Functions

Ba Nguyễn

Functions

Functions (***hàm - phương thức***) được sử dụng để *đóng gói* các *khối mã* - ***code block***, cho phép tái sử dụng code, giảm thiểu code, giảm lỗi, ... Tên hàm cũng sử dụng cú pháp ***camelCase***

```
// function declaration
function funcName() {
    // code
}

// run
funcName();
```

```
// function expression
let funcName = function () {
    // code
}

// run
funcName();
```

Functions

```
function isOdd() {
    let n = Number(prompt("Nhập một số"));
    if (isNaN(n)) {
        console.log("Vui lòng nhập một số hợp lệ!");
    } else {
        if (n % 2 == 1) {
            console.log(n + " là số lẻ!");
        } else {
            console.log(n + " không phải là số lẻ!");
        }
    }
}
```

Exercise

1. Viết hàm `maxOfThree(a, b, c)` nhận vào 3 số, in ra số lớn nhất
2. Viết hàm `max()` nhập lượng số tùy ý in ra số lớn nhất
3. Viết hàm `isPrime(n)` nhận vào 1 số, kiểm tra và in ra số đó có phải số nguyên tố hay không

Local vs Global variables

Một biến được khai báo bên trong hàm ***chỉ tồn tại bên trong hàm*** đó, biến đó được gọi là ***local variable***

```
function local() {
    let myName = "Ba";
    console.log(myName);
}
local(); // "Ba"
console.log(myName); // ✗ error
```

Một biến khai báo bên ngoài tất cả các hàm (*khối code*) được gọi là ***global variable***, hàm có thể truy cập tới biến được khai báo bên ngoài

```
let myName = "Ba";
function global() {
    console.log(myName);
}
global(); // "Ba"
console.log(myName); // "Ba"
```

💡 Hàm chỉ tham chiếu tới biến bên ngoài, ***nếu*** trong hàm không có biến ***local*** nào trùng tên, ngoài ra, ***không nên*** tham chiếu trực tiếp tới biến ***global***, thay vào đó, sử dụng ***parameters***

Function Parameters

Tham số (*parameters*) cho phép truyền giá trị từ bên ngoài vào hàm, *tham số* được khai báo cùng với hàm. Khi một hàm được gọi với ***đối số*** (*arguments*), giá trị của đối số sẽ được gán cho tham số tương ứng

```
function isOdd(number) {
    // khai báo hàm với tham số number
    if (number % 2 == 0) {
        console.log(number + " không phải số lẻ");
    } else {
        console.log(number + " là số lẻ");
    }
}

isOdd(1); // number = 1
isOdd(2); // number = 2
```

Default Parameters

Khi hàm được gọi mà không truyền giá trị, tham số sẽ nhận giá trị `undefined`. Có chỉ định một giá trị mặc định cho tham số sẽ được sử dụng trong trường hợp đó:

```
// Cú pháp mới cho phép gán giá trị
// mặc định trực tiếp trong khai báo hàm
function hi(name, es6 = "New syntax") {
    // Cú pháp gán giá trị mặc định kiểu cũ
    name = name || "stranger";
    console.log("Hello " + name);
}

hi(); // "Hello stranger"
hi("Ba"); // "Hello Ba"
```

Return

Một hàm luôn trả về một giá trị nào đó khi nó được gọi, mặc định là `undefined`. Để chỉ định một giá trị trả về từ hàm, sử dụng câu lệnh `return`:

```
function hi(name) {
    console.log(name);
}

hi("Ba");
// log "Ba"
// then return undefined
```

```
function isOdd(number) {
    return number % 2 == 1; // return result
}

let check = isOdd(5); // true

if (isOdd(5)) {
    // code
}
```

💡 Câu lệnh `return` tương tự `break`, nó *dừng* hàm, bỏ qua mọi câu lệnh bên dưới

Exercise

1. Viết hàm `printPrime(n)` in ra các số nguyên tố trong khoảng từ `0 → n`
2. Viết hàm `convertTemperature(temp, to)` chuyển đổi nhiệt độ từ *Celsius* sang *Farenheit* hoặc *Kevin*, mặc định sẽ chuyển từ *Celsius* sang *Kevin* và trả về kết quả
3. Viết hàm `nearest(a, b)` nhận vào 2 số, kiểm tra số nào gần với `100` nhất và trả về kết quả

Functions

Một số lưu ý khi sử dụng hàm

- Một hàm chỉ nên thực hiện *một công việc duy nhất*, nếu một hàm phức tạp với nhiều công việc nhỏ, hãy *tách ra thành các hàm nhỏ hơn*
- Tên hàm nên *mô tả chính xác chức năng* của nó, ví dụ: `isOddNumber()`, `calcAverage()`, ...

