this 的值到底是什么？一次说清楚

作者：方应杭
链接：https://zhuanlan.zhihu.com/p/23804247
来源：知乎

你可能遇到过这样的 JS 面试题：

```
var obj = {
  foo: function(){
    console.log(this)
  }
}

var bar = obj.foo
obj.foo() // 打印出的 this 是 obj
bar() // 打印出的 this 是 window
```

请解释最后两行函数的值为什么不一样。

初学者关于 this 的理解一直很模糊。今天这篇文章就要一次讲清楚了。

而且这个解释，你在别的地方看不到。看懂这篇文章，所有关于 this 的面试题，都是小菜。

有用请点赞。

函数调用

首先需要从函数的调用开始讲起。

JS（ES5）里面有三种函数调用形式：

```
func(p1, p2)
obj.child.method(p1, p2)
func.call(context, p1, p2) // 先不讲 apply
```

一般，初学者都知道前两种形式，而且认为前两种形式「优于」第三种形式。

从看到这篇文章起，你一定要记住，第三种调用形式，才是正常调用形式：

```
func.call(context, p1, p2)
```

其他两种都是语法糖，可以等价地变为 call 形式：

```
func(p1, p2) 等价于
func.call(undefined, p1, p2)

obj.child.method(p1, p2) 等价于
obj.child.method.call(obj.child, p1, p2)
```

请记下来。（我们称此代码为「转换代码」，方便下文引用）

至此我们的函数调用只有一种形式：

```
func.call(context, p1, p2)
```

这样，this 就好解释了

this，就是上面代码中的 context。就这么简单。

this 是你 call 一个函数时传的 context，由于你从来不用 call 形式的函数调用，所以你一直不知道。

先看 func(p1, p2) 中的 this 如何确定：

当你写下面代码时

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

func()
```

等价于

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

func.call(undefined) // 可以简写为 func.call()
```

按理说打印出来的 this 应该就是 undefined 了吧，但是浏览器里有一条规则：

> 如果你传的 context 就 null 或者 undefined，那么 window 对象就是默认的 context（严格模式下默认 context 是 undefined）

因此上面的打印结果是 window。

如果你希望这里的 this 不是 window，很简单：

```
func.call(obj) // 那么里面的 this 就是 obj 对象了
```

再看 obj.child.method(p1, p2) 的 this 如何确定

```
var obj = {
  foo: function(){
    console.log(this)
  }
}

obj.foo()
```

按照「转换代码」，我们将 obj.foo() 转换为

```
obj.foo.call(obj)
```

好了，this 就是 obj。搞定。

回到题目：

```
var obj = {
  foo: function(){
    console.log(this)
  }
}

var bar = obj.foo
obj.foo() // 转换为 obj.foo.call(obj)，this 就是 obj
bar()
// 转换为 bar.call()
// 由于没有传 context
// 所以 this 就是 undefined
// 最后浏览器给你一个默认的 this —— window 对象
```

[ ] 语法

```
function fn (){ console.log(this) }
var arr = [fn, fn2]
arr[0]() // 这里面的 this 又是什么呢？
```

我们可以把 arr[0]( ) 想象为arr.0( )，虽然后者的语法错了，但是形式与转换代码里的 obj.child.method(p1, p2) 对应上了，于是就可以愉快的转换了：

```
        arr[0]()
假想为    arr.0()
然后转换为 arr.0.call(arr)
那么里面的 this 就是 arr 了 :)
```

总结

1. this 就是你 call 一个函数时，传入的 context。
2. 如果你的函数调用形式不是 call 形式，请按照「转换代码」将其转换为 call 形式。

以后你遇到所有跟 this 有关的笔试题，都不会有疑问了。

完。

更多精彩教程，请看我的首页加 QQ 群。

P.S.

1. 使用 new 时，情况又不一样，可以看另一篇文章《JS 的 new 到底是干什么的？》。
2. 有人说你怎么不讲 strict mode 呢，strict mode 也会影响 this 呀。我认为 strict mode 只是影响了 context 的默认值而已，你看懂此文稍微看看 strict mode 就懂了。我只讲最重要的内容。
3. 有人问怎么不讲 bind 对 this 的影响。那是因为 bind 本质就是自动在你调用一个函数的时候，把 context 换掉而已，你看看 bind 的 polyfill 就知道，其核心依然是 apply（跟 call 差不多）。

补充阮一峰一题

```
var x = 0;

function test(){

    console.log(this.x);

}

var o={};

o.x = 1;
```

```
        o.m = test;

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