第四章 变量、作用域和内存问题

基本类型与引用数据类型

- 基本数据类型：Undefined Null Boolean Number String，但不可添加方法和属性，虽然不会报错，但是会undefined

```
var name = 'nick';//类型为String
name.age = 18;
alert(name.age);
```

- 引用数据类型：可能由多个值构成的对象，是可以为其添加属性和方法的，也可删除属性和方法

```
var person = new Object();
person.name = 'Nick';
alert(person.name);
```

- 复制变量的值
    - 基本数据类型的复制会产生一个完全独立的副本，而引用数据类型则是对同一个对象的不同的引用。

```
var obj1 = new Object();
 var obj2 = obj1;
 obj1.name = 'nick';
 alert(obj2.name);//nick
```

- 传递参数
    - 按值传递，即外部传入的数据只是数值相等，而引用不同，因此函数内部做的操作不影响外部定义的值；

```
function addTen(num){
    num += 10;
    return num
}
var count = 20;
var result = addTen(count);
alert(count);//20
alert(result);//30
```

- 按引用传递

```
function setName(obj){
obj.name = 'nick';
}
var person = new Object();
setName(person);
alert(person.name);//nick


function setName(obj){
obj.name = 'nick';
obj = new Object();//指向一个新的局部的对象(在函数执行完后会立即销毁)
obj.name = 'jack';//即使修改了传入对象的引用 且修改了属性，但原始的引用保持不变
}
var person = new Object();
setName(person);
alert(person.name);//因此输出的还是 nick
```

- 检测类型

1. typeof()函数： 能确定一个变量是String Number Boolean Undefined 但是值是对象或者null的话都会返回object

执行环境及作用域

- js是没有块级作用域的，如在for循环结束后，在for中定义的i仍然存在；但在java中一旦for循环结束，for中定义的i就会被销毁。
- 但在es6之后就有块级作用域了，此外还有全局作用域、函数作用域
- 声明变量
    - 使用var声明的变量将添加到最接近的环境中，比如函数内部最接近的就是函数的局部环境；如果初始化时没有使用var声明，则该变量将会自动添加到全局环境；