从 4 个面试题了解「浏览器的垃圾回收」

Marica 大迁世界 2020-11-06

> 浏览器垃圾回收一直是前端面试常考的部分，我一直不太理解。最近深入学习了一下，争取一篇文章说清楚。

我们首先带着这 4 个问题，来了解浏览器垃圾回收的过程，后面会逐一解答：

1. 浏览器怎么进行垃圾回收?
2. 浏览器中不同类型变量的内存都是何时释放?
3. 哪些情况会导致内存泄露？如何避免?
4. `weakMap` `weakSet` 和 `Map` `Set` 有什么区别?

ok, let's go!

什么是垃圾数据?

生活中你买了一瓶可乐，喝完之后可乐瓶就变成了垃圾，应该被回收处理。

同样地，我们在写 js 代码的时候，会频繁地操作数据。

在一些数据不被需要的时候，它就是垃圾数据，垃圾数据占用的内存就应该被回收。

变量的生命周期

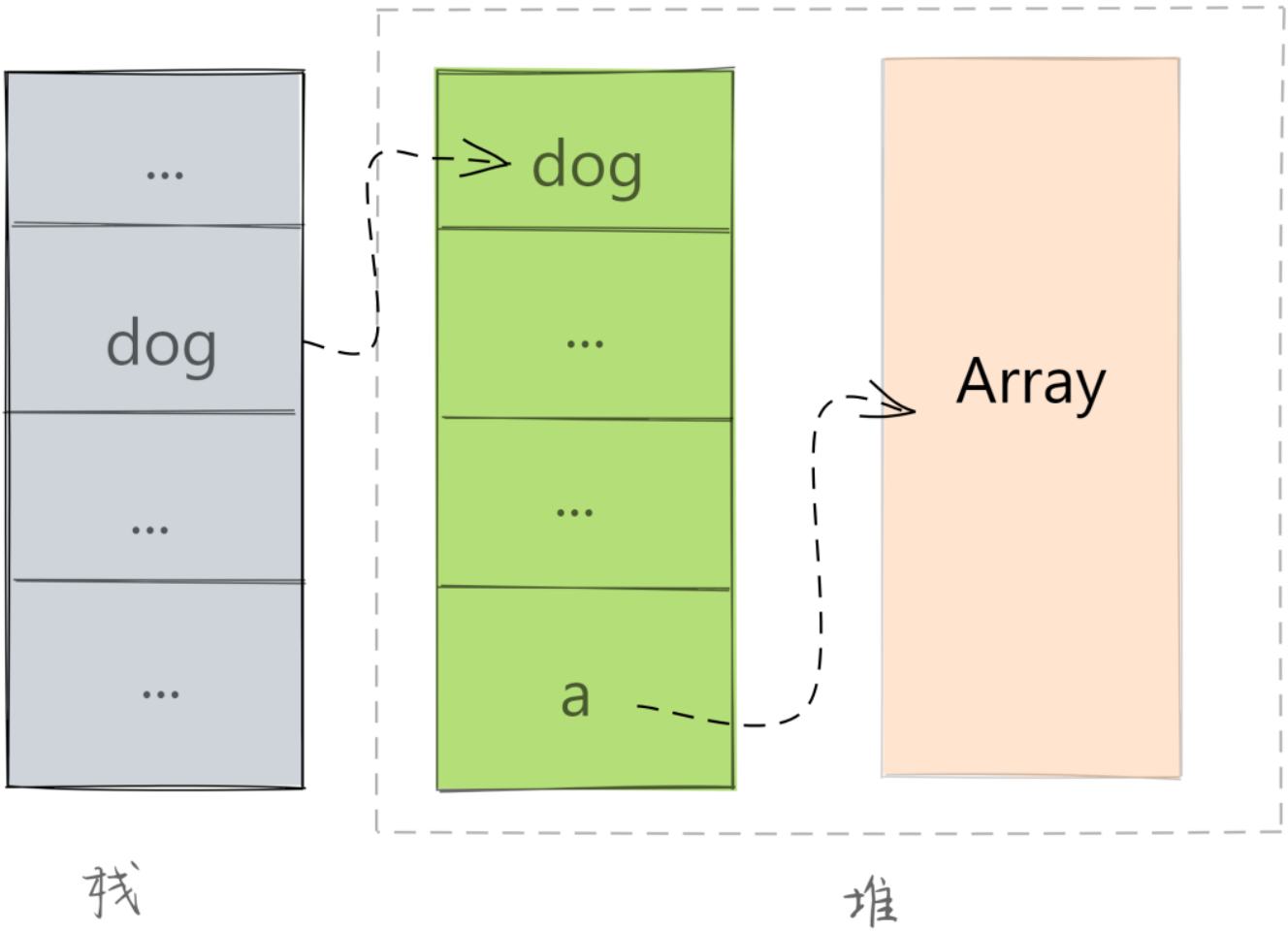
比如这么一段代码：

```
let dog = new Object()let dog.a = new Array(1)
```

