Javascript数据结构

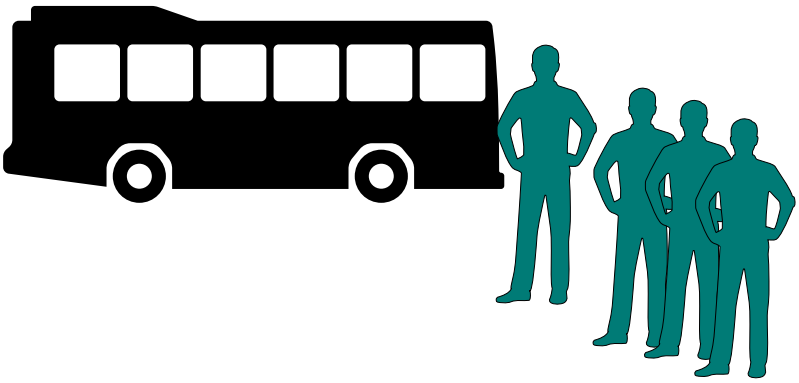
队列

Skipper

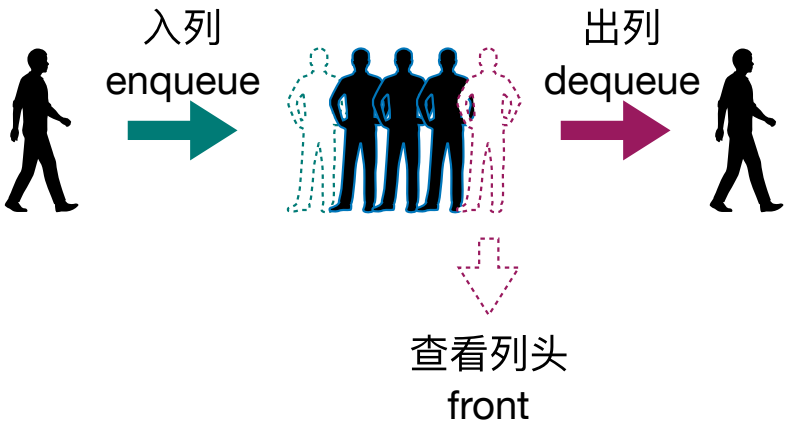
红点工场
redpoint.live

什么是队列?

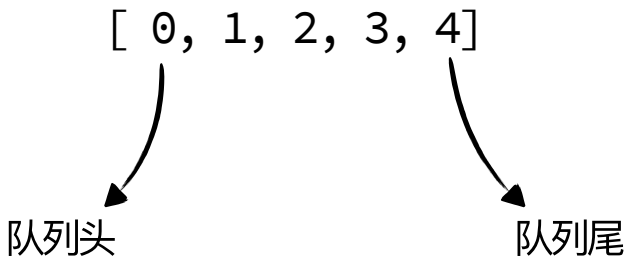
一种先进先出（FIFO）的数据结构



队列操作



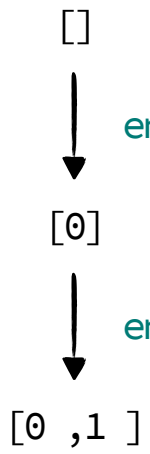
使用数组实现栈结构



类封装队列操作

```
var Queue = function(){
    var items = [];
}
```

入队操作：enqueue




```
this.enqueue = function(element){
    items.push(element);
}
```

出队操作：dequeue

[0 ,1]

dequeue()

[1]

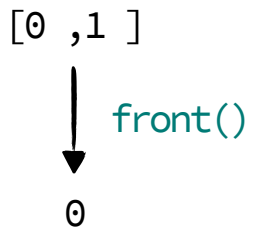
dequeue()

[]

```
this.dequeue = function(){
    return items.shift();
}
```

*数组的shift方法可以删除第一位元素

查看队列头：front




```
this.front = function(){
    return items[0];
}
```

其他操作

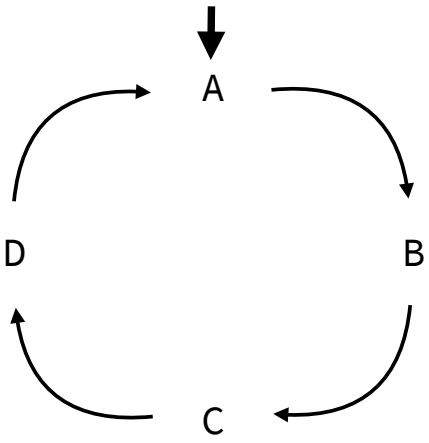
检查队列是否为空 `isEmpty()`

```
this.isEmpty = function(){
    return items.length == 0;
}
```

