Javascript 雜湊表(HashTable)

```javascript
function hash(key, size) {
    let hashCode = 0;
    for (let index in key) {
        hashCode += key.charCodeAt(index);
    }
    return hashCode % size;
}

class Box {
    constructor(key,value) {
        this.key=key;
        this.value=value;
    }
}

class HashTable {
    constructor() {
        this.size = 16;
        this.slots = new Array(this.size).fill(null);
        this.length = 0;
    }
}
```

這個程式是雜湊表(HashTable)

概念是透過雜湊函數計算出key所對應的一個slot，將該組key與value存入該slot之中，進而建立雜湊表格，如果不同的key找到相同對應的slot時，則在該slot內依序存進該slot中的陣列裡面。當要尋找資料時，key能快速找到其相對位置的slot，如果不同的key找到相同對應的slot時，則在該slot內的陣列中線性搜尋該相同的key，並取得其相對應的value，如此有加密與節省尋找時間的功能。
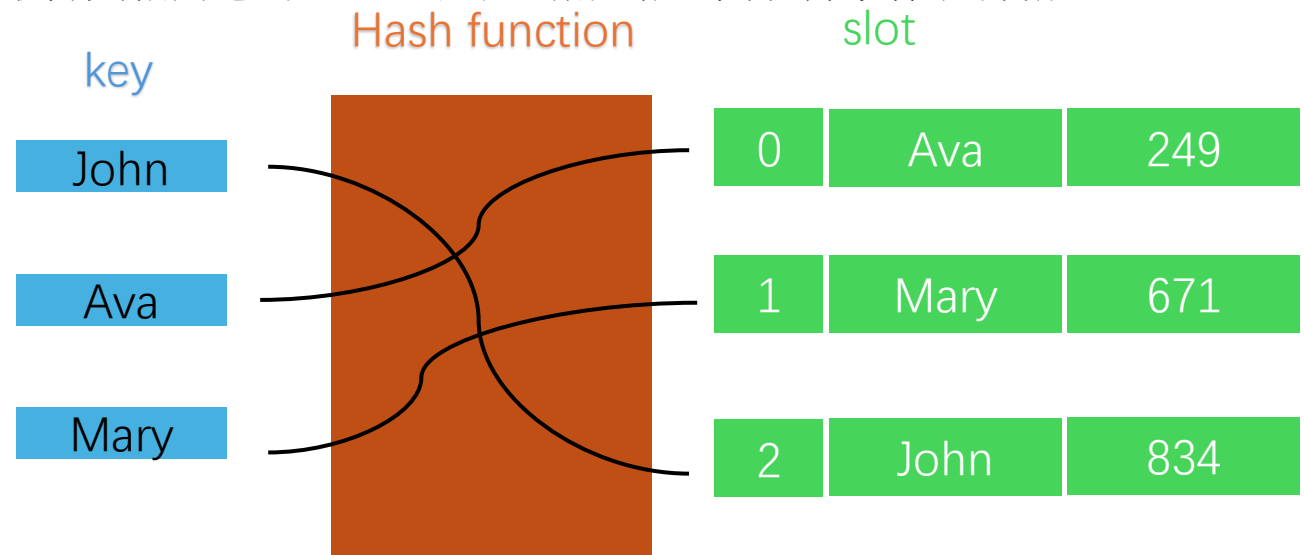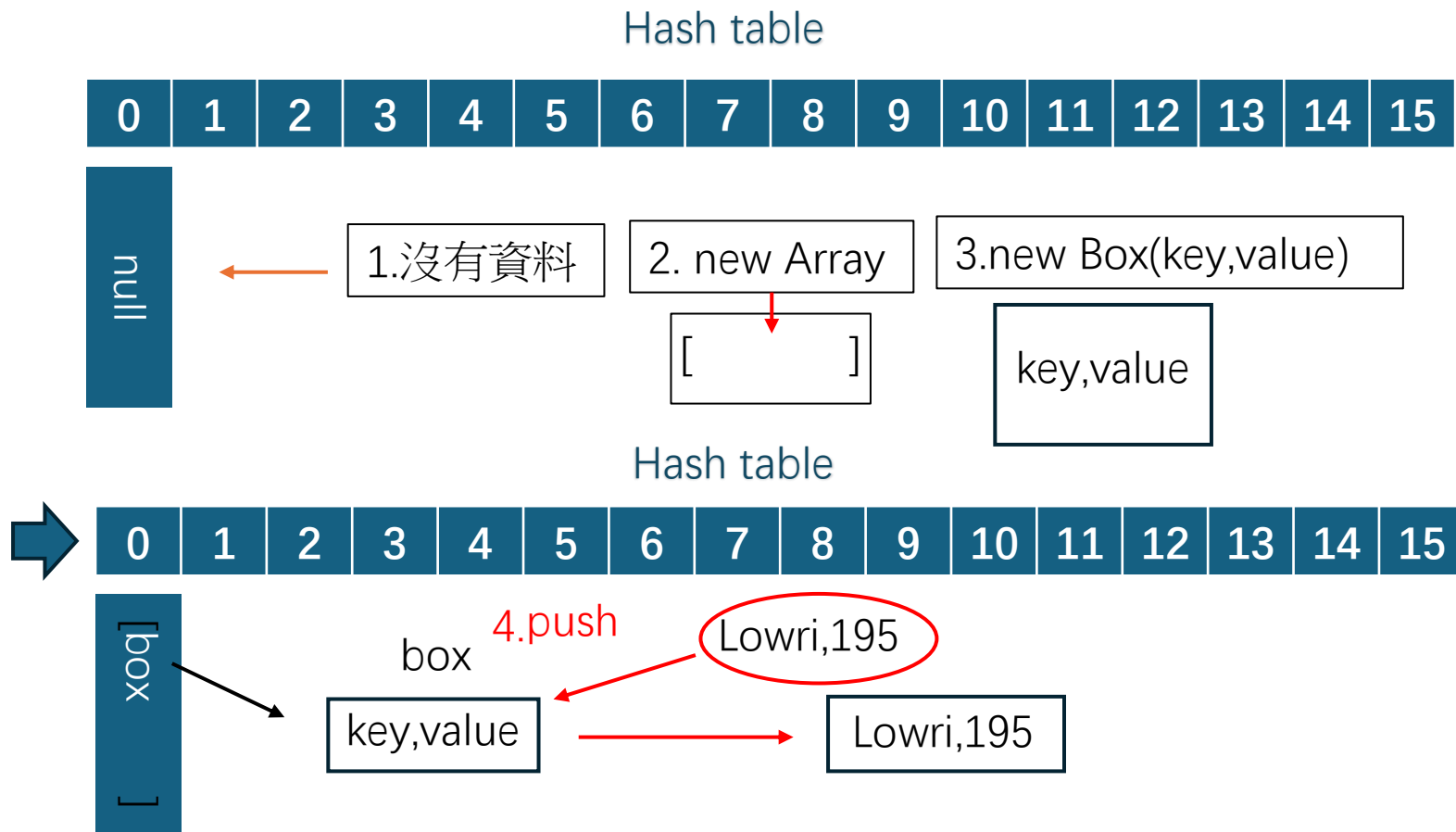

