高级Day 05---this使用总结

this是在函数执行的过程中自动创建的一个指向一个对象的内部指针。确切的说，this并不是一个对象，而是指向一个已经存在的对象的指针，也可以认为是this就是存储了某个对象的地址。

this的指向不是固定的，会根据调用的不同，而指向不同的地方。

可以从两个大的方方向来分析：

一、一些认知的澄清

1.1 对在全局作用域中定义的变量和函数的进一步认识

永远记住：只要是在全局作用域声明的任何变量和函数默认都是作为window对象的属性而存在的.

> 看下面的代码：

```
<script type="text/javascript">
    var num = 10;
    function sum () {
        alert("我是一个函数");
    }
    alert(window.num);  // 10
    window.sum();    // 我是一个函数
    // 在调用的时候window对象是可以省略的。
</script>
```

1.2 构造函数和非构造函数的澄清

在JavaScript中构造函数和非构造函数没有本质的区别。唯一的区别只是调用方式的区别。

- 使用new 就是构造函数
- 直接调用就是非构造函数

> 看下面的代码：

```
<script type="text/javascript">
    function Person () {
        this.age = 20;
        this.sex = "男";
    }
    //作为构造函数调用,创建一个对象。 这个时候其实是给p添加了两个属性
    var p = new Person();
    alert(p.age + "  " + p.sex);

    //作为普通函数传递，其实是给 window对象添加了两个属性
    Person();
    alert(window.age + " " + window.sex);
</script>
```

二、在全局作用域中使用this

> 全局作用域中使用this，也就是说不在任何的函数内部使用this，那么这个时候this就是指的 window

> 看下面的代码：

```
<script type="text/javascript">
    //向this对象指代的对象中添加一个属性 num， 并让属性的值为100
    this.num = 100;
    // 因为this就是window，所以这时是在修改属性num的值为200
    window.num = 200;
    alert(this === window);  // true this就是指向的window对象，所以是恒等
    alert(this.num);    //200
    alert(window.num);  //200
</script>
```

三、在函数中this的指向

3.1 非构造函数中的this指向

> 非构造函数中this执行的就是 调用这个方法的那个对象

```
<script type="text/javascript">
    function test() {
        alert(this == window);
        this.age = 20;
    }
    test();  //其实是 window.test();  所以这个时候test中的this指向window
</script>
```

> 看下面的代码：

```
<script type="text/javascript">
    var p = {
        age : 20,
        sex : "女",
        sayAge: function (argument) {
            alert(this.age);
        }
    }
    p.sayAge(); //调用对象p的方法sayAge()  所以这个时候this指的是 p 这个对象
</script>
```

> 再看下面的代码：

```
<script type="text/javascript">
    var p = {
        age : 20,
        sex : "女",
        sayAge: function (argument) {
            alert(this.age);
            alert(this === p);  //true
        }
    }
    var again = p.sayAge;   //声明一个变量(方法)，把p的方法复制给新的变量
  //调用新的方法：  其实是window.again().  所以 方法中的this指代的是window对象，这个时候age属性是
undefined
  // this和p也是不相等的。
    again();
</script>
```

> 综上：this的指代和代码出现的位置无关，只和调用这个方法的对象有关。

3.2 构造方法中的this指向

> 构造方法中的this指代的要未来要创建的那个对象。

```
<script type="text/javascript">
    function Person () {
        this.age = 20;
       return this;  //作为构造函数的时候，这个行代码默认会添加
    }
    var p1 = new Person();  //这个时候 Person中的this就是指的p1
    var p2 = new Person();  //这是时候 Person中的this就是知道p2
</script>
```

> 多了解一点：其实用new调用构造函数的时候，构造函数内部其实有个默认的 return this; 这更容器为什么this指代那个要创建的对象了。

四、改变this的指向

> 在JavaScript中，允许更改this的执行。

> 通过call方法或apply方法

```
<script type="text/javascript">
    var age = 20;
    function showPropertyValue (propertyName) {
        alert(this[propertyName]);
    }
    //使用call的时候，第一个参数表示showPropertyValue中的this的执行，后面的参数为向这个函数传的值。
    //注意一点：如果第一个参数是null，则this仍然是默认的指向。
    showPropertyValue.call(null, "age");
    showPropertyValue.call(this, "age");
    showPropertyValue.call({age:50}, "age")
</script>
```