JavaScript @ AstroCamp

Part 2: 集合、迭代與資料結構

前情提要

Cheat Sheet

```
/* Variable (變數) */
// Declear a variable (宣告一個變數)
let a;
// Assign value to the variable (幫變數指派一個值)
a = 1;
// 宣告同時指派
// ***接東西***
let b = 2;

/* Primitive types 基本資料型別 */
// Number 數字
1 // Inteter 整數
0.3 // Float 浮點數

//String 字串
"foo"
'bar'
`baz` // 只有這個裡面可以放 string interpolation

// 布林值
true
false
```

```
/* Function 函式 */
// ***販賣機***
function foo(input) {
  // things within the curly braces are called "function body"
  // 大括號內叫"函式本體"
  return someOutput; // return value 回傳值 // ***丟東西出去***
}

foo(100); // call the function 呼叫函式

/* 分支條件 branches */
// if/else
if (someCondition) {
  // someCondition 符合時會跑這塊
} else if (otherCondtion) {
  // otherCondition 符合時會跑這塊
} else {
  // 都不符合時會跑這塊
}

// switch case
switch(someExpression) {
  case 'x':
    // do something
    break;
  case 'y':
    // do something
    break;
  default:
    // do something
}
```

Section 0: 關於 JavaScript 函式的一些補充

1. 定義時的參數數量在呼叫時**完全不檢查**
2. destructive assignment

Arity

```
function foo(x, y, z) {
}

foo(1, 2, 3)
foo(1, 2, 3, 4, 5) // 這樣可以嗎？
foo() // 那這樣呢？
```

Destructive assignment: on Array

```
let numbers = [1, 2, 3, 4, 5]

let first = numbers[0];
let second = numbers[1];

// become
let [first, second] = numbers;

// advance
let [head, ...tails] = numbers;
```

Destructive assignment: on Object

```
let student = {name: 'John', age: 18}

let name = student.name
let age = student.age

// become
let {name: name, age: age} = student;

// then becom
let {name, age} = student;
```

Destructive assignment: in function definition

```
function foo(student) {
  let name = student.name;
  let age = student.age;
}

// become
function foo({name: name, age: age}) {
  // use name and age directly
}

// become
function foo({name, age}) {
  // use name and age directly
}
```

Section 1: 兩種集合

三種用法

1. 丸子串：陣列

```
let genre = ['pop', 'jazz', 'tango', 'blues']
```

>> 通常是一群類似的東西**依順序**放在一起(稍後有個小小的例外)

陣列的取值跟設值

```
genre[0] // 取值
genre[5] = 'kpop' // 設值
genre[0] = 'jpop' // 改值
```

陣列可以用的屬性 (attribute) 跟方法 (method)

```
let fruits = ['apple', 'banana', 'orange', 'pear']
fruits.length

fruits.join()

fruits.indexOf()
fruits.concat()
```

從頭尾的單個操作

```
frutis.pop() // 把最後一個元素丟出來
fruits.shift() // 把最前面的元素丟出來

fruits.push('pear') // 在最後面加一個元素
fruits.unshift('grape') // 在最前面加一個元素
```

tips: FIFO (queue) vs LIFO (stack)

會不會改變原先的資料？(mutation)

```
fruits.slice(-3)
```

```
fruits.splice(1, 1, 'x', 'y')
```

Section 2: 迭代 (iteration)

迴圈：for...of

會跟陣列(或是其它可迭代的東西)一起用

```
let langs = ['JavaScript', 'Ruby', 'Elixir', 'Haskell']

for(let lang of langs) {
  console.log(`I love ${lang}`)
}
```

迴圈：for

```
for(let i = 0; i < langs.length; i++) {
  console.log(`I love ${langs[i]}`)
}
```

Note: 其實還有一種 `for...in`，但要被廢棄了，儘量不要用

2. 像是字典的東西：鍵值對

```
let fruitPrices = {"apple": 3,  "banana": 2,  "orange": 4}
```

key-value pair 在其它的語言裡有不同的名字，Ruby 裡叫 `Hash`、Elixir 裡叫 `Map`、Java 裡叫 `HashMap`、Python 跟 C# 裡叫 `Dict`(dictionary)。在 JavaScript 裡叫 `Object`(先忘記其它的語言上的 Object)。

物件的兩種用法

1. 查表
2. 表示一個實體的屬性

1. 查表

```
let country = 'tw'
let currencies = {'us': 'USD', 'uk': 'GBP', 'tw': 'NTD', 'jp': 'JPY'}

console.log(currencies[country])
```

2. 表示一個東西的狀態 (通常是名詞)

```
let user = {
  name: 'bob',
  age: 18
  // 想一想，還會有哪些？
}
```

物件的取值跟設值

```
let user = {name: 'bob', age: 18}

user.name
let key = 'age'
user[key]

user.gender = 'M' // 設值
user['gender'] = 'M'

user['age'] = 19 // 改值
```

物件常用的屬性與方法

```
Object.keys(user)
Object.values(user)
Object.entries(user)

Object.fromEntries()
Object.assign()
```

Section 3:
把基本型別跟集合一起使用

資料結構 (Data structure)

用各種型別來表示一個小小的世界

```
let user = {
  name: 'Joy',
  age: 18,
  gender: 'M'
}
```

再往上一層呢？

```
let classRoom = {
  name: '三年八班',
  teacher: {name: 'tai', age: 20},
  assistant: {name: 'fred', age: 18},
  students: [{name: 'Joan', age: 18},{name: 'b', age: 18}],
}
```

練習一下

怎麼操作它們

```
function allAges(classRoom) {
  let teacherAge = classRoom.teacher.age
  let assistantAge = classRoom.assistant.age
  let studentAges = []
  // classRoom.students = [{age: 18}, {age: 20}, {age: 30}]
  for(let s of classRoom.students) {
    studentAges.push(s.age)
  }
  return [teacherAge, assistantAge].concat(studentAges)
}

let ages = allAges(classRoom)
```

隱藏的資料結構 Tuple (元組)

```
let students = [
  {name: 'john', age: 20, gender: 'M'},
  {name: 'mary', age: 18, gender: 'F'},
  {name: 'jean', age: 21, gender: 'F'},
]
//
let students2 = [
  ['john', 20, 'M'],
  ['mary', 18, 'F'],
  ['jean', 21, 'F'],
]
```

另一個例子：座標

```
let points = [
  {x: 1, y: 1},
  {x: 2, y: 2},
  {x: 3, y: 3},
]
//
let points2 = [
  [1, 1],
  [2, 2],
  [3, 3],
]
```

注意的點：一個陣列裡，是放同類別的東西，還是不同類別的東西

偷偷爆個雷：為什麼 JavaScript 的物件叫物件？

```
let cat = {
  name: 'mimi',
  age: 3,
  eat: function() { /* ... */ }
}
```

some ideas

- 一間餐廳的菜單
- 購物車
- 音樂播放清單
- 一間旅館
- 洗脫烘衣機
- ...

作業

7-11 的商品跟不同的打折方式

Happy hacking!