

JavaScript进阶

---JS中的this



河北师范大学软件学院
Software College of Hebei Normal University

内容提纲

- **JS this简介及特点**
- **JS this四种应用场景**
- **JS this缺陷及解决方法**



JavaScript语言中的this和其他语言中区别

- Java等面向对象语言中的this

在 Java 等面向对象的语言中，this 关键字的含义是明确且具体的，即指代当前对象，一般在编译期就已经确定下来，称为**编译期绑定**

```
public class Point{
    private int x = 0;
    private int y = 0;

    public Point(x,y){
        this.x = x;
        this.y = y;
    }
}
```

- JavaScript语言中的this

在 JavaScript 中，this 是动态绑定，或称为**运行期绑定**的。由于其运行期绑定的特性，JavaScript 中的 this 含义要丰富得多，它可以是**全局对象**、**当前对象**或者**任意对象**，这完全取决于函数的调用方式

- this不进行作用域传递（函数嵌套时的this缺陷）

内容提纲

- JS this简介及特点
- JS this四种应用场景
- JS this缺陷和解决方法



JavaScript语言中的this主要有以下4种应用场景

- 一般函数中的this
- 对象方法中的this
- 构造函数中的this
- 间接调用中的this

一、一般函数中的this（非严格松散模式下）

- 一般函数中的this（非严格松散模式下）指代全局对象

```
> function thisTest(){  
    console.log(this === window);  
}  
thisTest();
```

true

- 可以通过this在函数内添加、删除、修改全局对象属性

```
> var a=10,b="Hi";  
    function thisTest(){  
        this.a=20;  
        delete this.b;  
        this.c = "新添加的全局变量";  
    }  
    thisTest();  
    console.log(a,c);
```

20 "新添加的全局变量"



一、一般函数中的this（严格模式）

- 一般函数中的this（严格模式）为undefined

```
> function thisTest(){  
    'use strict'  
    console.log(this);  
}  
thisTest();
```

undefined

- 可以用此特性来判断当前是否为严格模式

```
> //'use strict'  
function isStrictMode(){  
    return this===undefined?true:false;  
}  
isStrictMode();  
← false
```



二、对象方法中的this（无函数嵌套的情况下）

- 函数作为对象的一个属性时，称之为对象的方法
- 对象方法中的this指代调用此方法的对象（无嵌套的情况下）

```
> var point = {  
  x : 0,  
  y : 0,  
  moveTo : function(x, y) {  
    this.x = x;  
    this.y = y;  
  }  
};  
point.moveTo(1, 1)//this 绑定到当前对象，即 point 对象  
console.log(point);  
  
► Object {x: 1, y: 1}
```



三、构造函数中的this

- 构造函数中的this指代通过new新创建的对象

- JS (ES5) 并没有类 (class) 的概念, 而是使用基于原型 (prototype) 的继承方式
- JS中的构造函数充当了类的角色, 如果不使用 new 调用, 则和普通函数一样。
- 如果作为构造函数正确调用时, 构造函数中的this 绑定到新创建的对象上

```
> function Point(x, y){  
    this.x = x;  
    this.y = y;  
}  
var p = new Point(2,3);  
p;  
◀ ▶ Point {x: 2, y: 3}
```

四、间接调用中的this (call、apply)

- 通过call/apply间接调用的函数中的this (指代第一个参数)
 - JS中函数也是对象 (函数对象), 也有属性和方法 (length、call、apply等)
 - JS中函数可以通过call和apply进行间接调用, 动态的指定由谁来调用此函数

```
objA = {name:"AA",x:1};  
objB = {name:"BB",x:5};  
function test(){  
    console.log(this.name,this.x);  
}  
objA.fun = test;  
objA.fun(); //AA 1  
objA.fun.call(objB); //BB 5
```

