

# GỌI HÀM TRONG HÀM: LƯỒNG DỮ LIỆU

```
const cutPieces = function (fruit) {  
  return fruit * 4;  
};  
  
const fruitProcessor = function (apples, oranges) {  
  const applePieces = cutPieces(apples);  
  const orangePieces = cutPieces(oranges);  
  
  const juice = `Juice with ${applePieces} pieces of  
apple and ${orangePieces} pieces of orange.`;  
  return juice;  
};  
  
console.log(fruitProcessor(2, 3));
```

The diagram illustrates the data flow in the provided JavaScript code. Red arrows represent the return values of the `cutPieces` function, which are passed as arguments to the `fruitProcessor` function. A yellow arrow represents the final return value of the `fruitProcessor` function, which is passed to the `console.log` function.

- The `cutPieces` function is called with `apples = 2` and `oranges = 3`.
- The `cutPieces` function returns `2 * 4 = 8` for `applePieces` and `3 * 4 = 12` for `orangePieces`.
- The `fruitProcessor` function receives these values and returns a string: `Juice with 8 pieces of apple and 12 pieces of orange.`
- The `console.log` function receives the string and logs it to the console.

# ÔN TẬP VỀ HÀM: 3 LOẠI HÀM KHÁC NHAU

## 👉 Khai báo hàm

Hàm có thể được sử dụng trước khi nó được khai báo

```
function calcAge(birthYear) {  
  return 2037 - birthYear;  
}
```

## 👉 Biểu thức hàm

Cơ bản là một hàm  
*giá trị* được lưu trữ trong một biến

```
const calcAge = function (birthYear) {  
  return 2037 - birthYear;  
};
```

## 👉 Hàm mũi tên

Phù hợp cho các hàm chỉ có  
1 dòng ngắn gọn. Không  
có từ khóa this

```
const calcAge = birthYear => 2037 - birthYear;
```



Có 3 cách viết hàm khác nhau, nhưng tất cả đều hoạt động một cách tương tự: nhận dữ liệu **đầu vào**, **biến đổi** dữ liệu, sau đó **xuất** dữ liệu.

# ÔN TẬP VỀ HÀM: PHÂN TÍCH HÀM



