JavaScript 的 this

向鮎田

2018/11/16

當函數是在全域的環境下執行，this 會等於 window 物件

JavaScript 的 this

- JavaScript 的 this 是根據函數如何被呼叫而決定。
- This 沒辦法在函數執行時改變
- 有可能每次函數被呼叫時 this 的值都會不一樣。
- 要改變 this 可以透過 .call(), .bind(), .apply()。
- this 的值和呼叫 function 的物件是同一個。

Global Context

- 當函數是在全域環境下執行，this 會等於 window 物件

```
var myFunction = function(){
    console.log(this);
    console.log(this=== window);
};
myFunction();
```

當函數是在全域的環境下執行，this 會等於 window 物件

在 strict 模式，this 會是 undefined

```
'use strict'

var myFunction = function(){
    console.log(this);
    console.log(this=== window);
};
myFunction();
```

如果是巢狀函數呢？

```
function A() {
    function B() {
        console.log(this === window);
    }

    B();
}

A();
```

函數做為物件的方法

- this 會等於 method 所在的物件

```
var dog = {
    name: 'Doggie'
};
var animal = {
    weight: 100,
    walk: function () {
        console.log(this.weight);
        console.log(window.dog);
        console.log(this === animal);
        console.log(this);
    }
};

animal.walk();
```

另一個例子

```
var availableStock = 37;

var product = {
    name: 'Apple iPhone X'
};

var fullName = function () {
    console.log(this.name);
    console.log(window.availableStock);
    console.log(this === product);
    console.log(this);
    console.log(this === window);
};

product.getFullName = fullName;
product.getFullName();
```

建構子，new 繫結

```
function Product(name) {
    this.name = name;
}

var phone = new Product('Apple iPhone X');

console.log(phone.name);
```

IIFE 在函數之外

```
(function () {
    console.log(this);
})();
```

IIFE 在函數之內

```
var obj = {};

var someFunc = function () {
    console.log(this === obj);

    // IIFE
    (function () {
        console.log("IIFE this");
        console.log(this); // Output: Window object
        console.log("IIFE");
    })()
};

obj.func = someFunc;

obj.func();
```

隱含的繫結

```
function run() {
    console.log(this.model);
}

var car = {
    model: 'nissan',
    run: run
}

car.run();
```

失去隱含的繫結

```
function run() {
    console.log(this.model);
}

var car = {
    model: 'nissan',
    run: run
}

var alt = car.run;
alt();
```

明確的繫結

```
function run() {
    console.log(this);
}

var car = {
    maker: 'nissan',
    model: 'tiida'
}

run.call(car);
```

硬繋結

```
function run() {
    console.log(this);
}

var car = {
    maker: 'nissan',
    model: 'tiida'
}

var carRun = run.bind(car);

carRun();
```

硬繋結

```
function run() {
    console.log(this);
}

var car = {
    maker: 'nissan',
    model: 'tiida'
}

run.bind(car)();
```

硬繋結

```
function run() {
    console.log(this);
}

var car = {
    maker: 'nissan',
    model: 'tiida'
}

run.bind(car)();
```

決定 this 為何？

函數呼叫是以 new 進行的嗎？

- 若是， this 就是那個新建構出來的物件

```
var bar = new foo();
```

函數呼叫是透過 call 或是 apply 呼叫的嗎？

- 若是， this 就是那個明確指定的的物件

```
var bar = foo().call(obj)
```

函數是透過一個 Context 被呼叫的嗎？

- 若是， this 就是那個情境物件

```
var bar = obj.foo();
```

如果前面都不適用

- this 就是預設值
- 在 strict mode，this = undefined
- 在一般模式，this = global 物件(window)