检查队列长度 `size()`

```
this.size = function(){
    return items.length;
}
```

循环队列实战：击鼓传花



由A开始，每数到3踢出一名玩家

问题分解

初始化：全部入列　[A, B, C, D]

第一次击鼓传花　1. [A, B, C, D]

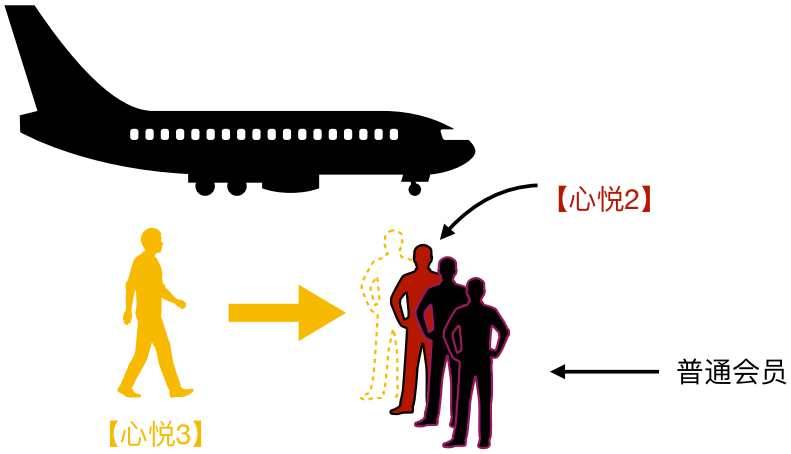
2. [B, C, D, A]

3. [C, D, A ,B] → C 出列 [D ,A ,B]

循环队列实现

```
queue.enqueue(queue.dequeue());
```

更常用的队列：优先队列



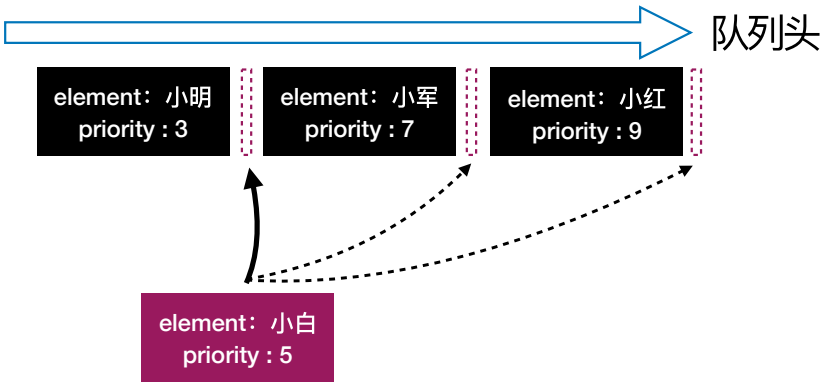
使用辅助类实现优先队列

```
var PriorityQueue = function(){
    var items = [];
    var QueueElement = function(element ,priority){
        this.element = element;
        this.priority = priority;
    }
}
```

new QueueElement 

element
priority

优先队列入列：enqueue



从队列头开始，逐一比较优先级（priority）

代码实现

```
this.enqueue = function(element ,priority){
    var queueElement = new QueueElement(element ,priority);
    var added = false;

    for(var i = 0; i < items.length ; i++){
        if(queueElement.priority > items[i].priority){
            items.splice(i ,0 ,queueElement);
            added = true;
            break;
        }
    }
    if(!added){
        items.push(queueElement);
    }
}
```

实现一个打印方法测试


```
this.print()
```

小红 - 9
小军 - 7
小明 - 3

打印方法

```
this.print = function(){
    for(var i = 0; i < items.length ; i++){
        console.log(items[i].element,' - ',items[i].priority );
    }
}
```