javascript 基本类型与引用类型

ECMAScirpt 变量有两种不同的数据类型：基本类型，引用类型。也有其他的叫法，比如原始类型和对象类型，拥有方法的类型和不能拥有方法的类型，还可以分为可变类型和不可变类型，其实这些叫法都是依据这两种的类型特点来命名的。

1.基本类型

基本的数据类型有：undefined，boolean，number，string，null.基本类型的访问是按值访问的，就是说你可以操作保存在变量中的实际的值。

基本类型有以下几个特点：

(1)基本类型的值是不可变得：

任何方法都无法改变一个基本类型的值，比如一个字符串：

javascript 代码

```
var name = 'jozo';
name.toUpperCase(); // 输出 'JOZO'
console.log(name); // 输出  'jozo'
```

会发现原始的 name 并未发生改变，而是调用了 toUpperCase()方法后返回的是一个新的字符串。

再来看个：javascript 代码

```
var person = 'jozo';
person.age = 22;
person.method = function(){//...};
console.log(person.age); // undefined
console.log(person.method); // undefined
```

通过上面代码可知，我们不能给基本类型添加属性和方法，再次说明基本类型是不可变得；

(2)基本类型的比较是值的比较：

只有在它们的值相等的时候它们才相等。

可能会这样比较：

```
var a = 1;
var b = true;
console.log(a == b);//true
```

它们不是相等吗？其实这是类型转换和 == 运算符的知识了，也就是说在用==比较两个不同类型的变量时会进行一些类型转换。像上面的比较先会把 true 转换为数字 1 再和数字 1 进行比较，结果就是 true 了。这是当比较的两个值的类型不同的时候==运算符会进行类型转换,但是当两个值的类型相同的时候，即使是==也相当于是===。

```
var a = 'jozo';
var b = 'jozo';
console.log(a === b);//true
```

（3）基本类型的变量是存放在栈区的（栈区指内存里的栈内存）

假如有以下几个基本类型的变量：

```
var name = 'jozo';
var city = 'guangzhou';
var age = 22;
```

那么它的存储结构如下图：

栈区	
name	jozo
city	guangzhou
age	22

栈区包括了 变量的标识符和变量的值。

2.引用类型

javascript 中除了上面的基本类型(number,string,boolean,null,undefined)之外就是引用类型了，也可以说是就是对象了。对象是属性和方法的集合。

也就是说引用类型可以拥有属性和方法，属性又可以包含基本类型和引用类型。来看看引用类型的一些特性：

(1)引用类型的值是可变的

我们可为引用类型添加属性和方法，也可以删除其属性和方法， javascript 代码如下：

```
var person = {};//创建个控对象 --引用类型
person.name = 'jozo';
person.age = 22;
person.sayName = function(){console.log(person.name);}
person.sayName();// 'jozo'
delete person.name; //删除 person 对象的 name 属性
person.sayName(); // undefined
```

上面代码说明引用类型可以拥有属性和方法，并且是可以动态改变的。

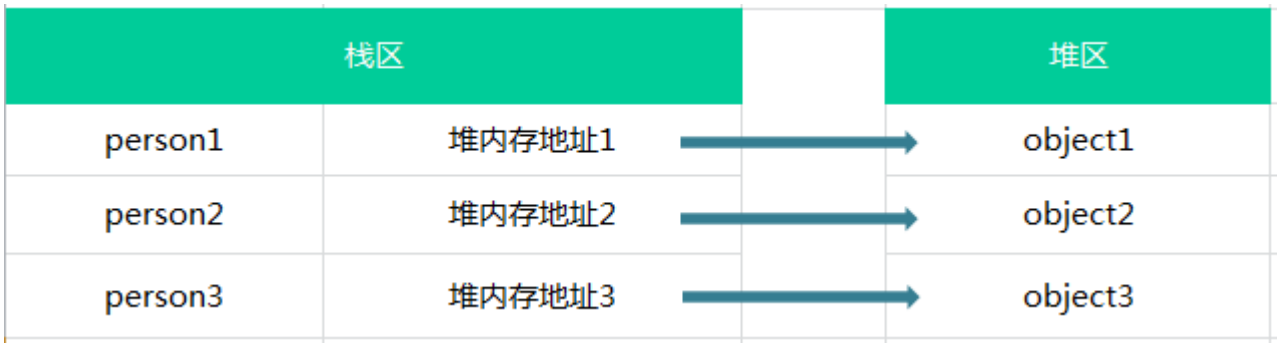
（2）引用类型的值是同时保存在栈内存和堆内存中的对象

javascript 和其他语言不同，其不允许直接访问内存中的位置，也就是说不能直接操作对象的内存空间，那我们操作什么呢？ 实际上，是操作对象的引用，所以引用类型的值是按引用访问的。准确地说，引用类型的存储需要内存的栈区和堆区（堆区是指内存里的堆内存）共同完成，栈区内存保存变量标识符和指向堆内存中该对象的指针，也可以说是该对象在堆内存的地址。

