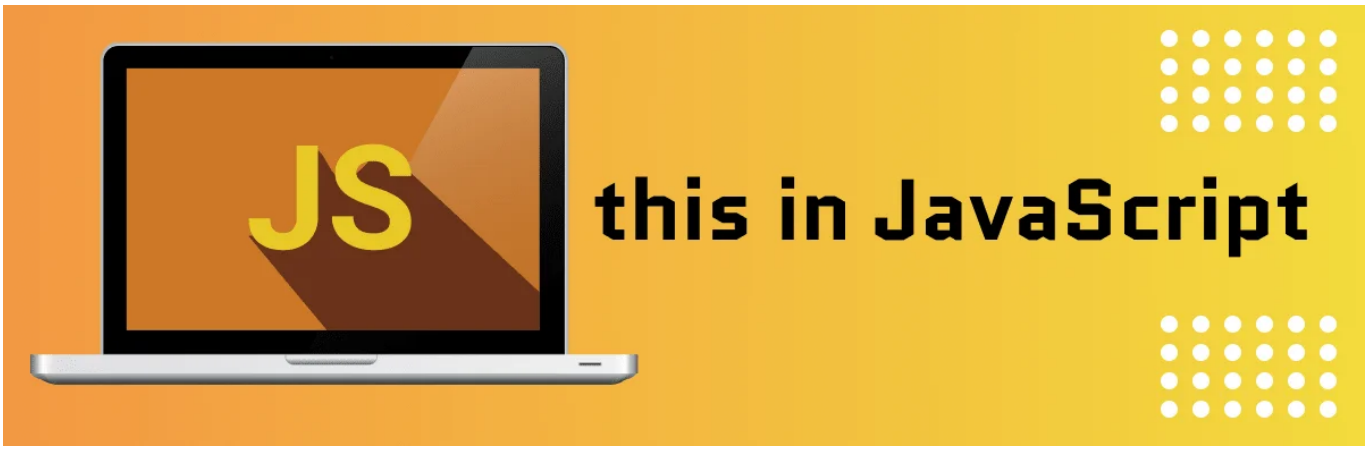
This



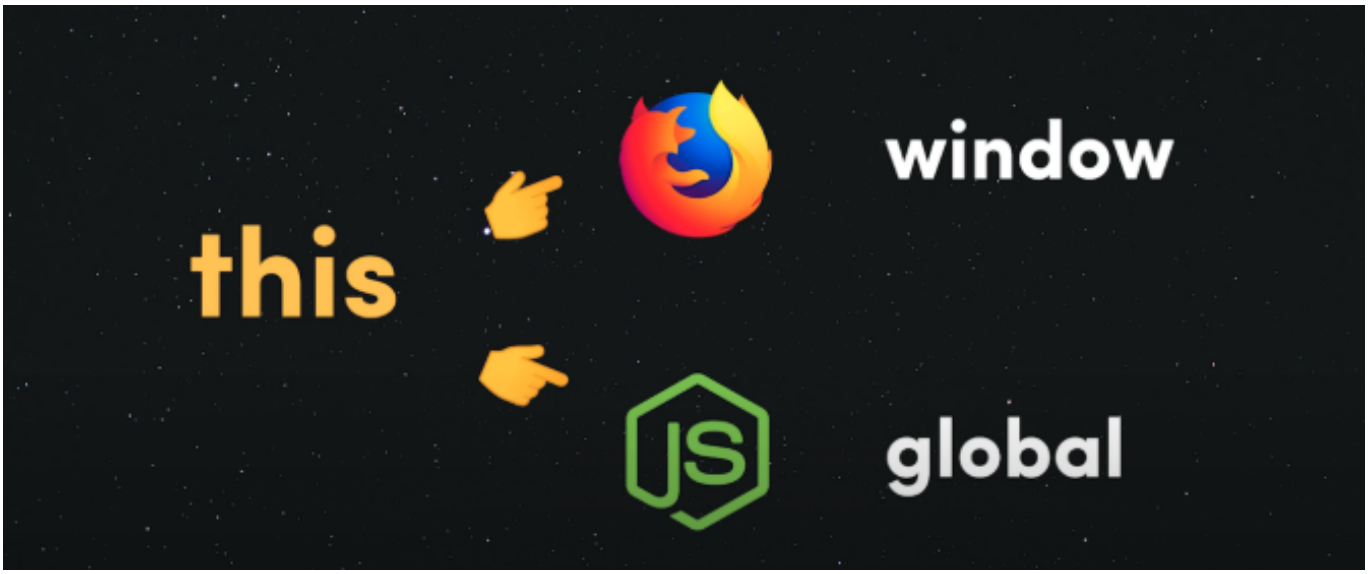
> Source: https://www.tutorialstonight.com/js/this-in-javascript.php

