2016年11月24日 星期四

# this

---

**1、定义：this是包含它的函数作为方法被调用时所属的对象。**

说明：这句话有点咬嘴，但一个多余的字也没有，定义非常准确，我们可以分3部分来理解它！

1、包含它的函数。2、作为方法被调用时。3、所属的对象。

看例子：

```
function to_green(){
  this.style.color="green";
}
to_green();
```

上面函数中的this指的是谁?

分析：包含this的函数是，to_green

该函数作为方法被调用了

该函数所属的对象是。。？我们知道默认情况下，都是window对象。

OK，this就是指的window对象了，to_green中执行语句也就变为，window.style.color="green"

这让window很上火，因为它并没有style这么个属性，所以该语句也就没什么作用。

我们在改一下。

```
window.load=function(){
   var example=document.getElementById("example");
   example.onclick=to_green;
}
```

这时this又是什么呢?

我们知道通过赋值操作，example对象的onclick得到to_green的方法，那么包含this的函数就是onclick，

那么this就是example引用的html对象。

this的环境可以随着函数被赋值给不同的对象而改变！

## 2、在JS基础语法中

情况一：纯粹的函数调用

这是函数的最通常用法，属于全局性调用，因此this就代表全局对象Global。

请看下面这段代码，它的运行结果是1。

```
function test(){
    this.x = 1;
    alert(this.x);
}
test(); // 1
```

为了证明this就是全局对象，我对代码做一些改变：

```
var x = 1;
function test(){
    alert(this.x);
}
test(); // 1
```

运行结果还是1。再变一下：

```
var x = 1;
function test(){
    this.x = 0;
}
test();
alert(x); //0
```

情况二：作为对象方法的调用

函数还可以作为某个对象的方法调用，这时this就指这个上级对象。

```
function test(){
    alert(this.x);
}
var o = {};
o.x = 1;
o.m = test;
o.m(); // 1
```

情况三 作为构造函数调用

所谓构造函数，就是通过这个函数生成一个新对象（object）。这时，this就指这个新对象。

```
function test(){
    this.x = 1;
}
var o = new test();
alert(o.x); // 1
```

运行结果为1。为了表明这时this不是全局对象，我对代码做一些改变：

```
var x = 2;
function test(){
    this.x = 1;
}
var o = new test();
alert(x); //2
```

运行结果为2，表明全局变量x的值根本没变。

情况四 apply调用

apply()是函数对象的一个方法，它的作用是改变函数的调用对象，它的第一个参数就表示改变后的调用这个函数的对象。因此，this指的就是这第一个参数。

```
var x = 0;
function test(){
    alert(this.x);
}
var o={};
o.x = 1;
o.m = test;
o.m.apply(); //0
```

apply()的参数为空时，默认调用全局对象。因此，这时的运行结果为0，证明this指的是全局对象。

如果把最后一行代码修改为

```
o.m.apply(o); //1
```

运行结果就变成了1，证明了这时this代表的是对象o。

## 3、this会出现的问题

```
var o = {
  x : 1,
  func : function() { console.log(this.x) },
  test : function() {
    setTimeout(function() {
      this.func();
    }, 100);
  }
};
//这个时候this.func()中的this是global对象，因为 setTimeout(function() {…
//完整写法为：global. setTimeout(function() {…
```

o.test(); // TypeError : this.func is not a function

上面代码得不到预期结果，因为 this 发生了改变，这个问题可以这样修正：

JS

```
var o = {
    x : 1,
    func : function() { console.log(this.x) },
    test : function() {
        var _this = this;
        setTimeout(function() {
            _this.func();
        }, 100);
    }
};

o.test();
```