this的定义

this是js中的一个关键字，返回一个对象，可以理解为上下文关系；this在不同位置调用会返回不同的对象，大概可分为3种：
1.在函数中：this通常是一个隐含的参数。
2.在函数外（顶级作用域）：在浏览器中this值的是全局对象；在Node.js中指的是模块(module)的导出(exprots)。
3.传递到eval()中的字符串：如果eval()是被直接调用，this指的是当前对象；如果eval()是被间接调用，this为全局对象。

在函数中又可分为几种情况，纯粹的函数调用，作为对象方法的调用，作为构造函数的调用。
纯函数调用this返回window，对象方法调用返回该对象，构造函数调用this指向实例对象。

综上：this的指向跟函数的定义没有关系，this的指向调用对象。

更改this指向

根据调用方式的不同，this的指向也会不同，但我们可以通过call、apply和bind更改this的指向

func.call(context, p1, p2)，通过call关键词调用函数，this的指向可变更为context，若不传context参数时...
如果 context 是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）
通过此方法，可以判断绝大多数情况下this的指向；apply类同，只是函数参数是以伪数组的形式传入。

bind：es5新api，生成一个新的函数，称之为绑定函数，传入bind方法的第一个参数作为这个绑定函数的this对象，从第二个参数开始依照先后顺序构成绑定函数的参数

Function.prototype.bind() 实现方法

```
Function.prototype.bind = function (scope) {
   var fn = this;//this是调用bind方法的对象（别的方法对象）
```

```
    return function () {
        return fn.apply(scope);//把fn环境中的this替换为scope
    };
}
```

bind与call、apply的区别：call、apply是改动函数的作用域且马上运行。而bind内部调用apply返回一个新的函数，不是马上运行。

严格模式和箭头函数

严格模式中函数体内的this指向undefined；箭头函数体内的this对象是定义时所在的对象，因为箭头函数的this是继承自外面的。

总结：this指向最近调用的对象。

作用域和执行上下文

上下文（context）指的是一种环境，函数被调用时， this指向的那个object。
作用域（scope）指的是函数被调用时，各个变量的作用区域。