内容提纲

- JS this简介及特点
- JS this四种应用场景
- JS this缺陷和解决方法



对象方法中的this（有函数嵌套的情况下）

- this不进行作用域传递、函数嵌套中的this存在缺陷

```
var point = {  
  x : 0,  
  y : 0,  
  moveTo : function(x, y) {  
    // 内部函数  
    function moveToX(x) {  
      this.x = x; // this 绑定到了哪里?  
    };  
    // 内部函数  
    function moveToY(y) {  
      this.y = y; // this 绑定到了哪里?  
    };  
    moveToX(x);  
    moveToY(y);  
  }  
};
```

```
point.moveTo(2,2);  
console.log(point);  
console.log(window.x,window.y);  
► Object {x: 0, y: 0, moveTo: function}
```

2 2

moveTo方法中嵌套的
moveToX函数作为一般函数
看待，此函数中的this此时指
向的是全局对象window而不
是point



如何解决对象方法中嵌套函数的this指向问题

- 使用变量替代的方法 (that、self)

```
> var point = {  
  x : 0,  
  y : 0,  
  moveTo : function(x, y) {  
    var that = this;  
    // 内部函数  
    function moveToX(x) {  
      that.x = x; //此时that指向point  
    };  
    // 内部函数  
    function moveToY(y) {  
      that.y = y; //此时that指向point  
    };  
    moveToX(x);  
    moveToY(y);  
  }  
};
```

```
> point.moveTo(2,2);  
console.log(point);  
console.log(x,y); //报ReferenceError  
  
▶ Object {x: 2, y: 2}
```

moveTo方法中嵌套的函数中的that此时指向的是point对象。

这是软绑定形式，除此之外，还可以通过函数对象的bind方法进行硬绑定或通过call/apply间接指定this对象



构造函数中的this（有函数嵌套的情况下）

- 构造函数中的this同样存在函数嵌套缺陷，**解决办法同上**

```
function Point(x, y){
    this.x = x;
    this.y = y;
    this.moveXY = function(x,y){
        function moveX(x){
            this.x+=x;
        }
        function moveY(y){
            this.y+=y;
        }
        moveX(x);
        moveY(y);
    };
}
var p = new Point(2,3);
p.moveXY(1,1);
console.log(p);//输出为 Piont{x:2,y:3}
```

```
function Point(x, y){
    this.x = x;
    this.y = y;
    this.moveXY = function(x,y){
        var that = this;//此处的this指new出来的新对象
        function moveX(x){
            that.x+=x;
        }
        function moveY(y){
            that.y+=y;
        }
        moveX(x);
        moveY(y);
    };
}
var p = new Point(2,3);
p.moveXY(1,1);
console.log(p);//输出为 Piont{x:3,y:4}
```

总结

- JS this简介及特点
- JS this四种应用场景
- JS this缺陷和解决方法



补充：JS this本质概述

- 函数对象的内部方法 `[[call]]`
 - 函数对象有一个叫`[[Call]]`内部方法，函数的执行其实是通过`[[Call]]`方法来执行的
 - `[[Call]]`方法接收两个参数`thisArg`和`argumentList`
 - `thisArg`和`this`的指向有直接关系，`argumentList`为函数的实参列表
- `thisArg`与4种情况的对应关系
 - 普通方法调用`thisArg`为`undefined`。
 - 通过`call`或`apply`调用，`thisArg`既为第一个参数。
 - 通过对象调用，`thisArg`指向该对象。
 - 在构造方法中，`thisArg`为新构造的对象
- 总原则：`this`指的是调用函数的那个对象

The background of the slide is decorated with various abstract shapes in shades of green and yellow. These shapes, which include circles, ovals, and irregular blobs, are scattered across the top and right sides of the slide, creating a modern, organic feel.

Thank You!



河北师范大学软件学院
Software College of Hebei Normal University

补充内容：函数对象的bind方法

- 补充bind，思考bind与call/apply的优缺点
- 思考：将一个对象方法赋值给某个变量后，变量调用时，此时的this指的是什么，是否会有问题
- 更为深入理解this的
 - <https://segmentfault.com/a/1190000003906484>
 - <https://segmentfault.com/a/1190000003802851>
 - <http://es5.github.io/#x15.3>

间接调用中的this (bind)

- bind后，函数中的this指向绑定的对象，硬绑定

```
var x = 0;  
function test(){  
    console.log(this.x);  
}  
var obj = {};  
obj.x = 1;  
var newTest = test.bind(obj);  
newTest(); // 0还是1?
```