假如有以下几个对象：

```
var person1 = {name:'jozo'};
var person2 = {name:'xiaom'};
var person3 = {name:'xiaoq'};
```

则这三个对象的在内存中保存的情况如下图：



（3）引用类型的比较是引用的比较

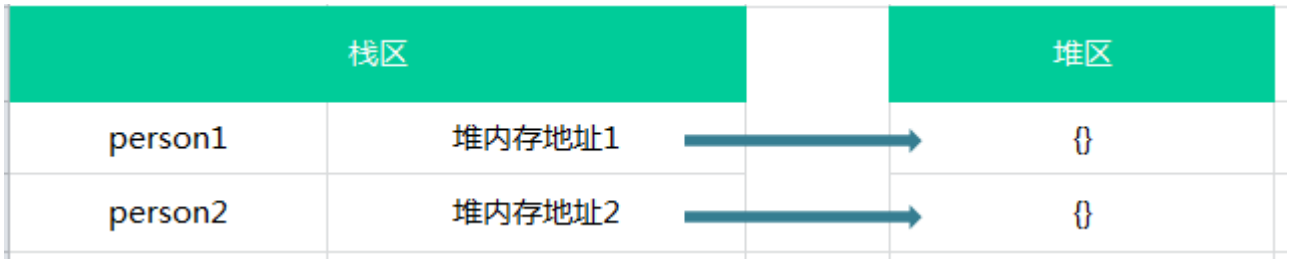
例:var person1 = '{}';

```
var person2 = '{}';
console.log(person1 == person2); // true
```

上面讲基本类型的比较的时候提到了当两个比较值的类型相同的时候，相当于是用 `===` ，所以输出是 `true` 了。若：

```
var person1 = {};
var person2 = {};
console.log(person1 == person2); // false
```

上面比较的是两个字符串，而下面比较的是两个对象，为什么长的一模一样的对象就不相等了呢？别忘了，引用类型时按引用访问的，换句话说就是比较两个对象的堆内存中的地址是否相同，那很明显，`person1` 和 `person2` 在堆内存中地址是不同的：



所以这两个是完全不同的对象，所以返回 `false`。

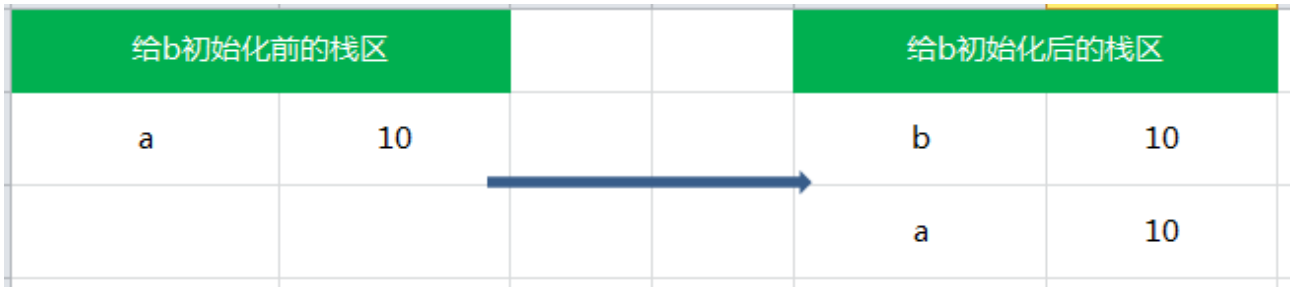
3.简单赋值

在从一个变量向另一个变量赋值基本类型时，会在该变量上创建一个新值，然后再把该值复制到为新变量分配的位置上：

例：

```
var a = 10;
var b = a;
a ++ ;
console.log(a); // 11
console.log(b); // 10
```

此时，`a` 中保存的值为 `10` ，当使用 `a` 来初始化 `b` 时，`b` 中保存的值也为 `10`，但 `b` 中的 `10` 与 `a` 中的是完全独立的，该值只是 `a` 中的值的一个副本，此后，这两个变量可以参加任何操作而相互不受影响。



也就是说基本类型在赋值操作后，两个变量是相互不受影响的。

4.对象引用

当从一个变量向另一个变量赋值引用类型的值时，同样也会将存储在变量中的对象的值复制一份放到为新变量分配的空间中。前面讲引用类型的时候提到，保存在变量中的是对象在堆内存中的地址，所以，与简单赋值不同，这个值的副本实际上是一个指针，而这个指针指向存储在堆内存的一个对象。那么赋值操作后，两个变量都保存了同一个对象地址，则这两个变量指向了同一个对象。因此，改变其中任何一个变量，都会相互影响：

例：

```
var a = {}; // a 保存了一个空对象的实例
var b = a;  // a 和 b 都指向了这个空对象
```

```
a.name = 'jozo';
console.log(a.name); // 'jozo'
console.log(b.name); // 'jozo'
b.age = 22;
console.log(b.age);// 22
console.log(a.age);// 22
console.log(a == b);// true
```

它们的关系如下图：

因此，引用类型的赋值其实是对象保存在栈区地址指针的赋值，因此两个变量指向同一个对象，任何的操作都会相互影响。