当 JavaScript 执行这段代码的时候，

会先在全局作用域中添加一个 `dog` 属性，并在堆中创建了一个空对象，将该对象的地址指向了 `dog` 。

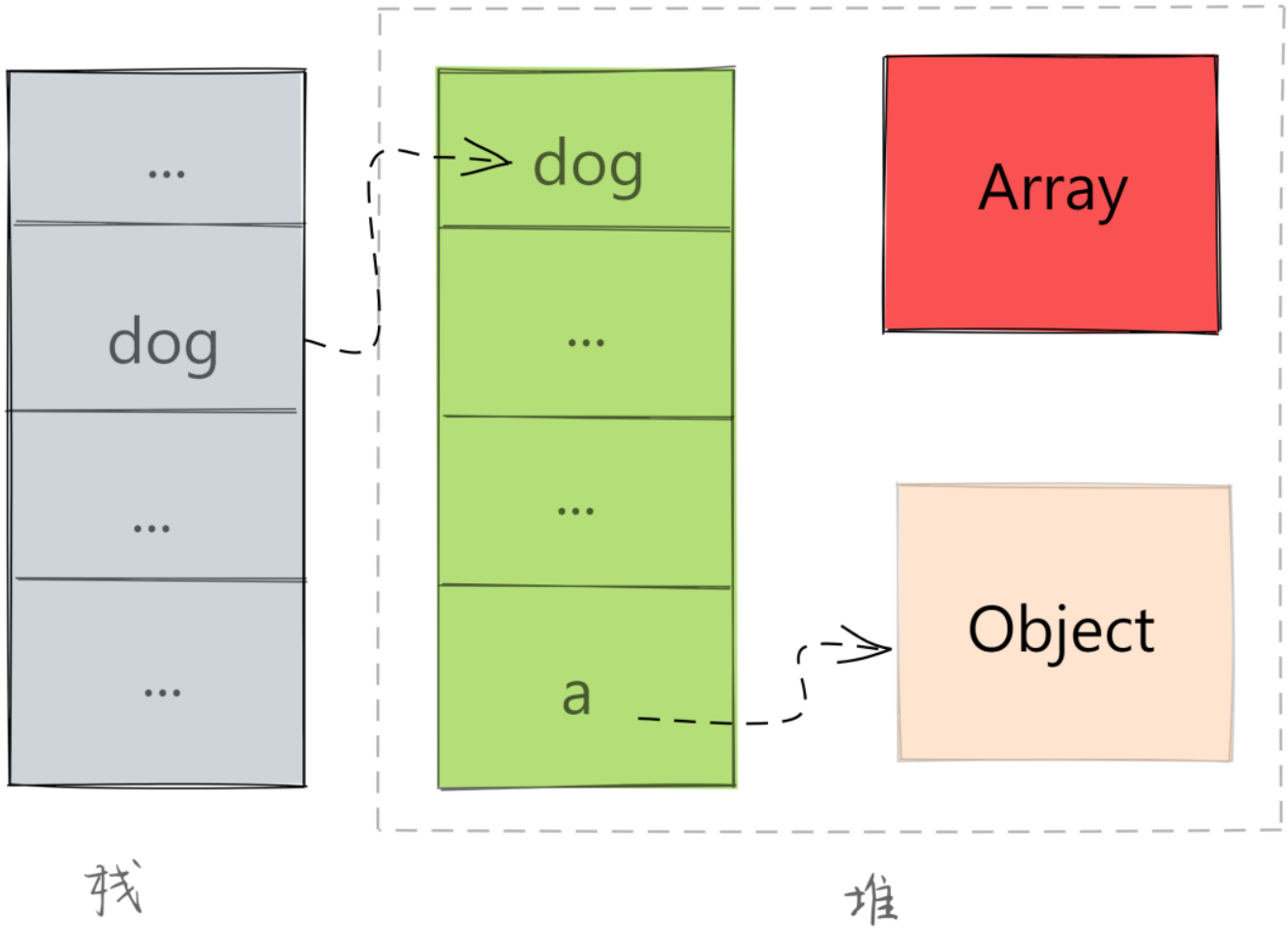
随后又创建一个大小为 1 的数组，并将属性地址指向了 `dog.a` 。此时的内存布局图如下所示：



