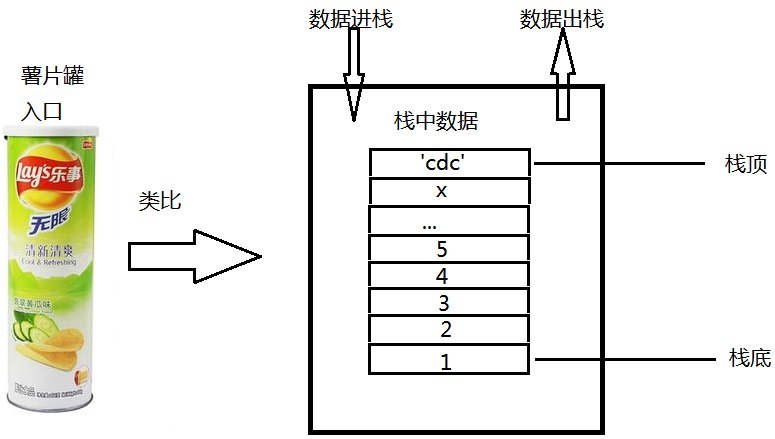
对于前端开发来说，内存空间的概念并没有像其他语言那样被重视，平时涉及到内存相关的情况比较少，通常就是内存溢出之类的。但这并不代表内存空间这块知识并不重要，相反的是，想要对于javascript有一个深刻的认知的话，必须对内存空间有一个清晰的认知！

内存空间中有三种数据结构，分别是堆(heap)，栈(stack)与队列(queue)。

**栈数据结构**

栈结构具有【先进后出，后进先出】的特点。比如在javascript的执行上下文中就是使用这种栈数据结构，来协调执行顺序(关于执行上下文会在后面文章讲述)。下面我们类比薯片罐加下图描述栈的存取方式：



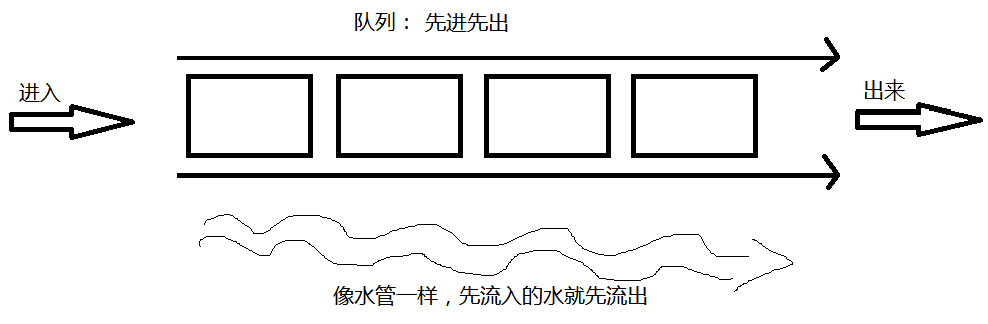
栈数据的存取跟薯片罐类似。放薯片的时候（存数据），最先放进去的薯片位于罐子底部，最后放的薯片位于罐子的顶部。吃薯片的时候（取数据），最先被拿出来的是最后放进去的那一片。如果我们想要拿最后的那片薯片，必须把前面的薯片都先取出来。这就是【先进后出，后进先出】的特点。

**堆数据结构**

堆数据结构比较类似我们去超市商场用的储物柜。什么意思呢？柜子虽然整整齐齐的摆放在一起，但是我们并不用去按顺序的打开，我们只需要记住自己储物柜的编号，便可以用钥匙打开。类似JSON格式的数据，采用key-val键值对无序的方式。

**队列数据结构**

队列是一种先进先出的数据结构。类似水管流水一样，先进去的水就先流出来，如下图：



经过上面的介绍，我们大致了解内存空间的数据结构，它们在实际应用有以下几种场景。

**变量的存放**

首先我们应该知道内存中有栈和堆，那么变量应该存放在哪里呢，堆？栈？实际上他们存储的介质都是内存，只是形式不一样分为堆内存、栈内存。

1. 基本类型 --> 保存在栈内存中，因为这些类型在内存中分别占有固定大小的空间，通过按值来访问。基本类型一共有6种：Undefined、Null、Boolean、Number 、String和Symbol
2. 引用类型 --> 如数组Array、对象Object，保存在堆内存中，因为这种值的大小不固定，因此不能把它们保存到栈内存中，但内存地址值大小的固定的，因此保存在堆内存中。在栈内存中存放的只是该对象的访问地址。当查询引用类型的变量时， 先从栈中读取内存地址， 然后再通过地址找到堆中的值。这种方式，我们称为按引用访问。