本程式使用的雜湊函數是以charCodeAt()，將key的字串先換成整數後除以slots的size取餘數來決定key與value放的slot位置，在本程式中的雜湊表size是16，所以就是餘16。
本程式用class 定義box 來存放key與value，定義HashTable 有16個slots預先填入null，並用length存入資料的數量。

add(key,value)是將key與value存入slot的方法，會先用if確認該slot內是否已經有陣列可以存放box，如果沒有就先創建一個陣列，再創建1個box放入陣列中，然後將box用push填入box陣列中，

```
add(key,value) {
    const hashIndex = hash(key, this.size);
    const slot = this.slots[hashIndex];
    if (!slot) {
        this.slots[hashIndex] = new Array();
        let box = new Box(key,value);
        this.slots[hashIndex].push(box);
        this.length++;
    } else {
        let found=false;
        for(let i=0;i<slot.length;i++) {
            let box = slot[i];
            if (box.key==key) {
                box.value=value;
                found=true;
                break;
            }
        }
        if (!found) {
            let box = new Box(key,value);
            this.slots[hashIndex].push(box);
            this.length++;
        }
    }
}
```

Hash table

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

null

1.沒有資料

2. new Array

[   ]

3.new Box(key,value)

key,value

Hash table

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

[box]

4.push

box

Lowri,195

key,value → Lowri,195

```
22    add(key,value) {
23        const hashIndex = hash(key, this.size);
24        const slot = this.slots[hashIndex];
25        if (!slot) {
26            this.slots[hashIndex] = new Array();
27            let box = new Box(key,value);
28            this.slots[hashIndex].push(box);
29            this.length++;
30        } else {
31            let found=false;
32            for(let i=0;i<slot.length;i++) {
33                let box = slot[i];
34                if (box.key==key) {
35                    box.value=value;
36                    found=true;
37                    break;
38                }
39            }
40            if (!found) {
41                let box = new Box(key,value);
42                this.slots[hashIndex].push(box);
43                this.length++;
44            }
45        }
46    }
```

如果slot中已經有box陣列的情況下，如果有相同key不同value的資料加入時，本程式會用新資料替換掉舊資料，用一個found變數當flag，預設為false，然後用for迴圈在陣列中線性尋找是否有相同的key的box。如果有找到相同key的box，則將新的value替換進該box中，將found設為true，然後離開迴圈。這樣就可以將舊value換成新value了。而如果沒有找到相同的key，則該key為新的key，則創建1個新的box放入box陣列中，然後將資料用push填入box陣列中。
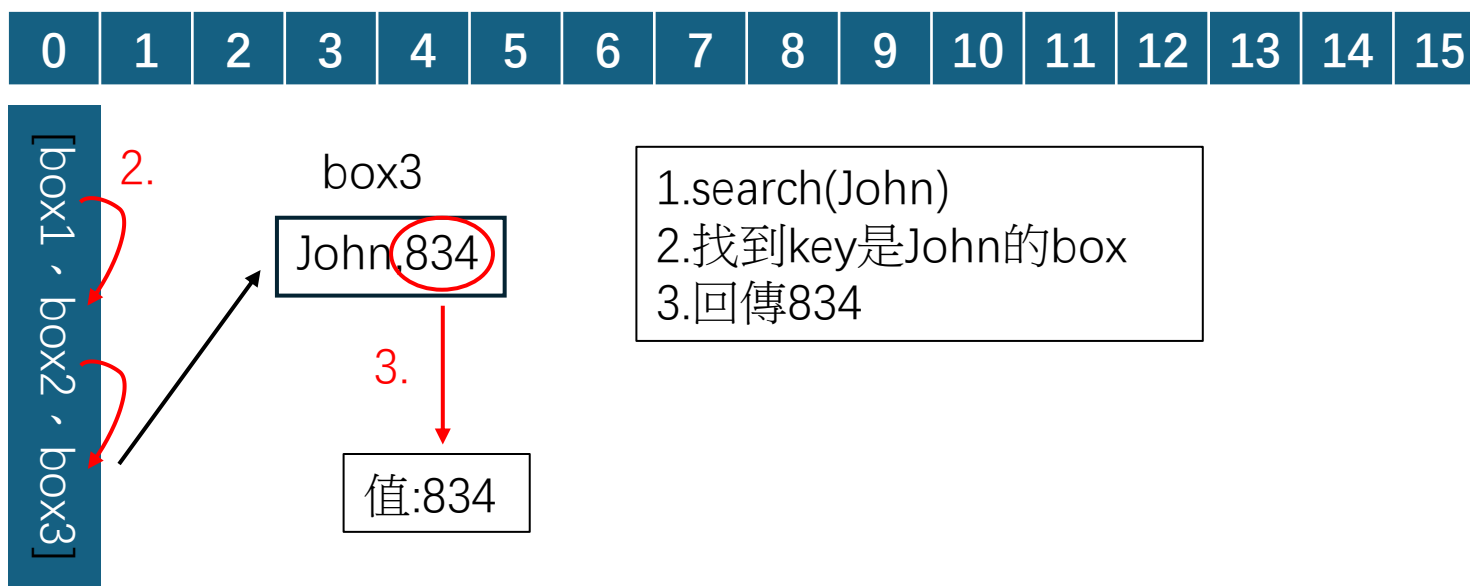
唯有替換資料的部分沒有放入**this.length++**，因為資料總數不變。

```
search(key) {
    const hashIndex = hash(key, this.size);
    const slot = this.slots[hashIndex];
    for (let i=0;i<slot.length;i++) {
        let box = slot[i];
        if (box.key==key) {
            return box.value;
        }
    }
    return "查無此資料!";
}
```

