**演算法和函數 - gcd為例**

# 大綱

1. **什麼是集合（Set）？**

集合是一個源自於數學理論中擁有不同元素的集合：

N = {0, 1, 2, 3, 4, 5, 6, …}

空集合：{}

其特性在於它是由一組無序且不重複的項目組成。你也可以想成是一個沒有重複元素和無順序的陣列。在這篇文章我們會介紹如何實作集合資料結構並使用 交集、聯集、差集等集合操作方式。

1. **集合初體驗**

事實上，在 ES6 中就有[原生的 Set](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set)，在這邊我們試著使用 JavaScript 物件模仿 ES6 的 Set 設計集合資料結構（使用物件好處是物件 key 是唯一）。

*The Set object lets you store unique values of any type, whether primitive values or object references.*

1. 建立集合類別和方法

|  |
| --- |
| function Set() {  var items = {};  this.add(value) {};  this.remove(value) {};  this.has(value) {};  this.clear() {};  this.size() {};  this.values() {}; } |

* + has(value)：判斷是否元素在集合中，若有回傳 true，反之傳回 false

|  |
| --- |
| this.has = function(value) {  return items.hasOwnProperty(value);  *// 由於我們是使用 JavaScript 物件實作所以也可以使用 value in items* } |

* + add(value)：新增元素到集合

|  |
| --- |
| this.add = function(value) {  if(!this.has(value)) {  items[value] = value;  return true;  }  return false; } |

* + remove(value)：刪除元素

|  |
| --- |
| this.remove = function(value) {  if(this.has(value)) {  delete items[value];  return true;  }  return false; }; |

* + clear()：移除集合所有元素

|  |
| --- |
| this.clear = function() {  items = {}; } |

* + size()：回傳集合元素數量
    1. 使用類似於 Linked List 的 length 變數計算法，當新增、刪除元素時順便增減長度

|  |
| --- |
| this.size = function() {  return length; } |

* + 1. 使用 JavaScript 內建 Object 類別的內建函數 keys

|  |
| --- |
| this.size = function() {  return Object.keys(items).length; } |

* + 1. 手動迭代判斷是否存在集合中並累加個數

|  |
| --- |
| this.size = function() {  let count = 0;  *// 特別注意在 JavaScript 中 for in 會一起把繼承於 Object 類別和物件自身的所有相關非相關資料結構的屬性一起迭代出來*  for(let prop in items) {  *// 判斷是否屬於 items 的屬性*  if(items.hasOwnProperty(prop)) {  ++count;  }  return count;  } } |

