数据结构（上）

队列、栈、链表、哈希表、树

版权声明

联系方式

如果你想要购买本课程
请微信联系 xiedaimala02 或 xiedaimala03

如果你发现有人盗用本课程
请微信联系 xiedaimala02 或 xiedaimala03

目前我们用过的数据结构

- 数组
  - ✓ 数组可以分为队列、栈等
- 哈希表
  - ✓ 用来存储 key-value 对

队列 Queue

「先进先出 FIFO」的数组

队列

- 题目
  - 请实现一个餐厅叫号网页
  - 点击「取号」按钮生成一个号码
  - 点击「叫号」按钮显示「请 X 号就餐」
- 代码
  - 首先选择队列 queue 作为数据结构
  - queue.push 为入队 / queue.shift 为出队
  - 记得练习一下 call 的用法
  - 其他的事情就顺其自然了，见完整代码
- 问
  - 如果不知道队列，你会怎么做?

栈 Stack

「后进先出 LIFO」的数组

栈

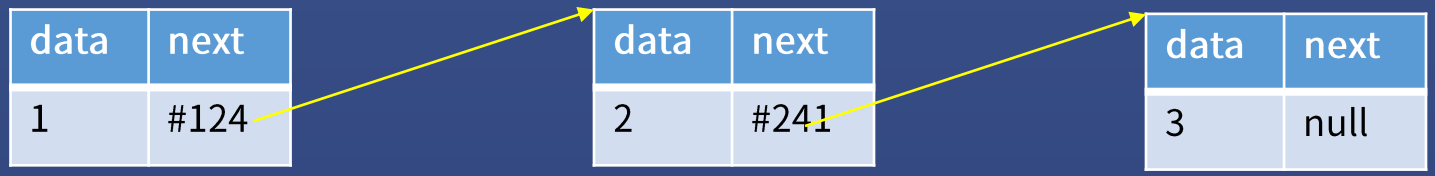
- 举例
  - ✓ JS 函数的调用栈 call stack 就是一个栈
  - ✓ 假设 f1 调用了 f2，f2 有调用了 f3
  - ✓ 那么 f3 结束后应该回到 f2，f2 结束后应该回到 f1
- 代码

```
function f1(){let a = 1; returb a+f2()}
function f2(){let b = 2; return b+f3()}
function f3(){let c = 3; return c}
f1()
```

  - ✓ 画一下压栈、出栈过程便知这是「后入先出」的栈

留一个悬念

- 内存图里的栈内存和这个调用栈
  - 是什么关系？——关系很大
  - 是同一块内存吗？——有很大的重叠
  - 我们之后再专门讲



链表 Linked List

一个链一个

链表

- 实际使用

```
let array = [1,2,3]
array.__proto__ === Array.prototype
Array.prototype.__proto__ === Object.prototype
```

- ✓ 从这个角度看对象，就是链表

- 代码

- ✓ list = create(value)
- ✓ node = get(index)
- ✓ append(node, value)
- ✓ remove(node)
- ✓ travel(list, fn)

链表的变形

- 双向链表
  - ✓ 每个节点有一个 previous 指向上一个节点
- 循环链表
  - ✓ 最后一个节点的 next 指向头节点

哈希表

key-value pairs

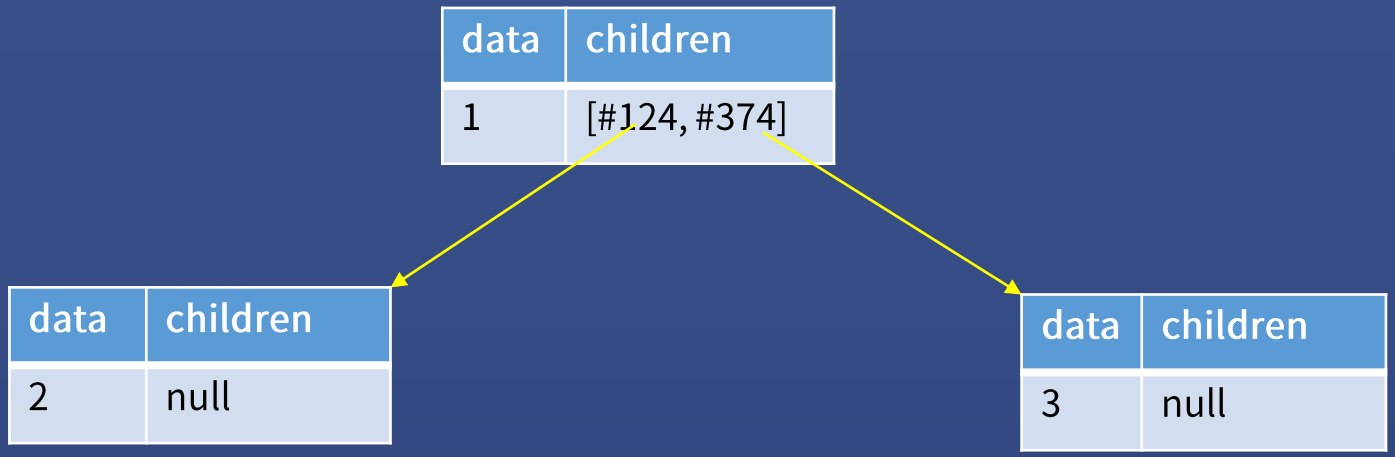
这玩意有什么难点吗

有的，难点在于哈希二字（不是嘻哈）

这里有一篇文章有兴趣可以看看

哈希表的难点

- 场景
  - 假设哈希表 hash 里有一万对 key-value
  - 比如 name: 'frank', age: 18, p1: 'property1' ...
  - 如何使得读取 hash['xxx'] 速度最快
- 解决办法
  - 不作任何优化，hash['xxx'] 需要遍历所有 key，O(n)
  - 对 key 排序，使用二分查找，O(log2 n)
  - 用字符串对应的 ASCII 数字做索引，O(1)
  - 对索引做除法取余数，O(1)
  - 冲突了怎么办，冲突了就顺延


data
children
1
[#124, #374]
data
children
2
null
data
children
3
null

树 Tree

一个链多个

树

- 实际使用
  - 中国的省市区，可以看成一棵树
  - 公司的层级结构，可以看成一棵树
  - 网页中的节点，可以看成一棵树
- 代码
  - let tree = createTree(value)
  - let node = createNode(value)
  - addChild(tree, node)
  - removeChild(node1, node2)
  - travel(tree)

数据结构就这么简单?

老师你是不是在骗我

程序员崇尚简洁优雅

如果你觉得某个编程概念很难

那么你一定理解错了

请试着重新理解一下

——方方

下节课学习二叉树

再见