ES6篇: Object Enhancement(物件的增強)

本章的目標是對ES6中的物件字面文字、以及物件中的方法，所提供的改進或增強部份，提供一些較為全面的說明。JavaScript本身就是一個完全的物件導向的程式語言，物件的字面文字語法相當的簡單，也有很多用途。在ES6中對於物件的字面文字語法，提供了許多小的改進，也加入了一些專門用於物件的新方法，這些許多小的改進，讓物件本身的運用可以更佳的彈性，語法上也會更為簡潔。

> 註: 本文章同步放置於Github庫的這裡。

物件字面語法(Object Literal)的增強

ES6中對於物件的字面文字語法，作了許多增強與改進。直接看與ES5中對比的例子會比較明確。

> 註: 下面的例子因為要對比ES5的語法，所以用var來定義變數，實際上你應該用let或const來定義變數/常數。

屬性初始設定簡寫法(Property Initializer Shorthand)

這個語法也有人寫作屬性值簡寫法(Property value shorthand)，都是指同樣的內容。

這是當如果物件字面中的屬性，是使用變數/常數來定義時，因為變數/常數中已經帶有值，所以直接使用變數/常數的識別名當作屬性即可，不需要再寫出屬性的值。通常會看到有兩個地方使用這個簡寫法，一個是函式中用於回傳某個物件時，另一個是用物件字面來定義新的物件時。

ES5中的語法如下:

```
function foo(a, b) {
    return {result: 'success', a: a, b: b}
}

var a = 'foo', b = 42, c = {}
var o = { a: a, b: b, c: c }
```

ES6的改寫的簡短語法如下，你應該有注意到少了用冒號(:)來指定物件中的屬性的值:

```
function foo(a, b) {
    return {result: 'success', a, b}
}

var a = 'foo', b = 42, c = {}
var o = { a, b, c }
```

類似的語法之前是只有陣列字面語法中可以用像下面的例子來寫，不過陣列沒有屬性這東西，它是用索引來對應裡面的值:

```
var a = 10
var b = [a]
```

這個簡寫語法現在在React或Redux裡很常見，例如在Action Creators(動作建立函式)中都會看到像下面的程式碼:

```
function addTodo(text) {
  return {
    type: ADD_TODO,
    text
  }
}
```

方法定義(Method Definitions)

之前在Class(類別)的章節中，你應該有看到在類別中定義的方法，是直接用方法名稱作定義，沒有使用 `function` 這個關鍵詞。在ES6中的物件字面中定義方法，現在也可以不需要 `function` 關鍵詞，這也是一種簡寫語法。

ES5中的語法如下:

```
var player = {
    fullName: 'Eddy',
    sayName: function() {
        console.log(this.fullName);
    }
}
```

ES6的改寫的簡短語法如下:

```
var player = {
    fullName: 'Eddy',
    sayName() {
        console.log(this.fullName);
    }
}
```

這樣的簡寫法大概是可以讓程式碼看起來更簡潔，在使用物件字面語法定義複雜的物件結構時，會看起來更佳清楚，閱讀性提高。

> 註: 上面所說的對屬性與方法的兩個簡寫法，在ESLint工具會協助檢查，規則是object-shorthand

預設傳入參數(Default Params)

函式的數設傳入參數之前有介紹過了，就不再多說。現在也可以用於物件字面語法中的方法中。例如以下的例子:

```
const player = {
    fullName: 'Eddy',
    sayName(word='Hi') {
        console.log(word + ' ' + this.fullName);
    }
}

player.sayName()
player.sayName('Hello')
```

計算得出的屬性名稱(Computed Property Names)

這是在ES6中可以讓你動態的產出屬性的識別名稱，這會用在某一些特定的情況下。要計算的屬性名需要用方括號([])框住，直接看例子會比較明確:

```
const prop = 'foo'

const o = {
  [prop]: 'hey',
  ['b' + 'ar']: 'there',
}

console.log(o) //Object {foo: "hey", bar: "there"}
```

