`

图解算法一

内存中的链表和数组

结论：

- JAVA不可自动增加内存空间,JAVASCRIPT可以自动增加内存空间
- 数组的存储空间必须连续，链表无要求
- 数组读取快O(1)，插入慢O(n)（链表相反）

比如：JS中声名数组后可以自由添加或删除数组，可以随意通过`.length`改变数组的长度，Java中不可。

数组的特性

- 数组名称其实就是连续内存地址的首地址
- 数组在内存中保存必须是相连的，因此数组新增时要重新分配一块连续的内存空间

首先：

- JavaScript可以预先申请内存空间或者不声明空间

> 比如只有2个单元数组，但可以预先声明10个位置 `new Array(10)`
>
> > 比如不知道会需要多大空间，不预先声名空间 `new Array()` or`[]`
> >
> > > 或者直接实例化`new Array(a,b,c)`

- 缺点：

> > > 额外请求的空间可能用不上造成内存浪费
> > >
> > > > 后续数组添加超过预申请的10个后，还是要重新转移内存地址

链表的特性

- 链表可以存储在内存中的任何地方，链表中的每个元素都存储了下一个元素的地址
- 这种地址的链式存储使一系列保存在随机内存地址中的数组元素串在一起。

首先：

- 添加新元素很简单，只需放入内存，然后把它的地址存储到前一个元素中去即可。

> 不需要必须是连续的内存空间
>
> > 不需要移动元素
> >
> > > 内存利用率更高

- 缺点：链表中每个元素的地址都必须在上一个元素身上才能获取，那么就会出现一个问题

> > > 获取链表中第N个元素时，须从链表的首个元素访问，链式获取内存地址才能找到，效率很低。
>
> > 因为数组是有序排列，只需简单的运算即可找到数组中的任何元素

总结

> **由二者特性可知：**

- 数组的插入操作时间为O(n),读取时间为O(1)
- 链表的插入操作时间为O(1)，读取的时间为O(n)

数组的读取速度快，插入速度慢，链表的读取速度慢（随机读取非全读取），插入速度快，根据实际业务场景选择合适的存储方案。