Javascript数据结构

字典

Skipper

红点工场
redpoint.live

什么是字典?

一种类似集合的数据结构

Set{ 1:1 , 2:2}	**[值：值]对**
Dictionary{ “name”:”红楼梦” , “price”:200}	**[键：值]对**

*JS的数据类型对象就是字典的一种实现

实现字典结构

```
var Dictionary = function(){
    var items = {};
}
```

字典主要操作

添加键值对：set(key ,value)

通过键值移除元素：delete(key)

检查键：has(key)

由键获取值：get(key)

检查键 has(key)

```
this.has = function(key){
    return key in items;
}
```

添加键值对 set(key ,value)

```
this.set = function(key ,value){
    items[key] = value;
}
```

通过键移除元素 delete(key)

```
this.delete = function(key){
    if(this.has(key)){
        delete items[key];
        return true;
    }
    return false;
}
```

由键获取值 get(key)

```
this.set = function(key ,value){
    return items.has(key) ? items[key] : undefined;
}
```

字典其他操作

一、以列表返回字典值

```
this.values = function(){
    var values = [];
    for(var k in items){
        if(this.has(k)){
            values.push(items[k]);
        }
    }
    return values;
}
```

二、获取全部键名

```
this.keys = function(){
    return Object.keys(items);
}
```

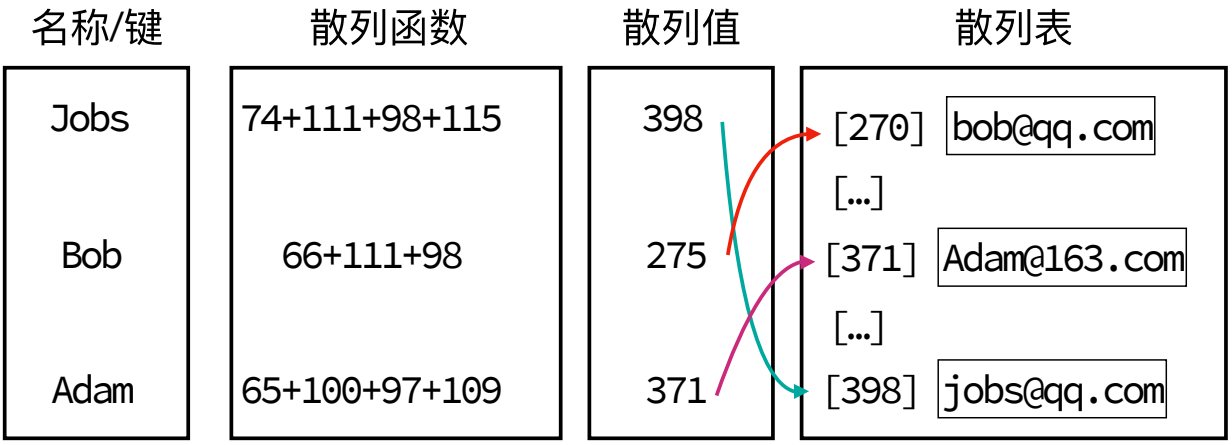
三、获取私有变量items

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

四、其他操作

clear()、size()与之前相同

散列表（哈希表）和散列算法



散列表VS其他数据结构

其他数据结构：获取值的时候需要遍历元素

散列表：可以快速定位元素

实现散列表结构

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

散列表基本方法——散列函数

```
var loseloseHashCode = function(key){
    var hash = 0;
    for(var i = 0; i < key.length ;i++){
        hash += key.charCodeAt(i);
    }
    return hash % 37;
}
```

散列表操作方法

一、添加元素 put(key ,value)
二、移除值 remove(key)
三、检索值 get(key)

散列表添加元素 put(key ,value)

```
this.put = function(key ,value){
    var position = loseloseHashCode(key);
    console.log(position + ‘ - ’ + value);
    table[position] = value;
}
```

散列表检索元素 get(key)

```
this.get = function(key){
    return table[loseloseHashCode(key)];
}
```

散列表移除元素 remove(key)

```
this.remove = function(key){
    table[loseloseHashCode(key)] = undefined;
}
```

解决散列表冲突

table[13] = Ana

Donnie因为添加早，被覆盖

分离链接法

为每一个位置创建链表

线性探查

如果位置被占用则线性向下移动

更好的散列函数：djb2

```
var djb2HashCode = function(key){
    var hash = 5381;
    for(var i = 0; i < key.length ;i++){
        hash = hash * 33 + key.charCodeAt(i);
    }
    return hash % 1013;
}
```