举个例子，在执行上下文中，会生成一个变量对象，变量对象详解会在后面文章分析，在本文，我们只需要把它理解为一个存储变量的对象。

文字描述总是素然无味，图片才比较有画面感，看图与代码：

**实例一：**

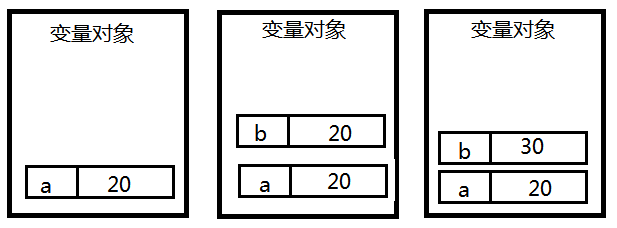
var a = 20

var b = a

b = 30

此时a的值是什么呢？

在内存中，变化如下图：



在上述案例中，a的值仍然是20。因为a与b的值都属于基本类型，他们之间是按值访问的，因此var b = a时，只是把a的值20直接赋予b，并不存在引用关系。

**实例二：**

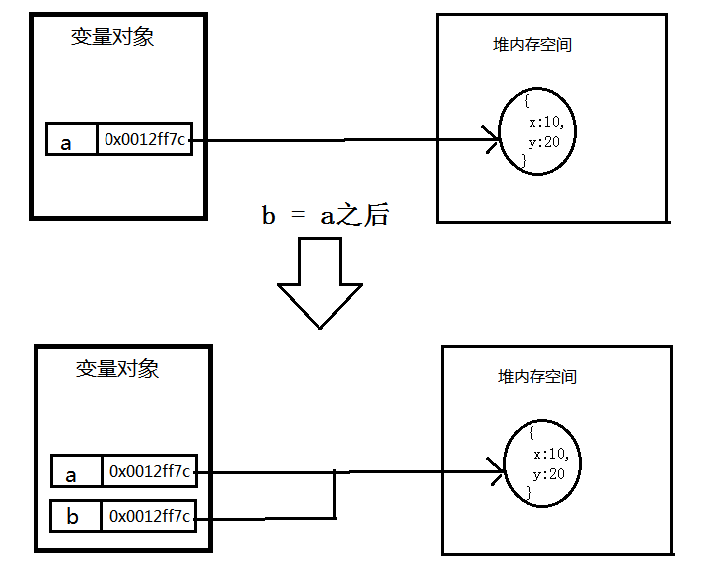
var a = { x:10, y:20}

var b = a

b.x = 15

请问此时a.x的值是多少？

在内存中，变化如下图：



上述示例，a.x的值是15。由于a是josn对象，属于引用数据类型数据，所以a保存的值，实际上是json对象在堆内存中的地址0x0012ff7c，通过内存地址找到对应的json数据，类似我们通过门牌号（内存地址）找到对应的房子（内存块存储的数据）。在var b = a中，实际上是把内存地址值赋值给了b，所以当b.x = 15时，是通过b的内存地址，找到对应的数据，并修改。此时，再访问a.x时，也是通过内存地址去访问到修改过后的值了。

function test(person) {

person.age = 26

person = {

name: 'hzj',

age: 18

}

return person

}

const p1 = {

name: 'fyq',

age: 19

}

const p2 = test(p1)

console.log(p1) // p1打印什么？为什么？

console.log(p2) // p2打印什么？为什么？

在函数传参的时候传递的是对象在堆中的内存地址值，test函数中的实参person是p1对象的内存地址，通过调用person.age = 26确实改变了p1的值，但随后person变成了另一块内存空间的地址，并且在最后将这另外一份内存空间的地址返回，赋给了p2。