1. Overview

- this = current execution context (bối cảnh thực thi hiện tại)
- this can be different in strict and non-strict mode (this có thể khác ở chế độ nghiêm ngặt và không nghiêm ngặt)
- this has different value depending on where it is used (this có giá trị khác nhau tùy thuộc vào nơi nó được sử dụng)

In	this refers to
alone / outside function	global object
normal function in strict mode	undefined
normal function	depends on how the func is called, default to global object
arrow function	lexical this / this from outer normal function
a method	owner object (object chủ)
an event	an element that received the event

2. this in global context




```
// browser
console.log(this); // window object
console.log(this === window); // true
this.name = 'Do Long';
console.log(window.name); // 'Do Long'
```

```
// nodejs
console.log(this); // global object
console.log(this === global); // true
this.name = 'Do Long';
console.log(global.name); // 'Do Long'
```

3. this in function context

Mặc định thì `this` sẽ đề cập đến **global object**

```
// non-strict mode
function sayHello() {
 console.log(this); // window or global
}

// anonymous function
[1, 2, 3].forEach(function(number) {
 console.log(this); // window or global
})
```

Nếu trong chế độ `'strict mode'` thì sẽ là `undefined`

```
'use strict'
function sayHello() {
```

```
 console.log(this); // undefined
}
// anonymous function
[1, 2, 3].forEach(function(number) {
 console.log(this); // undefined
})
```

4. This in arrow function

- `this` trong `arrow function` không có `this` riêng của nó.
- `arrow function` không tạo ra một ngữ cảnh `this` mới, nó sẽ sử dụng ngữ cảnh `this` của phạm vi bên ngoài nó (lexical this).

```
const sayHello = () => {
 console.log(this); // window or global
}
```

```
'use strict'
function sayHello() {
  console.log(this); // undefined
  const getLanguage = () => {
    console.log(this); // undefined
  }
  getLanguage();
}
```

5. this in a method

Trong phương thức, `this` đề cập đến **object chủ quản**

```
const student = {
  name: 'Bob',
  // ES5 property methods
  sayHello: function() {
    console.log('My name is', this.name);
  },
}
student.sayHello(); // 'My name is Bob'
```

```
const student = {
  name: 'Bob',
  // ES6 property methods
  sayHello() {
    console.log('My name is', this.name);
```

```
  }
}
student.sayHello(); // 'My name is Bob'
```

```
'use strict'
// avoid using arrow function in object methods
const student = {
  name: 'Bob',

  // arrow function
  sayHello: () => {
    console.log('My name is', this.name);
  }
}
student.sayHello(); // 'My name is undefined'
```

Trong JavaScript, khi sử dụng một arrow function như một phương thức trong một object, this trong arrow function sẽ không trỏ đến object chứa nó như mong đợi. Thay vào đó, this sẽ trỏ đến ngữ cảnh (context) của hàm bên ngoài, thường là global object hoặc undefined trong strict mode.

Lý do là bởi vì arrow function không tạo ra một ngữ cảnh this mới; thay vào đó, nó sử dụng ngữ cảnh this của hàm bên ngoài, là ngữ cảnh mà nó được khai báo trong đó. Trong trường hợp này, arrow function được khai báo trong phương thức của một object, nhưng không phải là một phần của object đó, vì vậy this không được gắn với object đó.

6. this in callback

`this` trong một hàm callback không phụ thuộc vào nơi hàm đó được định nghĩa, mà phụ thuộc vào cách hàm đó được gọi. Trong trường hợp của setTimeout, hàm callback được gọi trong ngữ cảnh toàn cục (global context) hoặc trong ngữ cảnh của hàm setTimeout (nếu bạn sử dụng "use strict").

`this` trong đoạn code này sẽ không đề cập đến object `delay`

```
const delay = {
  lastName: 'Long',
  print() {
    setTimeout(function () {
      console.log(this.lastName) // undefined
    }, 1000)
  }
}
delay.print()
```

để fix vấn đề này thì có thể dùng **arrow function**

```
const delay = {
  lastName: 'Long',
  print() {
    setTimeout(() => {
      console.log(this.lastName) // Long
    }, 1000)
  }
}
delay.print()
```

Lưu ý là `this` trong callback không đề cập đến function chứa callback đó, hãy cẩn thận! `this` dưới đây không đề cập đến `broke` mà nó đề cấp đến obj.

```
function broke(func) {
  const obj = {
    name: 'Long',
    func
  }
  return obj.func()
}

broke(function () {
  console.log(this) // obj
})
```

Vì thế để biết this trong callback đề cập đến cái nào thì phải hiểu được hàm chứa callback gọi callback như thế nào.

