









# Chuyên Sâu về **this** trong JavaScript

 **this** có ngữ cảnh riêng trong những loại hàm nào?

Loại hàm	<b>this</b> có ngữ cảnh riêng không?	Cách xác định <b>this</b>
Hàm thông thường ( <b>function</b> )	 Có	<b>this</b> phụ thuộc vào cách gọi hàm (object gọi nó, strict mode, global, v.v.).
Hàm ẩn danh ( <b>function expression</b> )	 Có	Tương tự như hàm thông thường, <b>this</b> cũng phụ thuộc vào cách gọi.
Hàm constructor ( <b>new Function()</b> )	 Có	<b>this</b> tham chiếu đến instance mới được tạo.
Hàm trong sự kiện DOM ( <b>onclick = function() {...}</b> )	 Có	<b>this</b> tham chiếu đến phần tử HTML gọi sự kiện.
Hàm trong setTimeout/setInterval ( <b>setTimeout(function() {...}, 1000)</b> )	 Có	<b>this</b> mặc định trỏ đến <b>window</b> hoặc <b>undefined</b> (strict mode).
Hàm trong method của Object ( <b>obj.method = function() {...}</b> )	 Có	<b>this</b> tham chiếu đến object chứa method.
Hàm trong class (method của class)	 Có	<b>this</b> tham chiếu đến instance của class.
Hàm mũi tên ( <b>=&gt;</b> )	 Không	<b>this</b> kế thừa từ phạm vi (scope) chứa nó.

## Ví dụ minh họa

**1** Hàm ẩn danh có **this** riêng

```
const obj = {
  name: "Alice",
  sayHello: function() { // Hàm ẩn danh
    console.log(this.name);
  }
};
obj.sayHello(); // Alice
```

**Giải thích:** **this** trong **sayHello()** tham chiếu đến **obj**.

## 2 Hàm mũi tên không có `this` riêng

```
const obj = {
  name: "Alice",
  sayHello: () => {
    console.log(this.name);
  }
};
obj.sayHello(); // undefined
```

### Giải thích:

- Hàm mũi tên **không có ngữ cảnh riêng**.
- `this` kế thừa từ phạm vi bên ngoài (global scope), dẫn đến `this.name` là `undefined`.

---

## 3 Hàm trong `setTimeout()`

```
const obj = {
  name: "Alice",
  sayHello: function() {
    setTimeout(function() {
      console.log(this.name); // undefined
    }, 1000);
  }
};
obj.sayHello();
```

### Giải thích:

- `this` trong `setTimeout()` không trở về `obj`, mà trở đến `window` hoặc `undefined` (strict mode).
- Để giữ `this`, có thể dùng `.bind(this)`, hoặc arrow function.

### Cách sửa bằng arrow function:

```
const obj = {
  name: "Alice",
  sayHello: function() {
    setTimeout(() => {
      console.log(this.name); // Alice
    }, 1000);
  }
};
obj.sayHello();
```

### Giải thích:

- Hàm mũi tên **không có `this` riêng**, kế thừa `this` từ `sayHello()`.
  - Kết quả: `this.name` vẫn là `"Alice"`.
- 

## Kết luận

- Chỉ có hàm mũi tên (`=>`) **không có `this` riêng**, mà kế thừa từ phạm vi cha.
- Các loại hàm khác (**`function expression`, `function declaration`, `setTimeout()`, method trong object, v.v.**) đều có **`this` riêng**, phụ thuộc vào cách gọi.
- Muốn đảm bảo **`this`** không bị thay đổi, nên dùng **arrow function** hoặc **`.bind(this)`**. 