

```
var | let | const functionName = (parameters) =>{  
    Statements;  
}
```

```
//Arrow Function  
var multiply1 = (number1, number2) => {  
    return number1 * number2;  
};  
//Arrow Function viết rút gọn  
var multiply2 = (number1, number2) => number1 * number2;  
//Call Arrow Function  
multiply1(5, 8); //-->output: 40
```

CÁC LOẠI FUNCTION - 3

- Function có nhiều tham số chưa xác định (Rest parameter – ES6)

```
const sum = function () {  
  let result = 0;  
  for (let i = 0; i < arguments.length; i++) {  
    result += arguments[i];  
  }  
  return result;  
};  
sum(1); //-->output:1  
sum(1, 2); //-->output:3  
sum(1, 2, 3); //-->output:6
```

```
const sum = function (...restParams) {  
  let result = 0;  
  for (let i = 0; i < restParams.length; i++) {  
    result += restParams[i];  
  }  
  return result;  
};  
sum(1); //-->output:1  
sum(1, 2); //-->output:3  
sum(1, 2, 3); //-->output:6
```


CÁC LOẠI FUNCTION - 4

- **Anonymous function – Hàm ẩn danh**

- Hàm ẩn danh không có tên
- Cú pháp:

```
function(){
    Statements;
}
```

- **Closures**

- Cho phép lồng các function trong nhau
- Các function con được định nghĩa bên trong function cha chứa nó
- Function con truy cập được các biến của function cha
- Function cha không thể truy cập biến của function con
- Cú pháp:

```
function father_func_name(parameters){
    function children_func_name(parameters){
```

```
// Anonymous function
setTimeout(function () {
    console.log("1s đã trôi qua");
}, 1000);
```

```
function autoGeneratorNumber() {
    let num = 0;
    function checkNumber() {
        console.log(num);
    }
    num++;
    return checkNumber;
}

var number = autoGeneratorNumber();
number();
```


CÁC LOẠI FUNCTION - 5

- **Callback function**

- Truyền đối số là một hàm
- Được sử dụng:
 - Xử lý đồng bộ trong javascript
 - Xử lý event
- Cú pháp:

```
function func_A(parameters){  
    Statements;  
}  
function func_B(parameters,callback){  
    Statements;  
    callback(arguments);  
}  
func_B(arguments,func_A);
```

```
function result(values) {  
    console.log(values);  
}  
function getSum(a, b, callback) {  
    let total = a + b;  
    callback(total);  
}  
getSum(5, 10, result); //-->Output:15
```



KẾT THÚC

HỌC VIỆN ĐÀO TẠO LẬP TRÌNH CHẤT LƯỢNG NHẬT BẢN