7. this ở trong một Event Handler (học buổi DOM)

Trong một HTML event handler, `this` đề cập đến HTML element mà nó nhận event.

Khi nhấn vào button dưới đây thì nó sẽ được set `display:none`

```
<button onclick="this.style.display='none'">Click to Remove Me!</button>
```

8. this in class (Học buổi sau)

9. Bài tập

```
// bài 1: this trong phương thức
const obj = {
    name: "John",
    sayHello: function () {
        console.log("this 1", this); // this trỏ về đâu
        const innerFunc = () => {
```

```
            console.log("this 2", this); // this trỏ về đâu
            console.log("this.name", `Hello, ${this.name}!`); // this trỏ về đâu
        };
        innerFunc();
    },
};

obj.sayHello();
```

```
// bài 2: this trong context khác nhau
const person = {
  name: "Alice",
  greet: function() {
    console.log("Hello, " + this.name + "!");
  }
};

const greetFunc = person.greet;
greetFunc(); // What does this print? Why?
```

```
//bài 3: this trong nested function
const obj = {
  name: "John",
  outerFunc: function() {
    console.log("Outer this:", this);
    const innerFunc = function() {
      console.log("Inner this:", this);
    };
    innerFunc();
  }
};

obj.outerFunc();
```

```
// bài 4: this trong setTimeout (callback)
const obj = {
  message: "Hello",
  greet: function() {
    setTimeout(function() {
      console.log(this.message);
    }, 1000);
  }
};

obj.greet();
```

Giải bài tập

Bài 1: this trong phương thức

```
const obj = {
  name: "John",
  sayHello: function() {
    console.log(this); // { name: 'John', sayHello: [Function: sayHello] }
    const innerFunc = () => {
      console.log(this); // { name: 'John', sayHello: [Function: sayHello] }
      console.log(`Hello, ${this.name}!`); // Hello, John!
    };
    innerFunc();
  }
};

obj.sayHello();
```

- this trong function ở ngữ cảnh này thì trỏ về object global
- this trong arrow function thì trỏ về this của function bên ngoài nó (lexical this)

Bài 2: this trong context khác nhau

```
const person = {
  name: "Alice",
  greet: function() {
    console.log("Hello, " + this.name + "!");
  }
};

const greetFunc = person.greet;
greetFunc(); // Output: Hello, undefined!
```

- Kết quả in ra là "**Hello, undefined!**"
- Điều này xảy ra vì khi gán greet vào greetFunc, không có ngữ cảnh nào được cung cấp cho this, nó chỉ là một hàm độc lập không được gắn với bất kỳ đối tượng nào, do đó this.name sẽ là undefined.

Bài 3 : this trong nested function

```
const obj = {
  name: "John",
  outerFunc: function() {
    console.log("Outer this:", this); // Output: obj global
    const innerFunc = function() {
      console.log("Inner this:", this); // Output: global object (window in
browser, undefined in strict mode)
    };
    innerFunc();
```

```
  }
};

obj.outerFunc();
```

Khi gọi `obj.outerFunc()`, this trong outerFunc được liên kết với đối tượng obj. Tuy nhiên, khi innerFunc được gọi bên trong outerFunc, this không còn liên kết với obj mà thay vào đó nó trỏ đến global object hoặc undefined nếu trong `'strict mode'`.

bài 4: this trong setTimeout (callback)

```
const obj = {
  message: "Hello",
  greet: function() {
    setTimeout(function() {
      console.log(this.message);
    }, 1000);
  }
};

obj.greet(); // Output: undefined (or throws an error if in strict mode)
```

Trong hàm callback của setTimeout, this không liên kết với đối tượng obj mà thay vào đó nó trỏ đến global object hoặc undefined trong strict mode. Do đó, this.message sẽ là undefined.

Để fix vấn đề này thì có thể dùng arrow function

```
const obj = {
  message: "Hello",
  greet: function() {
    setTimeout(() => {
      console.log(this.message);
    }, 1000);
  }
};

obj.greet(); // Output: Hello
```