如果此时，我将另外一个对象赋给了 `a` 属性，代码如下所示：

```
dog.a = new Object()复制代码
```

此时的内存布局图：



`a` 的指向改变了， 此时堆中的数组对象就成为了不被使用的数据，专业名词叫「不可达」的数据。

这就是需要回收的垃圾数据。

垃圾回收算法

可以将这个过程想象成从根溢出一个巨大的油漆桶，它从一个根节点出发将可到达的对象标记染色， 然后移除未标记的。

第一步：标记空间中「可达」值。

V8 采用的是可达性 (reachability) 算法来判断堆中的对象应不应该被回收。

这个算法的思路是这样的：

- 从根节点（Root）出发，遍历所有的对象。

- 可以遍历到的对象，是可达的（reachable）。
- 没有被遍历到的对象，不可达的（unreachable）。

在浏览器环境下，根节点有很多，主要包括这几种：

- 全局变量 `window`，位于每个 `iframe` 中
- 文档 `DOM` 树
- 存放在栈上的变量
- ...

这些根节点不是垃圾，不可能被回收。

第二步：回收「不可达」的值所占据的内存。

在所有的标记完成之后，统一清理内存中所有不可达的对象。

第三步，做内存整理。

- 在频繁回收对象后，内存中就会存在大量不连续空间，专业名词叫「内存碎片」。
- 当内存中出现了大量的内存碎片，如果需要分配较大的连续内存时，就有可能出现内存不足的情况。
- 所以最后一步是整理内存碎片。(但这步其实是可选的，因为有的垃圾回收器不会产生内存碎片，比如接下来我们要介绍的副垃圾回收器。)

什么时候垃圾回收?

浏览器进行垃圾回收的时候，会暂停 JavaScript 脚本，等垃圾回收完毕再继续执行。

对于普通应用这样没什么问题，但对于 JS 游戏、动画对连贯性要求比较高的应用，如果暂停时间很长就会造成页面卡顿。

这就是我们接下来谈的关于垃圾回收的问题：什么时候进行垃圾回收，可以避免长时间暂停。

分代收集

浏览器将数据分为两种，一种是「临时」对象，一种是「长久」对象。

- 临时对象：
  - 大部分对象在内存中存活的时间很短。
  - 比如函数内部声明的变量，或者块级作用域中的变量。当函数或者代码块执行结束时，作用域中定义的变量就会被销毁。
  - 这类对象很快就变得不可访问，应该快点回收。
- 长久对象：
  - 生命周期很长的对象，比如全局的 `window、DOM、Web API` 等等。
  - 这类对象可以慢点回收。