search(key)是用key找到已存入資料的方法，透過hash函數找到slot，再利用for迴圈在box陣列內線性搜尋是否有相同的key的box，若有相同，則回傳該box中的value，否則回傳"查無此資料!"。

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

[box1、box2、box3]

2.

box3

John,834

3.

值:834

1.search(John)
2.找到key是John的box
3.回傳834

```
58    remove(key) {
59        const hashIndex = hash(key, this.size);
60        const slot = this.slots[hashIndex];
61        for (let i=0;i<slot.length;i++) {
62            let box = slot[i];
63            if (box.key==key) {
64                slot.splice(i,1);
65                this.length--;
66            }
67        }
68        return "資料已清除!";
69    }
70 }
71
72 const ht = new HashTable();
73
74 ht.add("Louis", "767");
75 ht.add("Lowri", "195");
76 ht.add("Michelle", "162");
77 ht.add("Julie", "934");
78 ht.add("John", "329");
79 ht.add("Theresa", "331");
80 ht.add("John", "834");
81 ht.add("Eddie", "378");
82 ht.add("Robin", "77");
83 ht.add("Alfie", "532");
84 ht.add("Ava", "249");
85 ht.add("Owen", "208");
86 ht.add("Theodore", "539");
87 ht.add("Savannah", "54");
```

remove(key)是用來移除資料的方法，同樣透過透過hash函數找到slot，再利用for迴圈在box陣列內線性搜尋是否有相同的key的box。如果有，就使用splice將該位置的box移除。最後回報"資料已清除!"。

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

[box1、box2、box3]

2.

box3

John,834

3.

[box1、box2]
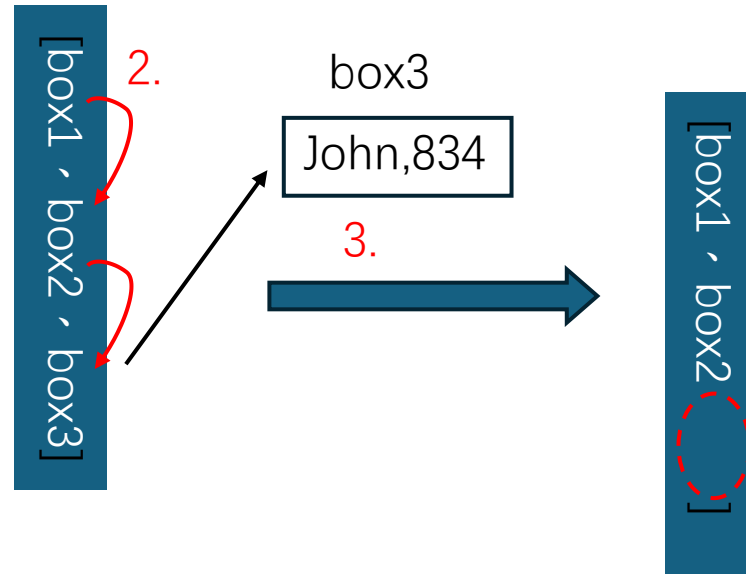
1.remove(John)
2.在box陣列中找到key是John的box
3.移除box

```
88    ht.add("Marc", "938");
89    ht.add("Sara", "420");
90    ht.add("Carmen", "599");
91    ht.add("Christine", "415");
92    ht.add("Mary", "671");
93    ht.add("Melissa", "171");
94    ht.add("Gethin", "483");
95    ht.add("Vanessa", "396");
96    ht.add("Eden", "638");
97    ht.add("Zoya", "72");
98    ht.add("Mae", "23");
99    ht.add("Edith", "609");
100   ht.add("Elizabeth", "660");
101   ht.add("Abdul", "619");
102   ht.add("Tabitha", "745");
103   ht.add("Jon", "957");
104   ht.add("Francis", "621");
105   ht.add("Millie", "767");
106
107   console.log(ht.search("John"));
108   console.log(ht.search("Ava"));
109   console.log(ht.remove("John"));
110   console.log(ht.search("John"));
```

