JavaScript中变量的内存分配、拷贝和赋值

作者: xxx

**【摘要】**

在项目开发过程中，可能会遇到这种情况: 当我们给一个对象赋值后，修改新变量的数据会导致被赋值对象的数据发生改变，修改一个弹窗中的表格数据，明明点了取消，外面的数据却还是发生了变化。如果我们深刻理解了浅拷贝、深拷贝和赋值的区别后，这一系列问题将会迎刃而解。

**【正文】**

本篇文章将从内存层面上解析JavaScript中的浅拷贝、深拷贝和赋值问题。读了这篇文章，你将能够明白：

1. JavaScript的数据类型有哪些
2. 计算机对这两种数据类型是如何分配内存的
3. 浅拷贝、深拷贝和赋值三者究竟有什么区别

# 、数据类型：简单类型和复杂类型

## 简单类型

简单类型在存储变量时，存储的是值本身，因此也叫值类型。

包含：string，number，boolean，undefined，null等

## 复杂类型

复杂类型在存储变量时，存储的仅仅是一个地址（引用），因此也叫引用类型。

包含：Object，Array，Date等

# 、栈和堆

 值得注意的是JavaScript中并没有栈和堆的概念，在此引入是为了了解代码执行的方式。

## 栈

简单数据类型存放到“栈”中。这种变量的内存由操作系统自动分配释放。例如：函数的参数值、局部变量的值等。

## 堆

复杂数据类型存放到“堆”中。这种变量的内存一般由程序员分配释放，否则由垃圾回收机制回收。

## 图解“栈”和“堆”



# 、两种数据类型的内存分配

## 定义变量



## 图解内存存储



## 总结

### 简单数据类型存放到“栈”中，存的是数值

### 复杂数据类型存放到“堆”中，它首先会在“栈”中存放一个十六进制的地址，然后该地址会指向“堆”中的数据，形成一种“间接引用”的关系。

# 、浅拷贝、深拷贝和赋值

## 浅拷贝





函数调用相当于进行了一次浅拷贝，简单数据类型的“值”会被重新复制一份，修改简单数据类型不会影响“数据源”的值；

复杂数据类型的“十六进制地址”会被重新复制一份，因此它们共用同一个“数据源”，修改这个值将意味着“数据源”的值被修改了。

## 深拷贝





JSON方法内部进行了深拷贝处理，简单数据类型的“值”会被重新复制一份，修改简单数据类型不会影响“数据源”的值；

复杂数据类型会生成一个新的“十六进制地址”，同时在“堆”中被重新复制成一份新数据，因此它们并没有共用同一个“数据源”，修改这个值就不会改变“数据源”的值。

## 赋值

赋值与浅拷贝基本一样，区别在于：浅拷贝会在内存中重新开辟新的空间，简单数据类型会拷贝“值”，复杂数据类型会拷贝“十六进制地址”。

# 、一句话总结

如果一个对象被赋值了，改变新对象的“基本数据类型”或“复杂数据类型”都会影响原对象；

如果一个对象被浅拷贝了，改变新对象的“基本数据类型”，不会影响原对象，改变它的“复杂数据类型”，会影响原对象；

如果一个对象被深拷贝了，改变新对象的“基本数据类型”或“复杂数据类型”都不会影响原对象。