这两种对象对应不同的回收策略，所以，V8 把堆分为新生代和老生代两个区域， 新生代中存放临时对象，老生代中存放持久对象。

并且让副垃圾回收器、主垃圾回收器，分别负责新生代、老生代的垃圾回收。

这样就可以实现高效的垃圾回收啦。

一般来说，面试回答到这就够了。如果想和面试官深入交流，可以继续聊聊两个垃圾回收器。

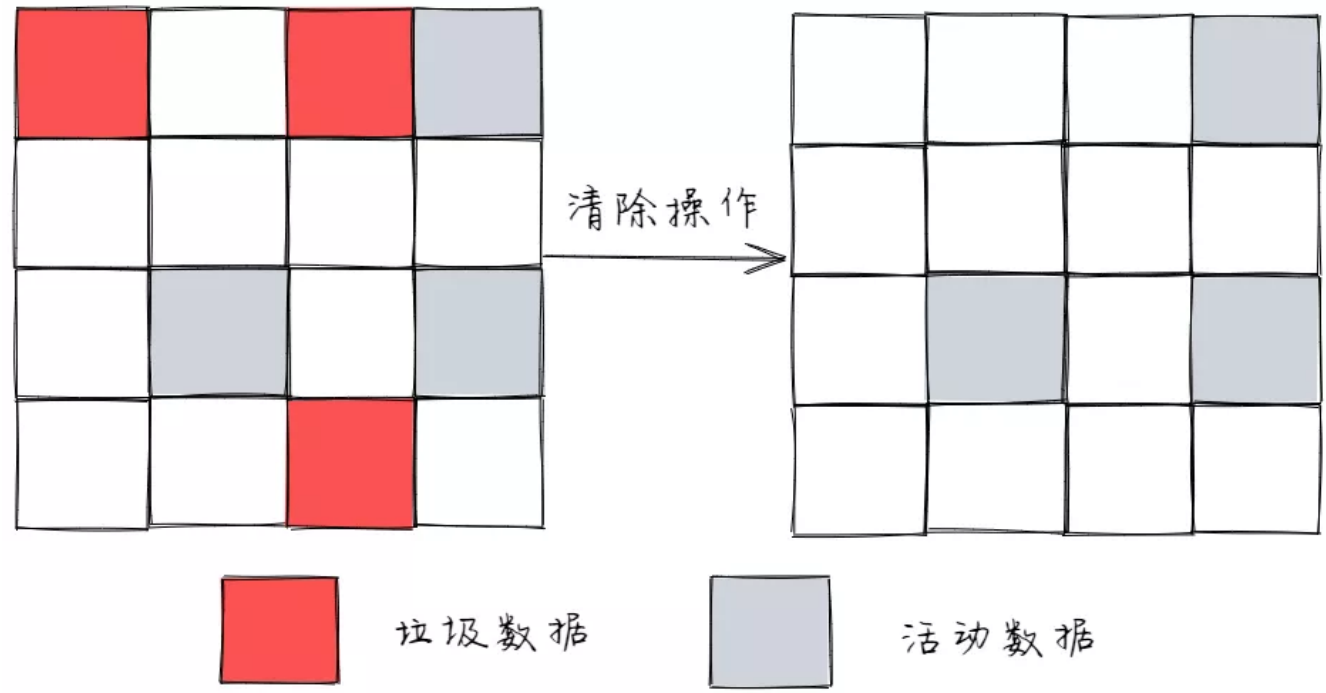
主垃圾回收器

负责老生代的垃圾回收，有两个特点：

1. 对象占用空间大。
2. 对象存活时间长。

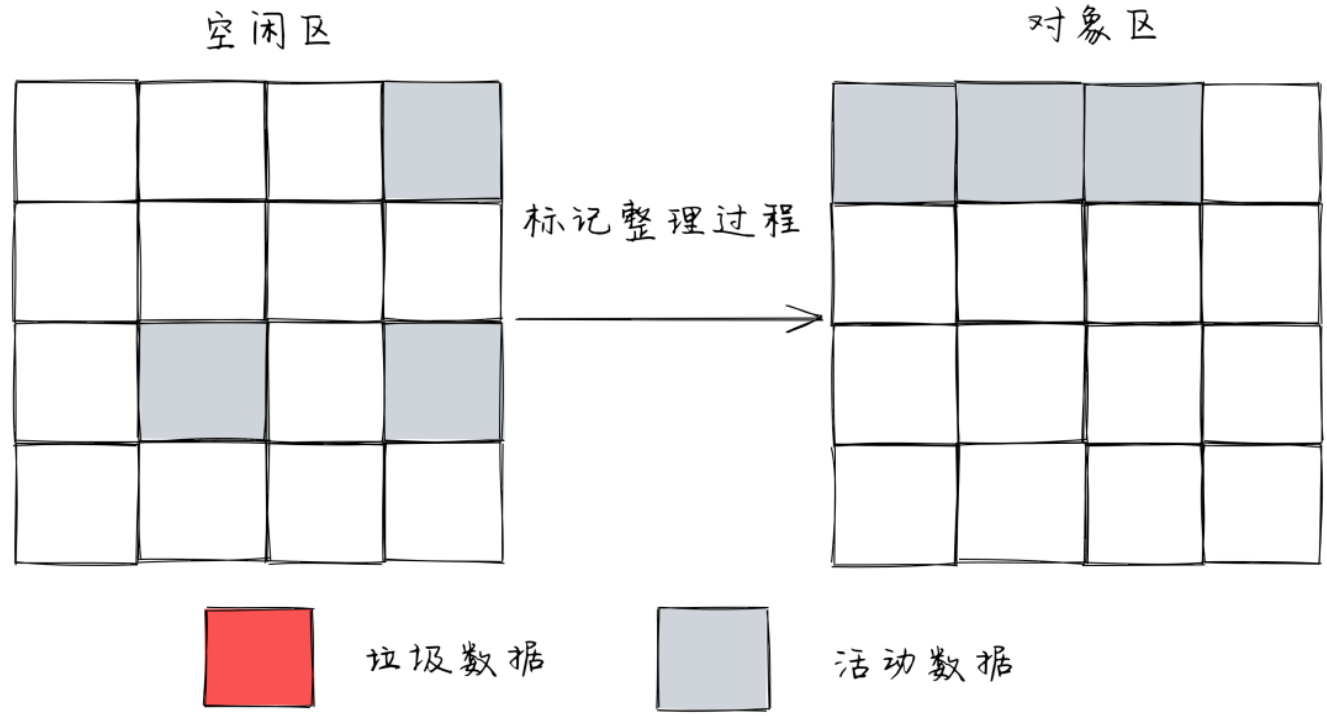
它使用「标记-清除」的算法执行垃圾回收。

1. 首先是标记。
   - 从一组根元素开始，递归遍历这组根元素。
   - 在这个遍历过程中，能到达的元素称为活动对象，没有到达的元素就可以判断为垃圾数据。
2. 然后是垃圾清除。



