apply

虽然在一个独立的函数调用中，根据是否是strict模式，this指向undefined或window，不过，我们还是可以控制this的指向的！

要指定函数的this指向哪个对象，可以用函数本身的apply方法，它接收两个参数，第一个参数就是需要绑定的this变量，第二个参数是Array，表示函数本身的参数。

用apply修复getAge()调用：

**function** getAge() {

**var** y = **new** Date().getFullYear();

**return** y - **this**.birth;

}

**var** xiaoming = {

name: '小明',

birth: 1990,

age: getAge

};

xiaoming.age(); *// 25*

getAge.apply(xiaoming, []); *// 25, this指向xiaoming, 参数为空*

另一个与apply()类似的方法是call()，唯一区别是：

* apply()把参数打包成Array再传入；
* call()把参数按顺序传入。

比如调用Math.max(3, 5, 4)，分别用apply()和call()实现如下：

Math.max.apply(null, [3, 5, 4]); *// 5*

Math.max.call(null, 3, 5, 4); *// 5*

对普通函数调用，我们通常把this绑定为null。

### 装饰器

利用apply()，我们还可以动态改变函数的行为。

JavaScript的所有对象都是动态的，即使内置的函数，我们也可以重新指向新的函数。

现在假定我们想统计一下代码一共调用了多少次parseInt()，可以把所有的调用都找出来，然后手动加上count += 1，不过这样做太傻了。最佳方案是用我们自己的函数替换掉默认的parseInt()：

窗体顶端

'use strict';

var count = 0;

var oldParseInt = parseInt; // 保存原函数

window.parseInt = function () {

count += 1;

return oldParseInt.apply(null, arguments); // 调用原函数

};

窗体底端

高阶函数

高阶函数英文叫Higher-order function。那么什么是高阶函数？

JavaScript的函数其实都指向某个变量。既然变量可以指向函数，函数的参数能接收变量，那么一个函数就可以接收另一个函数作为参数，这种函数就称之为高阶函数。

一个最简单的高阶函数：

**function** add(x, y, f) {

**return** f(x) + f(y);

}

当我们调用add(-5, 6, Math.abs)时，参数x，y和f分别接收-5，6和函数Math.abs，根据函数定义，我们可以推导计算过程为：

x = -5;

y = 6;

f = Math.abs;

f(x) + f(y) ==> Math.abs(-5) + Math.abs(6) ==> 11;

**return** 11;