简单类型和对象的区别

简单类型：number、string、boolean、undefined、null、symbol

复杂类型（引用类型）：object

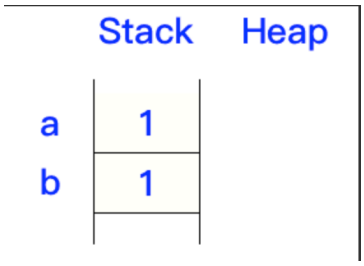
要了解两者的区别，首先要知道内存。当打开浏览器的时候，浏览器会占用一些内存，浏览器会把这些内存分布给网页，网页再将内存分配给页面渲染器、网络模块、浏览器外壳和 JS 引擎（V8 引擎）等。学习 JS 主要是研究 JS 引擎部分。JS 引擎将得到的内存分成代码区和数据区。要了解简单类型和对象的区别，就研究数据区。在数据区分为栈内存（Stack）和堆内存（Heap）。简单类型的数据直接存在 Stack 里。复杂类型的数据是把 Heap 地址存在 Stack 里。

我们通过一些例子来了解一下：

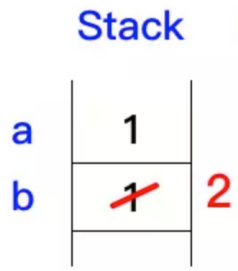
```
var a = 1
var b = a
b = 2
```

问 a 的值。

当执行到 `var b=a` 时，内存中可以表示为：



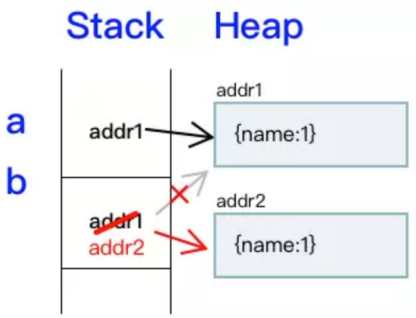
此时执行 `b=2` 直接将 2 的值放在 b 的 stack 中，如：



b 的改变并没有影响到 a，a 的结果还是 1。

```
var a = {name: 'a'}
var b = a
b = {name: 'b'}
```

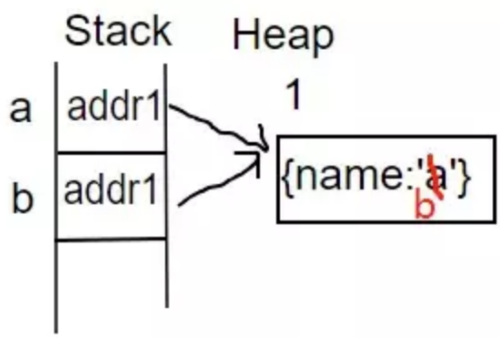
问 a.name 的值。



从上图可知对于复杂类型，在栈（Stack）上存储的是一个地址，在堆（Heap）上存储的是数据。地址有指向关系。运行到 `var b = a` 时，是将 a 中的 addr1 地址复制放到 b 的 stack 中。当运行 `b = {name: 'b'}` 时是在 heap 中重新开出一块放置 `{name: 'b'}` 且地址为 2，并且 b 的地址更改为 addr2。这个时候 b 指向 addr1 的作废，重新指向 2。所以这个时候 a 还是原来的。即 a.name 的结果是"a"。

```
var a = {name: 'a'}
var b = a
b.name = 'b'
```

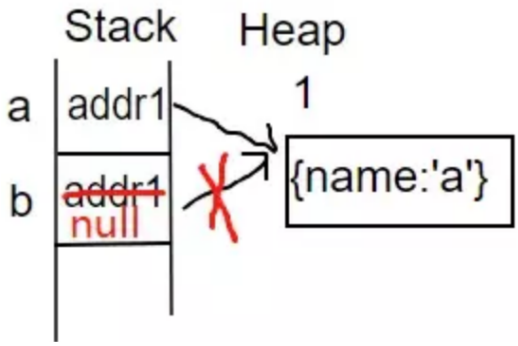
问 a.name 的值。



当运行至 var b = a 时 a 和 b 的指向相同，此时运行 `b.name = 'b'`，更改了 heap 中地址 1 中的信息，如图所示。所以这个时候 a.name 的结果是"b"。

```
var a = {name: 'a'}
var b = a
b = null
```

问 a 的值。



当运行至 `var b = a` 时 a 和 b 的指向相同，这个时候运行 `b = null`（这是一个普通类型，直接将 null 的值在 stack 中给向 b），所以导致 b 在 heap 中的指向消失并没有改变 heap 内存中数据。所以 a 的结果还是本身，即`{name: 'a'}`。

关于深拷贝和浅拷贝

```
var a = 1
var b = a
b = 2 //这个时候改变 b
```

`a 完全不受 b 的影响`

`那么我们就说这是一个深复制`

对于简单类型的数据来说，赋值就是深拷贝。 对于复杂类型的数据（对象）来说，才要区分浅拷贝和深拷贝。

这是一个浅拷贝的例子，因为我们对 b 操作后，a 也变了。

```
var a = {name: 'frank'}
var b = a
b.name = 'b'
a.name === 'b' // true
```