直接将标记为垃圾的数据清理掉。

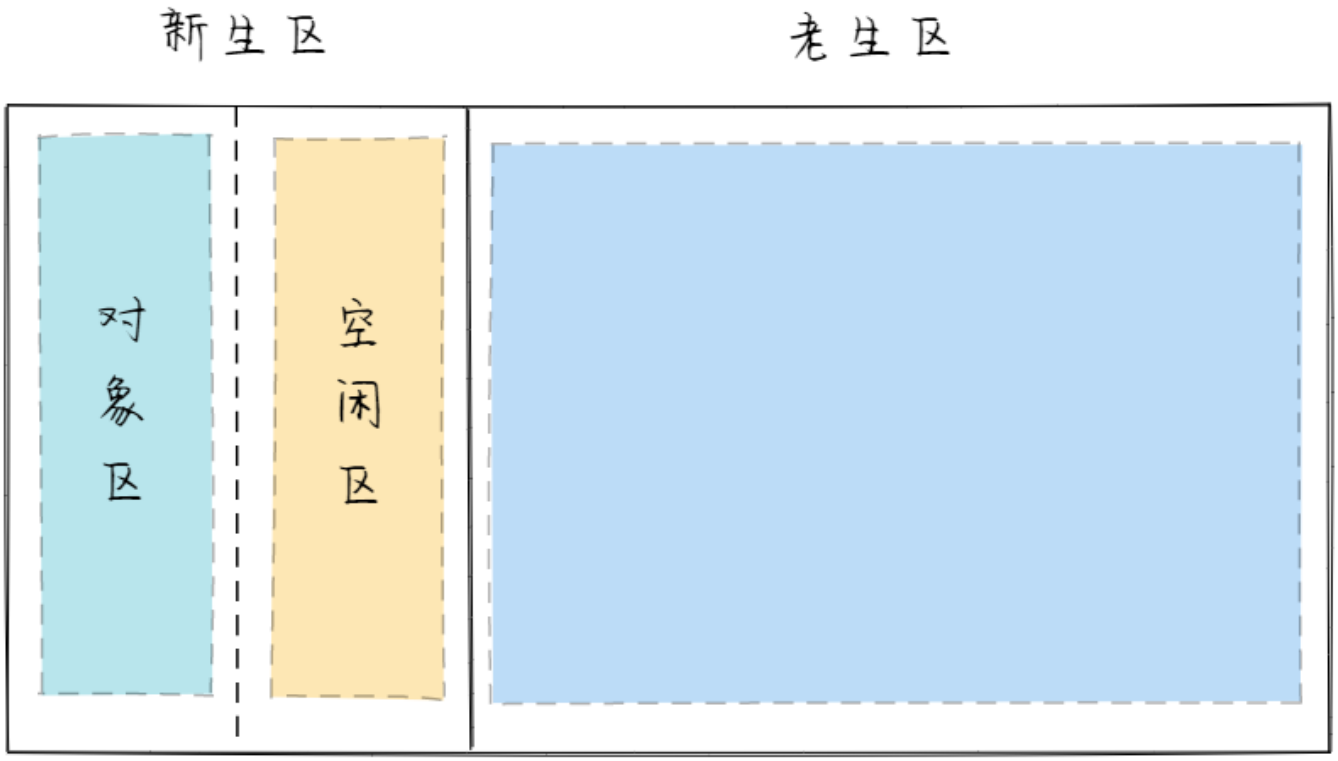
3. 多次标记-清除后，会产生大量不连续的内存碎片，需要进行内存整理。



副垃圾回收器

负责新生代的垃圾回收，通常只支持 1~8 M 的容量。

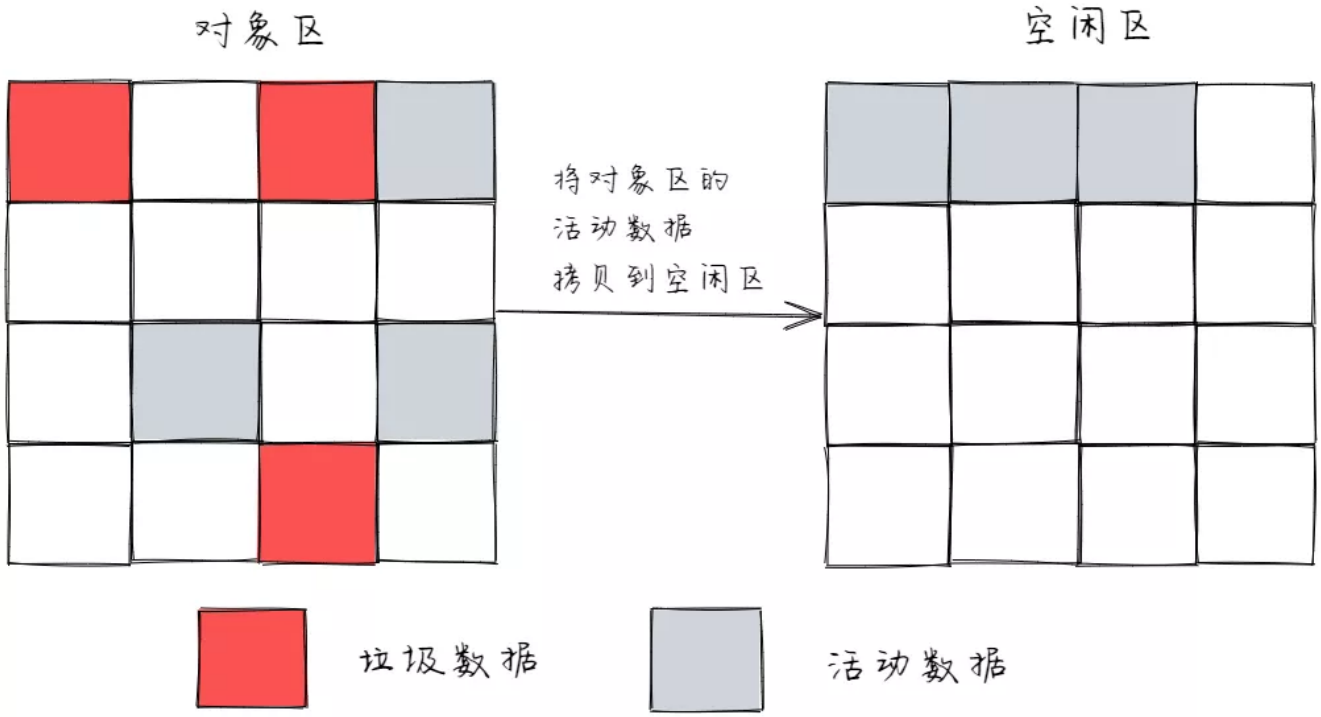
新生代被分为两个区域：一般是对象区域，一半是空闲区域。



V8 的堆空间

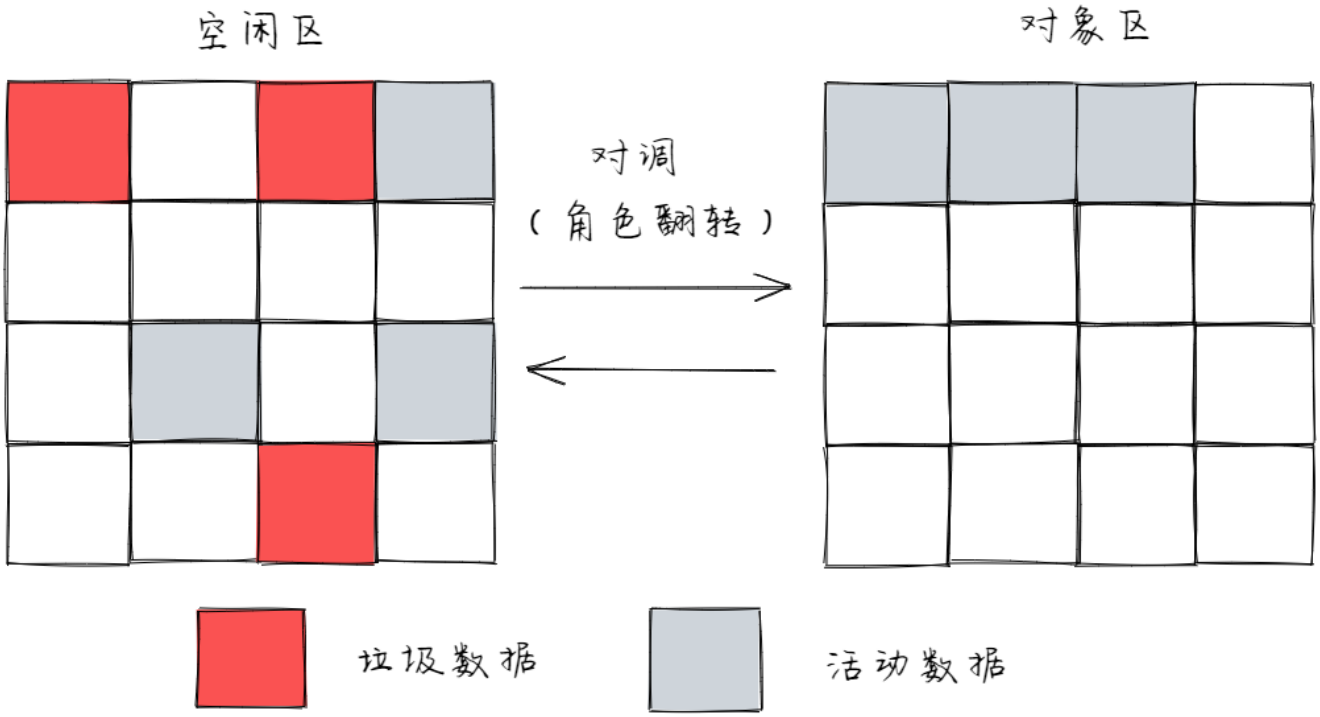
