数据类型存储

2020年02月17日, 星期一　　09:32

1.变量声明

使用方便的标识符，用于引用计算机内存地址

变量声明指向一块内存空间，用于保存数据

变量赋值

向变量指向的内存空间中存放数据

一般来说，系统会划分出两种不同的内存空间

栈内存（stack）

堆内存（heap）

2.栈内存：

存储的值大小固定

由系统自动分配内存空间

空间小，运行效率高

堆内存：

存储的值大小不定，可动态调整

由程序员通过代码进行分配

空间大，运行效率相对较低

3.基本类型的变量是存放在**栈区**的

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

栈区	
name	jozo
city	guangzhou
age	22

基本类型的值是不可变的：

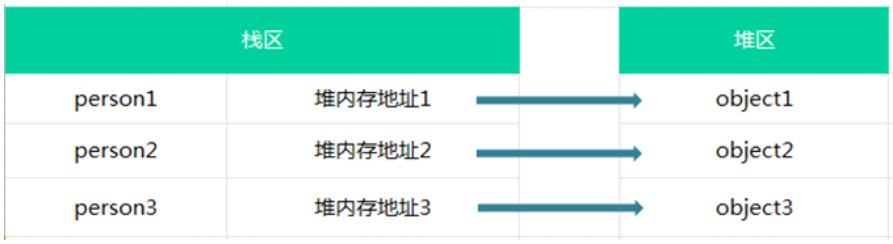
```
var name = 'jozo';
name.toUpperCase();
console.log(name); //jozo
```

toUpperCase();//字母转大写

name.toUpperCase();之后相当于开辟了一个新的栈区空间存放大写的JOZO，而name值不变仍然是小写的jozo

3.引用类型的值是**同时**保存在**栈内存**和**堆内存**中的对象

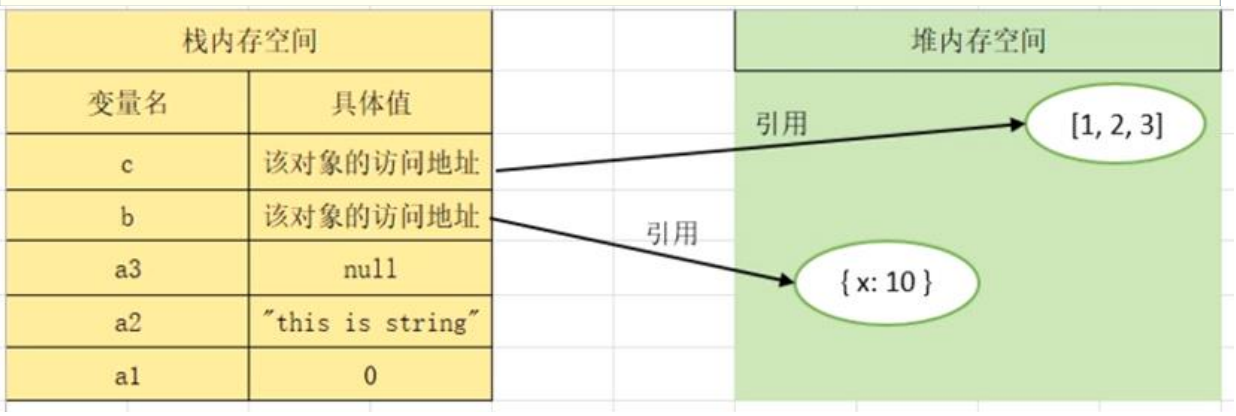
```
var person1 = {name: 'Lily'};
var person2 = {name: 'Lucy'};
var person3 = {name: 'Mary'};
```



引用类型的值可以改变吗？栈区中的值不改变，堆区中的值通过代码进行改变。

4.数据类型存储

```
var a1 = 0; // 栈内存
var a2 = "this is string" // 栈内存
var a3 = null; // 栈内存
var b = {x: 10}; // 变量b存在于栈中，{ x: 10 }作为对象存在于堆中
var c = [1, 2, 3]; // 变量c存在于栈中，[1, 2, 3]作为对象存在于堆中
```



5.基本类型与引用类型的区别

- 访问机制

  基本类型的值直接访问

  引用类型的值通过引用访问，不能直接访问

  - 首先，从栈中获取该对象的地址
  - 其次，再从堆内存中取得我们需要的数据

  因此，栈内存访问效率比较高

- 复制变量

  基本类型复制——相互独立互不影响

```
var a = 20;
var b = a;
b = 30;
console.log(a); // 20
```

这里拷贝的是值，拷贝完之后互不影响。

引用类型复制

● 引用类型复制

```
var a = { x: 10,y: 20};
var b = a;
b.x = 5;
console.log(a.x); // 5
```

河北师范大学软件学院

这里拷贝的是地址。

- 比较变量

  值类型是判断变量的**值**是否相等（值比较）

  引用类型是判断所指向的**内存空间（地址）**是否相同（引用比较）

```
var a = 20;
var b = 20;
a === b; //true
a = b;
a === b; //true
```

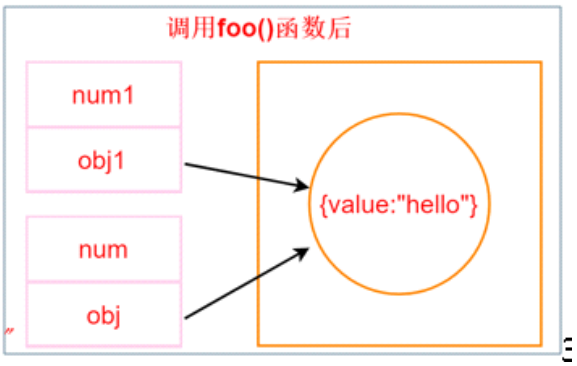
```
var a = {x: 10,y: 20};
var b = {x: 10,y: 20};
a === b; //false
a = b;
a === b; //true
```

- 参数传递

  ECMAScript中所有函数的参数都是**按值来传递**的

  - 基本类型值：把变量里的数据值传递给参数，之后参数和变量互不影响。
  - 引用类型值：把对象的引用（地址）值传递给参数，参数和对象都指向同一个对象，相互影响。

```
function foo(num, obj) {
    num = 20;
    obj.value = "hello world";
}
var num1 = 10;
var obj1 = {
    value: "hello"
};
foo(num1, obj1);
console.log(num1);//?10
console.log(obj1);//?"hello world!"
```



6.

```
var a = [1, 2, 3, 4];
var b = a;
b.shift();
console.log(a);
```

分析代码，在控制台输出结果为（）。

A [1, 2, 3, 4]

B [2, 3, 4]

shift（）把数组的第一个元素删除.

7.

```
var obj1 = {
  value: "111"
};
var obj2 = {
  value: "222"
};

function change(obj) {
  obj.value = "333";
  obj = obj2;
  return obj.value;
}
var foo = change(obj1);
console.log(foo); //?222
console.log(obj1.value); //? 333
```