也就是說物件中的屬性名稱可以是計算得到的(動態產生的)，而不是一開始就立即要定義好的。

當然，如果你用了計算得出的屬性名稱的語法，就不能再使用上面說的屬性初始設定簡寫法，這會造成語法錯誤，這一點要特別的注意，像下面的例子是錯誤的示範:

```
// 這是錯誤的例子
const prop = 'foo'

const o = {
  [prop]
}
```

物件的新方法

ES6中加入了一些對於物件在運算時的新方法，其中有些方法是非常常見的而且有用的。

Object.assign

這個方法在之前有一個填充用的函式庫，它是讓還沒有ES6這個特性的瀏覽器使用的，在React中已經用很久了。

Object.assign是用來作物件的合併(merges)或混合(Mixins)、拷貝(淺拷貝)使用的方法，語法相當的簡單像下面這樣，出自MDN:

```
Object.assign(target, ...sources)
```

拷貝一個物件在一般的物件中是很容易作得到的，當然這只是個淺拷貝，語法也很簡單:

```
const obj = { a: 1 }
const copy = Object.assign({}, obj)
```

合併物件成為新的物件，語法也很容易，在Object.assign傳入放愈後面位置的物件，它的屬性會覆蓋掉前面的物件，這不難理解:

```
const o1 = { a: 1, b: 1, c: 1 }
const o2 = { b: 2, c: 2 }
const o3 = { c: 3 }

const obj = Object.assign({}, o1, o2, o3)
console.log(obj) // { a: 1, b: 2, c: 3 }
```

如果你只需要把一個新的方法，加入到原來的物件中，在ES5中的用法是在原型上指定，像下面這樣的語法:

```
MyClass.prototype.foo = function (arg1, arg2) {
        //...
    }
```

使用 `Object.assign` 也可以作這件事，但它是類似於合併物件的語法，像下面這樣:

```
Object.assign(MyClass.prototype, {
        foo(arg1, arg2) {
            //...
        }
})
```

> 註: 由於ES6使用的是 `assign` 作為方法的名稱(也就是"指定值"的意思)，而不是 `mixin` (混合)，它與在JS中常用的mixin樣式的行為仍然有一些差異，例如它們對存取器屬性(accessor properties)的行為不同。

> 註: 關於Object.assign的討論可以相當的深入而且複雜，如果你有興趣可以參考這篇文章中的說明。

結論

本章很快的說明一些在ES6中關於物件字面語法的簡寫法，以及介紹一個新的物件方法 `Object.assign` ，當然ES6中的物件新方法不只這一個。會介紹這個方法與上一章的"純粹函式"有關，因為物件類型的資料的純粹函式的寫法，都是使用這個方法來寫，拷貝出一個新的物件，或是加入新的方法或屬性裡去這個物件中。

這些語法在接著的React與Redux章節中，都會不斷的看到，在這裡整理出來只是先用簡單的例子來說明與預先學習。之後當你看到這些語法時，就會有一些印象。

本章是ES6篇的最後一個章節。在短短的三十天的文章分享過程中，有些ES6中的大的新特性，沒有辦法一一在這裡分享，例如像Promise, Generators等等，這些特性大概也沒辦法用一篇文章就可以說得完整。因為我們的主題是React，而且是偏向入門的知識部份，所以在ES6篇中大部份都是較為簡單而且入門的知識，希望在開始進行React前，把大概會用到的ES6新特性，以及一些概念說完。

下一章將是進入真正的主題 - React。相信如果你已經讀過這些ES6篇裡的基本知識，學起來會感到更輕鬆，React當然也有它特殊的一些規則與設計，我們在接著的章節都會直接用簡單的實例來說明。就像我一直強調的重點，React與Redux並不難學，難的是因為你沒有紮實的ES6新特性的基礎。