新加入的对象都被放入对象区域，等对象区域快满的时候，会执行一次垃圾清理。

1. 先给对象区域所有垃圾做标记。
2. 标记完成后，存活的对象被复制到空闲区域，并且将他们有序的排列一遍。



这就回到我们前面留下的问题 -- 副垃圾回收器没有碎片整理。因为空闲区域里此时是有序的，没有碎片，也就不需要整理了。

3. 复制完成后，对象区域会和空闲区域进行对调。将空闲区域中存活的对象放入对象区域里。



这样，就完成了垃圾回收。

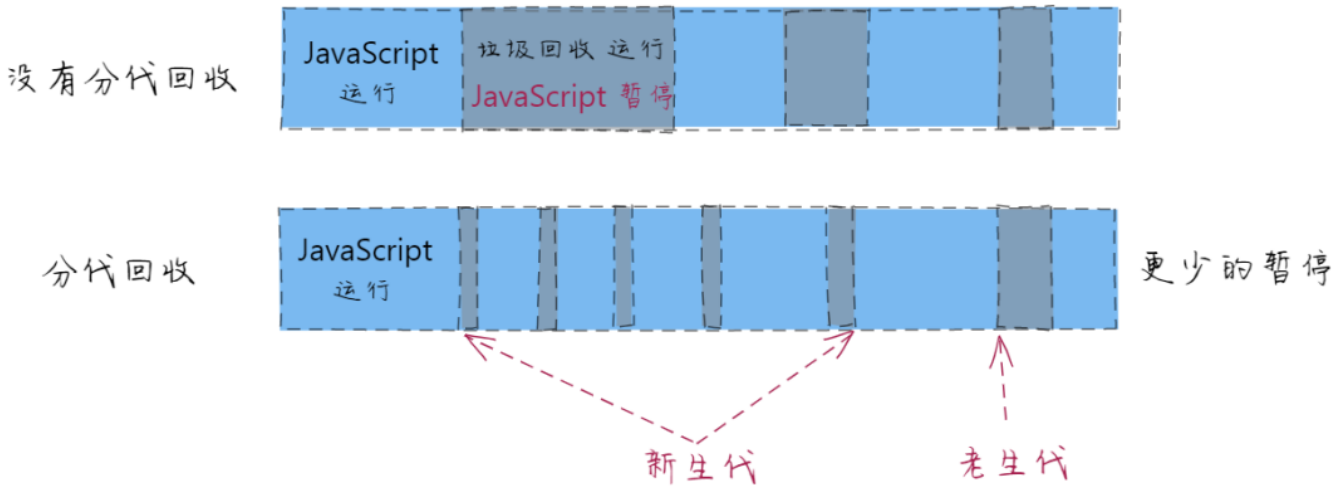
因为副垃圾回收器操作比较频繁，所以为了执行效率，一般新生区的空间会被设置得比较小。

一旦检测到空间装满了，就执行垃圾回收。

分代收集

一句话总结分代回收就是：将堆分为新生代与老生代，多回收新生代，少回收老生代。

这样就减少了每次需遍历的对象，从而减少每次垃圾回收的耗时。



增量收集

如果脚本中有许多对象，引擎一次性遍历整个对象，会造成一个长时间暂停。

所以引擎将垃圾收集工作分成更小的块，每次处理一部分，多次处理。

这样就解决了长时间停顿的问题。

闲时收集

垃圾收集器只会在 CPU 空闲时尝试运行，以减少可能对代码执行的影响。

面试题1：浏览器怎么进行垃圾回收?