這是實際程式表現出來的結果

| 問題 | 輸出 | 偵錯主控台 | 終端機 | 連接埠 |

C:\Program Files\node.exe .\hashtable.js
834
249
資料已清除！
查無此資料！

完整程式列表

```javascript
function hash(key, size) {
    let hashCode = 0;
    for (let index in key) {
        hashCode += key.charCodeAt(index);
    }
    return hashCode % size;
}

class Box{
    constructor(key,value){
        this.key=key;
        this.value=value;
    }
}

class HashTable {
    constructor() {
        this.size = 16;
        this.slots = new Array(this.size).fill(null);
        this.length = 0;
    }
    add(key,value){
        const hashIndex = hash(key, this.size);
        const slot = this.slots[hashIndex];
        if (!slot){
            this.slots[hashIndex] = new Array();
            let box = new Box(key,value);
            this.slots[hashIndex].push(box);
            this.length++;
        } else {
            let found=false;
```

```javascript
        for(let i=0;i<slot.length;i++){
            let box = slot[i];
            if(box.key==key){
                box.value=value;
                found=true;
                break;
            }
        }
        if(!found){
            let box = new Box(key,value);
            this.slots[hashIndex].push(box);
            this.length++;
        }
    }
}
search(key){
    const hashIndex = hash(key, this.size);
    const slot = this.slots[hashIndex];
    for(let i=0;i<slot.length;i++){
        let box = slot[i];
        if(box.key==key){
            return box.value;
        }
    }
    return "查無此資料!";
}
remove(key){
    const hashIndex = hash(key, this.size);
    const slot = this.slots[hashIndex];
    for(let i=0;i<slot.length;i++){
        let box = slot[i];
        if(box.key==key){
            slot.splice(i,1);
            this.length--;
        }
```

```javascript
        }
        return "資料已清除!";
    }
}

const ht = new HashTable();

ht.add("Louis", "767");
ht.add("Lowri", "195");
ht.add("Michelle", "162");
ht.add("Julie", "934");
ht.add("John", "329");
ht.add("Theresa", "331");
ht.add("John", "834");
ht.add("Eddie", "378");
ht.add("Robin", "77");
ht.add("Alfie", "532");
ht.add("Ava", "249");
ht.add("Owen", "208");
ht.add("Theodore", "539");
ht.add("Savannah", "54");
ht.add("Marc", "938");
ht.add("Sara", "420");
ht.add("Carmen", "599");
ht.add("Christine", "415");
ht.add("Mary", "671");
ht.add("Melissa", "171");
ht.add("Gethin", "483");
ht.add("Vanessa", "396");
ht.add("Eden", "638");
ht.add("Zoya", "72");
ht.add("Mae", "23");
ht.add("Edith", "609");
ht.add("Elizabeth", "660");
```

```
101    ht.add("Abdul", "619");
102    ht.add("Tabitha", "745");
103    ht.add("Jon", "957");
104    ht.add("Francis", "621");
105    ht.add("Millie", "767");
106
107    console.log(ht.search("John"));
108    console.log(ht.search("Ava"));
109    console.log(ht.remove("John"));
110    console.log(ht.search("John"));
```