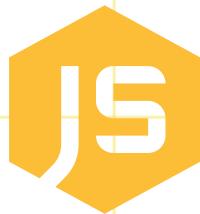


# JAVASCRIPT FUNCTION



# Lý do tại sao cần phải tách hàm

- **Tái sử dụng mã:** Hàm cho phép bạn viết mã một lần và sử dụng lại nhiều lần. Điều này không chỉ giúp tiết kiệm thời gian mà còn làm cho mã nguồn dễ bảo trì và quản lý hơn.
- **Giảm độ phức tạp của chương trình:** Hàm giúp phân chia một chương trình phức tạp thành các phần nhỏ hơn, dễ hiểu hơn. Điều này làm cho mã nguồn dễ đọc và dễ kiểm tra lỗi hơn.
- **Che giấu chi tiết thực thi:** Hàm cho phép bạn ẩn đi các chi tiết về cách một tác vụ cụ thể được thực hiện, điều này giúp người dùng của hàm chỉ cần quan tâm đến kết quả mà không cần biết chi tiết bên trong.



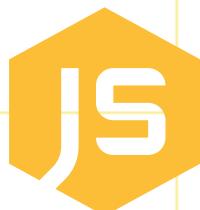
# Định nghĩa hàm

Trong JavaScript, một hàm được sử dụng để thực hiện một tác vụ cụ thể. Một hàm có thể nhận vào tham số, thực hiện các tính toán hoặc hành động, và trả về kết quả. Hàm có thể được gọi từ các phần khác của chương trình hoặc từ chính nó, và có thể trả về giá trị hoặc không.

**Ở Javascript chúng ta có 3 loại khai báo hàm chính:**

- Function Declarations (Khai báo hàm)
- Function Expressions (Biểu thức hàm)
- Arrow Functions (Hàm mũi tên)

# FUNCTION DECLARATIONS



# Function Declarations (Khai báo hàm)

function

tên hàm

(tham số truyền vào)

{

Các dòng lệnh được xử lí khi hàm thực thi

}

javascript

```
function greet(name) {  
    return `Hello, ${name}!`;  
}
```

Sao chép mã



# Hàm có tham số

**Tham số trong hàm đóng vai trò quan trọng trong lập trình bởi vì chúng giúp hàm trở nên linh hoạt và tái sử dụng được, ứng dụng chính của tham số là:**

- Tái sử dụng mã
- Động và linh hoạt
- Tùy biến cao

**Một ví dụ về cách tạo hàm tính điểm trung bình linh động tính toán cho từng sinh viên**

```
// Ví dụ tạo một hàm có tham số để tính điểm trung bình cho tất cả sinh viên
// tạo 2 tham số diemToan và diemVan để linh động hơn cho điểm của từng sinh viên
function tinhDiemTrungBinh(diemToan, diemVan) {
    let diemTrungBinh = (diemToan + diemVan) / 2;
    console.log(diemTrungBinh);
}
// tính điểm trung bình cho Lâm với điểm toán 5 và văn 7
tinhDiemTrungBinh(5, 7);
// tính điểm trung bình cho Sang với điểm toán 7 và văn 4
tinhDiemTrungBinh(7, 4);
```

**Lưu ý một số khái niệm:**

**Tham số (parameters)** là danh sách các tên tham số trong định nghĩa hàm

**Đối số (arguments)** là danh sách các tên tham số trong định nghĩa hàm



# Hàm có giá trị trả về

Trong lập trình, từ khóa `return` trong một hàm được sử dụng để kết thúc hàm và trả về một giá trị tới nơi gọi hàm. Việc sử dụng `return` mang lại nhiều lợi ích và tính năng quan trọng:

- Truyền Dữ liệu giữa các Hàm
- Kiểm Soát Luồng Chương Trình
- Tái sử dụng Mã

**Một ví dụ về cách tạo hàm tính tiền giảm giá cho món ăn**

```
// Ví dụ tạo một hàm tính toán giảm giá cho món ăn
// Người dùng sẽ truyền vào 2 tham số giaTien và giamGia
function tinhTienGiamGia(giaTien, giamGia) {
    let tienGiamGia = (giaTien * giamGia) / 100;
    return tienGiamGia;
}
// tính giảm giá cho món miXao có giá 100k và giảm giá 10%
let tienGiamGiaMiXao = tinhTienGiamGia(100, 10);
console.log(tienGiamGiaMiXao); // số tiền giảm giá
// tính giảm giá cho món phoBo có giá 50k và giảm giá 20%
let tienGiamGiaPhoBo = tinhTienGiamGia(50, 20);
console.log(tienGiamGiaPhoBo); // số tiền giảm giá
```

