接着上一节讲的话，应该轮到"执行上下文栈"了，但是这里不得不插入一节，把this说一下。因为this很重要，js的面试题如果不出几个与this有关的，那出题者都不合格。

其实，this的取值，分四种情况。我们来挨个看一下。
在此再强调一遍一个非常重要的知识点：在函数中this到底取何值，是在函数真正被调用执行的时候确定的，函数定义的时候确定不了。因为this的取值是执行上下文环境的一部分，每次调用函数，都会产生一个新的执行上下文环境。

情况1：构造函数

所谓构造函数就是用来new对象的函数。其实严格来说，所有的函数都可以new一个对象，但是有些函数的定义是为了new一个对象，而有些函数则不是。另外注意，构造函数的函数名第一个字母大写（规则约定）。例如：Object、Array、Function等。

```
function Foo() {
    this.name = '王福朋';
    this.year = 1988;

    console.log(this);  // Foo {name: "王福朋", year: 1988}
}

var f1 = new Foo();

console.log(f1.name);  // 王福朋
console.log(f1.year);  // 1988
```

以上代码中，如果函数作为构造函数用，那么其中的this就代表它即将new出来的对象。

注意，以上仅限new Foo()的情况，即Foo函数作为构造函数的情况。如果直接调用Foo函数，而不是new Foo()，情况就大不一样了。

```
function Foo() {
    this.name = '王福朋';
    this.year = 1988;

    console.log(this);  // Window {top: Window, window: Window, location: Location, exter
}

Foo();
```

这种情况下this是window，我们后文中会说到。

情况2：函数作为对象的一个属性

如果函数作为对象的一个属性时，并且作为对象的一个属性被调用时，函数中的this指向该对象。

```
var obj = {
    x: 10,
    fn: function () {
        console.log(this);    // Object {x: 10, fn: function}
        console.log(this.x);  // 10
    }
};

obj.fn();
```

以上代码中，fn不仅作为一个对象的一个属性，而且的确是作为对象的一个属性被调用。结果this就是obj对象。

注意，如果fn函数不作为obj的一个属性被调用，会是什么结果呢？

```
var obj = {
    x: 10,
    fn: function () {
        console.log(this);     // Window {top: Window, window: Wi
        console.log(this.x);  // undefined
    }
};

var fn1 = obj.fn;
fn1();
```

如上代码，如果fn函数被赋值到了另一个变量中，并没有作为obj的一个属性被调用，那么this的值就是window，this.x为undefined。

情况3：函数用call或者apply调用

当一个函数被call和apply调用时，this的值就取传入的对象的值。至于call和apply如何使用，不会的朋友可以去查查其他资料，本系列教程不做讲解。

```
var obj = {
    x: 10
};

var fn = function () {
    console.log(this);     //Object {x: 10}
    console.log(this.x);  //10
}
fn.call(obj);
```

情况4：全局 & 调用普通函数

在全局环境下，this永远是window，这个应该没有非议。

```
console.log(this === window);  // true
```

普通函数在调用时，其中的this也都是window。

```
var x = 10;

var fn = function () {
    console.log(this);     //Window {top: Window, window: Wi
    console.log(this.x);  //10
}
fn();
```

以上代码很好理解。

不过下面的情况你需要注意一下：

```
var obj = {
    x: 10,
    fn: function () {

        function f() {
            console.log(this);    //Window {top: Window, window: Wind
            console.log(this.x); //undefined
        }
        f();

    }
};
obj.fn();
```

函数f虽然是在obj.fn内部定义的，但是它仍然是一个普通的函数，this仍然指向window。

完了。

看到了吧，this有关的知识点还是挺多的，不仅多而且非常重要。

最后，既然提到了this，有必要把一个非常经典的案例介绍给大家，又是jQuery源码的。

```
jQuery.extend = jQuery.fn.extend = function () {

    // ……此处省略若干行……

    // extend jQuery itself if only one argument is passed
    if (i === length) {
        target = this;
        i--;
    }

    // ……此处省略若干行……
};
```

以上代码是从jQuery中摘除来的部分代码。jQuery.extend和jQuery.fn.extend都指向了同一个函数，但是当执行时，函数中的this是不一样的。

执行jQuery.extend(…)时，this指向jQuery；执行jQuery.fn.extend(…)时，this指向jQuery.fn。

这样就巧妙的将一段代码同时共享给两个功能使用，更加符合设计原则。

好了，聊完了this。接着上一节继续说“执行上下文栈”。

注意：还有一部分this的内容本文中没有讲到，已经补充到这里：

http://www.cnblogs.com/wangfupeng1988/p/3996037.html

本文对《深入理解javascript原型和闭包（10）——this》一篇进行补充，原文链接：

http://www.cnblogs.com/wangfupeng1988/p/3988422.html

原文中，讲解了在javascript中this的各个情况，写完之后发现还落下一种情况，就此补充。

原文中this的其中一种情况是构造函数的，具体的内容可以参考原文，此处不再赘述。

要补充的内容是，在构造函数的prototype中，this代表着什么。

```
1
2  function Fn() {
3      this.name = '王福朋';
4      this.year = 1988;
5  }
6
7  Fn.prototype.getName = function () {
8      console.log(this.name);
9  }
10
11 var f1 = new Fn();
12 f1.getName();          // 王福朋
13
```

如上代码，在Fn.prototype.getName函数中，this指向的是f1对象。因此可以通过this.name获取f1.name的值。

其实，不仅仅是构造函数的prototype，即便是在整个原型链中，this代表的也都是当前对象的值。