* + values()：回傳集合所有值，使用 JavaScript 內建 Object 類別的內建函數 keys 以陣列形式回傳

|  |
| --- |
| this.values = function() {  return Object.keys(items); } |

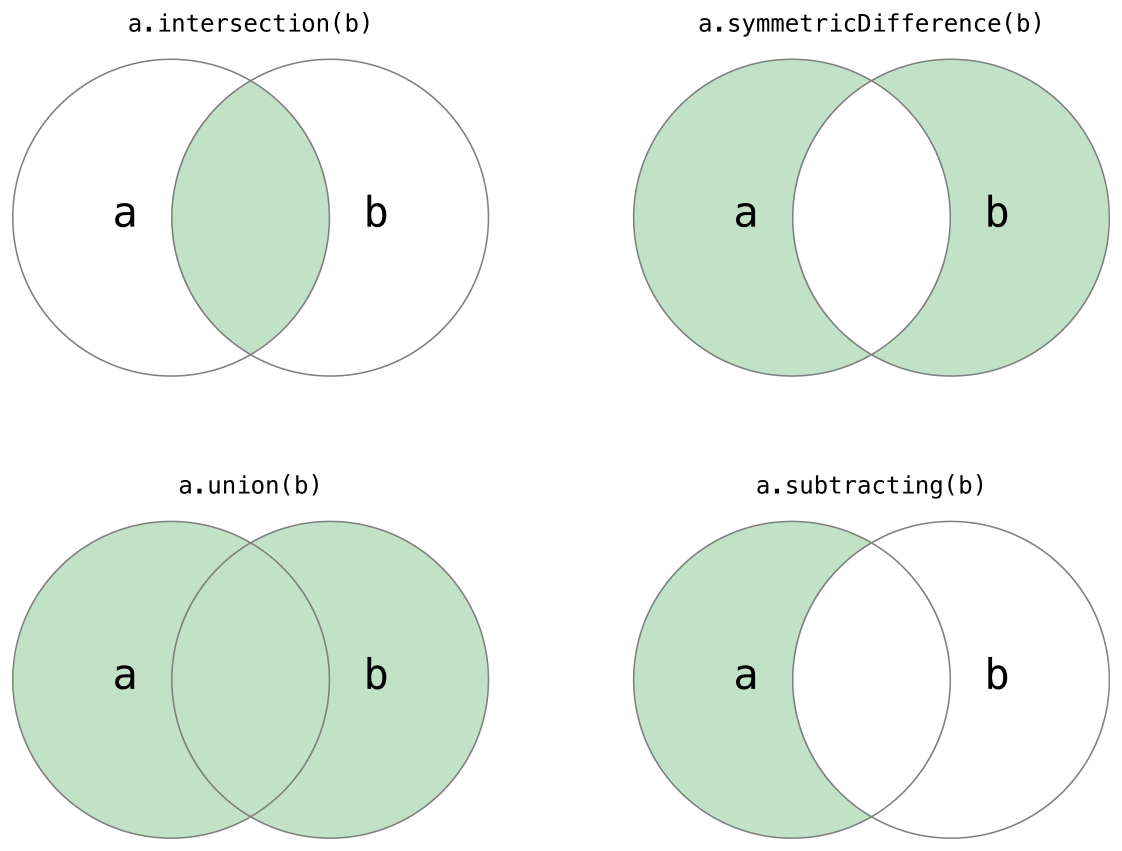
* + 另外一種瀏覽器相容性較高的寫法

|  |
| --- |
| this.value = function() {  let keys = [];  for(let key in items) {  keys.push(key);  }  return keys; } |

1. 使用集合類別

|  |
| --- |
| const set = new Set(); set.add(12); console.log(set.values()); console.log(set.has(12)); console.log(set.size()); set.add(12); set.add(7); set.remove(12); set.add(1); console.log(set.has(12)); console.log(set.values()); console.log(set.size()); |

1. **集合操作**



參考數學上的集合概念，我們可以針對集合進行以下操作：

1. 聯集  
   對於給定兩集合，回傳一個包含兩個集合中所有元素的新集合

|  |
| --- |
| this.union = function(otherSet) {  *// 首先建立代表聯集的新集合*  let unionSet = new Set();  let values = this.values();  for(var i = 0; i < values.length; i++) {  unionSet.add(values[i]);  }  values = otherSet.values();  for(let j = 0; j < values.length; j++) {  unionSet.add(values[i]);  }  return unionSet; } |

1. 在 console 測試：

|  |
| --- |
| let setA = new Set(); setA.add(1); setA.add(4); setA.add(2);  let setB = new Set(); setB.add(1); setB.add(4); setB.add(2); setB.add(7);  let unionSet = setA.unique(setB); console.log(unionSet); |

1. 交集  
   對於給定兩集合，回傳一個包含兩個集合中共有元素的新集合

|  |
| --- |
| this.intersection = function(otherSet) {  let intersectionSet = new Set();  let values = this.values();  for(let i = 0; i < values.length; i++) {  if(otherSet.has(values[i])) {  intersectionSet.add(values[i]);  }  }  return intersectionSet; } |

1. 在 console 測試：

|  |
| --- |
| let setA = new Set(); setA.add(1); setA.add(4); setA.add(2);  let setB = new Set(); setB.add(1); setB.add(4); setB.add(2); setB.add(7);  let intersectionSet = setA.intersection(setB); console.log(intersectionSet); |

1. 差集  
   對於給定兩集合，回傳一個包含所有存在第一個集合但不存在於第二集合的元素集合

|  |
| --- |
| this.difference = function(otherSet) {  let differenceSet = new Set();  let values = this.values();  for(let i = 0; i < values; i++) {  if(!otherSet.has(values[i])) {  differenceSet.add(values[i]);  }  }  return differenceSet; } |

1. 在 console 測試：

|  |
| --- |
| let setA = new Set(); setA.add(1); setA.add(4); setA.add(2);  let setB = new Set(); setB.add(1); setB.add(4); setB.add(2); setB.add(7);  let differenceSet = setA.difference(setB); console.log(differenceSet); |

1. 子集  
   驗證給定集合是否為另一個集合的子集

|  |
| --- |
| this.subSet = function(otherSet) {  if(this.size() > otherSet.size()) {  return false;  } else {  let values = this.values();  for(let i = 0; i < values.length; i++) {  if(!otherSet.has(values[i])) {  return false;  }  return true;  }  } } |

1. 在 console 測試：

|  |
| --- |
| let setA = new Set(); setA.add(1); setA.add(4); setA.add(2);  let setB = new Set(); setB.add(1); setB.add(4); setB.add(2); setB.add(7);  let subSet = setA.subSet(setB); console.log(subSet); |

1. **總結**

在這單元中我們學會了：

1. 什麼是集合（Set）？
2. 建立集合類別和方法
3. 集合操作（聯集、交集、差集、子集）

接下來我將繼續探索 Dictionary 和 Hash Table 的世界。