# FUNCTION EXPRESSIONS



# Function Expressions

let

tên biến

=

function

(tham số truyền vào)

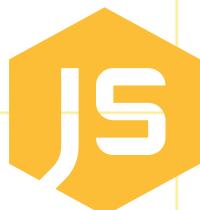
{

Các dòng lệnh được xử lí khi hàm thực thi

}

```
javascript  
  
const add = function(a, b) {  
    return a + b;  
};  
  
// Gọi hàm  
const result = add(5, 3);  
console.log(result); // In ra 8
```

Sao chép mã



# Một số ví dụ về function expression

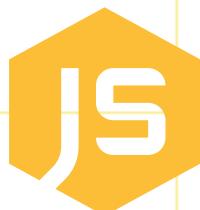
Ví dụ về tạo một hàm tính toán tổng tiền cho món ăn bằng function expression

```
// Ví dụ tạo một hàm tính toán tổng tiền cho món ăn
// Người dùng sẽ truyền vào 2 tham số giaTien và soLuong
let tinhTongTien = function (giaTien, soLuong) {
    let tongTien = giaTien * soLuong;
    return tongTien;
};

// tính tổng tiền cho 3 phần mì xào có giá 100k
let tongTienMiXao = tinhTongTien(100, 3);
console.log(tongTienMiXao); // số tiền phải trả

// tính tổng tiền cho 5 phần phở bò có giá 50k
let tongTienPhoBo = tinhTongTien(50, 5);
console.log(tongTienPhoBo); // số tiền phải trả
```

# ARROW FUNCTION



# Arrow function (Hàm mũi tên)

let

tên biến

=

(tham số truyền vào)

=>

{

Các dòng lệnh được xử lí khi hàm thực thi

}

```
javascript                                     Sao chép mã

const evaluateStudent = (score) => {
    if (score > 8) {
        return "Sinh viên giỏi";
    } else {
        return "Sinh viên khá";
    }
};
```



# Một số khai báo đặc biệt của arrow function

## Arrow Function với Một Tham Số

- Khi một arrow function chỉ nhận một tham số, bạn có thể bỏ qua cặp dấu ngoặc đơn xung quanh tham số đó. Điều này làm cho cú pháp trở nên ngắn gọn hơn.

```
// Ví dụ hiển thị câu chào cho sinh viên
// Người dùng sẽ truyền vào 1 tham số tenSinhVien
// Arrow function cho phép viết ngắn gọn hơn loại bỏ dấu () khi chỉ có 1 tham số
let chaoSinhVien = tenSinhVien => {
  console.log(`Chào ${tenSinhVien}`);
};

// chào sinh viên Lâm
chaoSinhVien("Lâm");
```

## Arrow Function khi chỉ có 1 lệnh return

- Khi body của arrow function chỉ chứa một biểu thức trả về, bạn có thể bỏ qua cặp dấu ngoặc nhọn {} và từ khóa return. Giá trị của biểu thức đó sẽ tự động được trả về.

```
// Ví dụ về hàm arrow function có một giá trị trả về
// Người dùng sẽ truyền vào 2 tham số a và b
// Khi sử dụng arrow function, nếu chỉ có một dòng lệnh thì không cần viết return
let tinhTong = (a, b) => a + b;

// tính tổng 2 số 5 và 7
let tong = tinhTong(5, 7);
```

# DEFAULT PARAMETER VALUES



# Default Parameter Values (ES6)

Kỹ thuật default parameter values cho phép bạn đặt giá trị mặc định cho tham số của hàm nếu không có giá trị nào được truyền vào khi hàm được gọi. Điều này làm cho hàm của bạn linh hoạt hơn, dễ dàng xử lý các trường hợp mà tham số không được cung cấp, và giúp giảm thiểu lỗi do thiếu dữ liệu đầu vào.

```
let tên biến = (tham số truyền vào= giá trị mặc định) =>
```

```
{
```