从三个点来回答什么是垃圾、如何捡垃圾、什么时候捡垃圾。

1. 什么是垃圾
    - 不再需要，即为垃圾
    - 全局变量随时可能用到，所以一定不是垃圾
2. 如何捡垃圾（遍历算法）
    - 标记空间中「可达」值。
        - 从根节点（Root）出发，遍历所有的对象。
        - 可以遍历到的对象，是可达的（reachable）。
        - 没有被遍历到的对象，不可达的（unreachable）
    - 回收「不可达」的值所占据的内存。

- 做内存整理。

3. 什么时候捡垃圾

- 前端有其特殊性，垃圾回收的时候会造成页面卡顿。
- 分代收集、增量收集、闲时收集。

面试题2：浏览器中不同类型变量的内存都是何时释放?

Javascritp 中类型：值类型，引用类型。

- 引用类型
  - 在没有引用之后，通过 V8 自动回收。
- 值类型
  - 如果处于闭包的情况下，要等闭包没有引用才会被 V8 回收。
  - 非闭包的情况下，等待 V8 的新生代切换的时候回收。

面试题3：哪些情况会导致内存泄露？如何避免?

> 内存泄露是指你「用不到」（访问不到）的变量，依然占居着内存空间，不能被再次利用起来。

以 Vue 为例，通常有这些情况：

- 监听在 `window/body` 等事件没有解绑
- 绑在 `EventBus` 的事件没有解绑
- `Vuex` 的 `$store`， `watch` 了之后没有 `unwatch`
- 使用第三方库创建，没有调用正确的销毁函数

解决办法： `beforeDestroy` 中及时销毁

- 绑定了 `DOM/BOM` 对象中的事件 `addEventListener`， `removeEventListener`。
- 观察者模式 `$on`， `$off` 处理。

- 如果组件中使用了定时器，应销毁处理。
- 如果在 `mounted/created` 钩子中使用了第三方库初始化，对应的销毁。
- 使用弱引用 `weakMap` 、 `weakSet` 。

闭包会导致内存泄露吗?

顺便说一个我在了解垃圾回收之前对闭包的误解。

闭包会导致内存泄露吗？正确的答案是不会。

内存泄露是指你「用不到」（访问不到）的变量，依然占居着内存空间，不能被再次利用起来。

~~闭包里面的变量就是我们需要的变量，不能说是内存泄露~~。

> 这个误解是如何来的？因为 IE。IE 有 bug，IE 在我们使用完闭包之后，依然回收不了闭包里面引用的变量。这是 IE 的问题，不是闭包的问题。参考这篇文章

面试题4：weakMap weakSet 和 Map Set 有什么区别?

在 ES6 中为我们新增了两个数据结构 WeakMap、WeakSet，就是为了解决内存泄漏的问题。

它的键名所引用的对象都是弱引用，就是垃圾回收机制遍历的时候不考虑该引用。

只要所引用的对象的其他引用都被清除，垃圾回收机制就会释放该对象所占用的内存。

也就是说，一旦不再需要，WeakMap 里面的键名对象和所对应的键值对会自动消失，不用手动删除引用。

更全面的介绍可以看这里：第 4 题：介绍下 Set、Map、WeakSet 和 WeakMap 的区别

总结

现在我们简单了解了浏览器的垃圾回收机制，还记得最初的 4 个问题吗?

1. 浏览器怎么进行垃圾回收?

答题思路：什么是垃圾、怎么收垃圾、什么时候收垃圾。

2. 浏览器中不同类型变量的内存都是何时释放?

答题思路：分为值类型、引用类型。

3. 哪些情况会导致内存泄露？如何避免?

答题思路：内存泄露是指你「用不到」（访问不到）的变量，依然占居着内存空间，不能被再次利用起来。

4. `weakMap` `weakSet` 和 `Map` `Set` 有什么区别?

答题思路： `WeakMap` 、 `WeakSet` 弱引用，解决了内存泄露问题。

分享一套 <React Hooks重构去哪网购票>视频教程，如果你对 React Hooks 感兴趣，点 在看 并在后台回复 "Hooks" 即可获得。

❤️爱心三连击

1.看到这里了就点个在看支持下吧，你的「点赞，在看」是我创作的动力。

2.关注公众号 大迁世界 ，回复「1」加入前端进阶交流群！「在这里有好多 前端 开发者，会讨论 前端 知识，互相学习」！

3.也可添加微信【qq449245884】，一起成长。

喜欢此内容的人还喜欢

HTML+CSS 回忆碎片悬停效果，帮忙点赞呗！

大迁世界

真人实测|原相机下的抖音化妆术，经得起考验吗?

十点种草

牛奶早上喝好还是晚上好？4 个最佳时间的真相，颠覆了

腾讯医典