Lưu ý khi dùng `return` với toán tử `?`

```
condition
    ? return value1
    : return value2; // ✗ error

return condition ? value : value; // 👌 oke
```

💡 Có thể sử dụng `return;` để ngắt hàm mà không trả về giá trị nào (`undefined`)

Function Call vs Reference

Hàm cũng chỉ là một giá trị, nó có thể được gán cho một biến, sao chép, ...

```
function func() {
    return "Oh wow";
}

// Function Reference
console.log(func); // function () { ... }
let x = func; // Assign
x(); // Function Call
```

Callback

Nếu một hàm được truyền vào hàm khác như một đối số, hàm đó được gọi là hàm ***callback***.

Lưu ý: Hàm được truyền dưới dạng ***giá trị hàm (reference)***

```
function double(n) {
    return n * 2;
}

function func(callback) {
    let result = callback(10); // double(10)
    return result;
}

func(double); // callback = double
// Result 20
```

Homework

1. Viết hàm cube(n) tính và trả về giá trị lập phương của một số n
2. Viết hàm sumFibo(n) tính và trả về tổng dãy số Fibonacci từ 0 → n
3. Viết hàm isPrime(n) kiểm tra ncó phải số Prime, kết quả trả về true hoặc false
4. Viết hàm isArmstrong(n) kiểm tra n có phải số Armstrong, kết quả trả về true hoặc false
5. Viết hàm isPerfetc(n) kiểm tra n có phải số Perfect, kết quả trả về true hoặc false
6. Viết hàm isPalindrome(n) kiểm tra n có phải số Palindrome, kết quả trả về true hoặc false
7. Viết hàm factorial(n) tính và trả về giai thừa của n
8. Viết hàm isExpo(a, b) kiểm tra số a có phải lũy thừa của b hay không, kết quả trả về true hoặc false
9. Viết hàm subByMultiAndSum(n) tính và trả về kết quả biểu thức hiệu giữa tích và tổng của các chữ số trong số n.
10. Viết hàm expo(x, y) tính và trả về kết lũy thừa bậc y của x, (y có thể âm)

Homework

1. Viết hàm random(a, b), trả về số ngẫu nhiên trong khoảng a - b
2. Viết hàm isTriangle(a, b, c) nhận vào 3 tham số là cạnh của tam giác, kiểm tra 3 cạnh có phải tam giác hợp lệ hay không?
3. Viết hàm guessNumber(), khởi tạo một số ngẫu nhiên trong khoảng 0 - 10, sau đó hiển thị bảng cho phép người dùng đoán giá trị. Nếu đoán đúng hiển thị "Â mây zing, gút chóp" và dừng hàm, nếu sai hiển thị "Không khớp" và cho phép nhập lại. Số lần nhập tối đa là 3 lần, hoặc người dùng có thể bấm cancel để dừng hàm.
4. Viết hàm nearest(a, b) kiểm tra và trả về số gần với 100 nhất. VD: nearest(90, 105) // 105, hoặc rearest(80, 90) // 90
5. Viết hàm sequense(a, b, c) kiểm tra xem a, b, c có có phải tăng dần (hoặc giảm dần) hay không.

Homework

1. Viết hàm countDecimal(number) trả về số chữ số trong phần thập phân của number
2. Viết hàm isAscending(number) kiểm tra dãy số của number có phải dãy tăng dần hay không?. VD: isAscending(123) // true
3. Viết hàm countEven(number) trả về số chữ số chẵn của number. VD: countEven(12345) // 2
4. Viết hàm find(number) trả về số n sao cho 1 + 2 + ... + n <= number. VD: find(7) // 3
5. Viết hàm sum(value1, unit1, value2, unit2) quy đổi về cùng đơn vị và trả về tổng giá trị value1 + value2 theo unit1, unit bao gồm km, m, cm, dm, mm. VD: sum(1, 'km', 100, 'm') // 1,1 km