Các dòng lệnh được xử lí khi hàm thực thi

```
}
```



# Ví dụ về default parameter values

```
// Ví dụ về tạo một hàm giúp tính toán số tiền món ăn sau khi giảm giá
// Người dùng sẽ truyền vào 3 tham số giaTien, giamGia và tenMon
// Người dùng yêu cầu nếu không truyền giảm giá thì giảm giá mặc định là 10
let tinhTienSauGiamGia = (giaTien, tenMon, giamGia = 10) => {
  let tienGiamGia = (giaTien * giamGia) / 100;
  let tienSauGiamGia = giaTien - tienGiamGia;
  console.log(`Món ${tenMon} sau khi giảm giá còn ${tienSauGiamGia}k`);
};

// tính tiền món mì xào có giá 100k và giảm giá 20%
tinhTienSauGiamGia(100, "mì xào", 20);
// tính tiền món phở bò có giá 50k và tự động giảm giá 10%
tinhTienSauGiamGia(50, "phở bò");
```

## Lợi ích của Default Parameters

- **Giảm bớt lỗi:** Khi các tham số không được cung cấp, hàm sẽ tự động sử dụng giá trị mặc định, giúp tránh các lỗi có thể xảy ra do giá trị undefined.
- **Tăng tính linh hoạt của hàm:** Bạn có thể thiết kế hàm để nó hoạt động tốt ngay cả khi một số thông tin không được cung cấp.
- **Làm cho chức năng của hàm rõ ràng hơn:** Khai báo giá trị mặc định ngay trong định nghĩa hàm giúp làm rõ mục đích và cách sử dụng của hàm.

# CALLBACK FUNCTION



# Callback function

Trong JavaScript, một callback function là một hàm được truyền vào hàm khác như một đối số, sau đó được hàm đó gọi lại (execute) bên trong bản thân nó. Callbacks là một phần thiết yếu trong lập trình JavaScript, đặc biệt là trong việc xử lý các hoạt động bất đồng bộ.

**Ứng dụng của callback function giúp chúng ta thực hiện xử lí kết quả từ một hàm khác với các hành động khác nhau**

```
javascript                                     Sao chép mã

function xepLoaiSinhVien(diem, callback) {
    let xepLoai;
    if (diem >= 8) {
        xepLoai = 'Giỏi';
    } else if (diem >= 5) {
        xepLoai = 'Khá';
    } else {
        xepLoai = 'Trung Bình';
    }
    callback(xepLoai);
}

// Hàm callback để in xếp loại
function inXepLoai(xepLoai) {
    console.log(`Sinh viên xếp loại: ${xepLoai}`);
}

// Gọi hàm xepLoaiSinhVien
xepLoaiSinhVien(8.5, inXepLoai); // Sinh viên xếp loại: Giỏi
```

# BÀI TẬP LUYỆN TẬP VỀ HÀM



# Dự án tính tiền Grab

## Dự án Tính tiền Grab

- Cho người dùng chọn 1 trong 3 loại Grab :
  - 1 : GrabCar
  - 2 : Grab SUV
  - 3 : GrabBlack
- Cho người dùng nhập vào số KM đi được và Thời gian chờ. Biểu phí như bên dưới

BÀNG GIÁ CƯỚC GRAB			
THEO KM	GRAB CAR (Đ)	GRAB SUV (Đ)	GRAB BLACK (Đ)
KM ĐẦU TIÊN	8000	9000	10000
Từ 1 đến 19	7500	8500	9500
Từ 19 trở lên	7000	8000	9000
Thời gian chờ trên 3 phút (moi 3 phut)	2000	3000	3500

Yêu cầu:

- 1) Tính tổng tiền đi được
- 2) In hóa đơn chi tiết như biểu mẫu

CHI TIẾT HÓA ĐƠN			
CHI TIẾT	SỬ DỤNG	ĐƠN GIÁ( 1000đ)	THÀNH TIỀN(1000đ)
KM ĐẦU TIÊN	0.8	(tùy theo loại grab)	
Từ .... đến	....		
Từ.....đến	.....		
Thời gian chờ	.....		
TỔNG TIỀN:.....			

[Link chứa layout bài tập](#)