Day05This

讨论this的指向情况

如果一个普通函数中有this，那么this指向的就是window。this的指向在函数创建的时候是决定不了的，在调用的时候才能决定，谁调用的就指向谁

```
function a(){
    var user = "追梦子";
    console.log(this.user); //undefined
    console.log(this); //Window
}
a();
```

```
var o = {
    user:"追梦子",
    fn:function(){
        console.log(this.user);  //追梦子
    }
}
o.fn();
```

```
错误示例：function Demo() {
                this.a = 12;

                setInterval(this.show, 1000);
        }

        Demo.prototype.show = function() {
                alert(this.a);
        };


        var demo = new Demo();
```

当在定时器中调用了this，this会指向window

```
解决方法：把定时器中的this换为正确的值:function Demo() {
                this.a = 12;
                var _this = this;
                setInterval(function(){
                        _this.show();
                }, 1000);
        }

        Demo.prototype.show = function() {
                alert(this.a);
        };


        var demo = new Demo();
```

如果一个对象中的方法有this，这个方法被上一级的对象所调用，那么this指向的就是上一级的对象

```
var obj = {
    name: "我的花名改为云溪了，就是为了好玩",
    getName: function(){
        console.log(this); // 在这里this指向于obj对象了
        console.log(this.name); // 打印我的花名改为云溪了，就是为了好玩
    }
};
obj.getName(); // 对象方法调用
```

```
var name = "全局对象名字";
var obj = {
    name: "我的花名改为云溪了，就是为了好玩",
    getName: function(){
        console.log(this);  // window
        console.log(this.name); // 全局对象名字
    }
};
var yunxi = obj.getName;
yunxi();
```

构造函数中的this指向

构造函数中的this会指向创建出来的对象

```
function Fn(){
    this.user = "追梦子";
}
var a = new Fn();
console.log(a.user); //追梦子
```

原理：使用new调用构造函数时，会先创建出一个空对象，然后调用call函数把构造函数中的指针修改为指向这个空对象，执行完构造函数后，这个空对象也就有了相关的属性方法

注意：当构造函数返回一个对象时，new创建的空对像中存放的就是返回出来的对象，this指针会指向返回的这个对象；返回其他类型的值（包括null）这不会更改this的指向

```
function fn()
{
    this.user = '追梦子';
    return {};
}
var a = new fn;
console.log(a.user); //undefined
```

```
function fn()
{
    this.user = '追梦子';
    return function(){};
}
var a = new fn;
console.log(a.user); //undefined
```

```
function fn()
{
    this.user = '追梦子';
    return 1;
}
var a = new fn;
console.log(a.user); //追梦子
```

```
function fn()
{
    this.user = '追梦子';
    return undefined;
}
var a = new fn;
console.log(a.user); //追梦子
```

```
function fn()
{
    this.user = '追梦子';
    return null;
}
var a = new fn;
console.log(a.user); //追梦子
```

在事件中调用了this，this指向触发事件的元素

```
function Btn() {
        this.b = 23;

document.getElementById('button').onclick = this.show;
        }
        Btn.prototype.show = function() {
                alert(this.b);
        };
        window.onload = function () {
                new Btn();
        }
```

```
解决方法：function Btn() {
                var _this = this;
                this.b = 23;
                document.getElementById('button').onclick = function () {
                        _this.show();
                };
        }
        Btn.prototype.show = function() {
                alert(this.b);
        };
        window.onload = function () {
                new Btn();
        }
```

http://blog.csdn.net/louisliaoxh/article/details/50323937

http://www.aiweibang.com/yuedu/92793184.html

使用call/apply修改this指针的指向

call

```
var longen = {
    name:'yunxi'
};
var longen2 = {
    name: '我叫涂根华'
};
var name = "我是来测试的";
var getName = function(){
    return this.name;
};
console.log(getName()); // 打印 "我是来测试的";
console.log(getName.call(longen)); // 打印 yunxi
console.log(getName.call(longen2)); // 打印 "我叫涂根华"
```

案例：

```
document.getElementById("longen").onclick = function(){
    console.log(this); // this 就指向于div元素对象了
    var func = function(){
        console.log(this); // 打印出window对象
    }
    func();
}
```

解决办法：

```
document.getElementById("longen").onclick = function(){
    console.log(this); // this 就指向于div元素对象了
    var func = function(){
        console.log(this); // 就指向于div元素对象了
    }
    func.call(this);
}
```

不使用call的解决办法：找一个变量把this所指向的标签先保存下来，然后在内部函数中使用这个临时变量

```
document.getElementById("longen").onclick = function(){
    console.log(this); // this 就指向于div元素对象了
    var self = this;
    var func = function(){
        console.log(self); // 就指向于div元素对象了
    }
    func();
}
```

利用call修复bug：在事件函数中，this指向元素，但是在内部再包含一个函数后，在函数内再继续调用this的话，那么现在的this指针就指向了window了