JavaScript ES6

JavaScript 語言新一代的標準

變數宣告

差異說明

var (variable)

作用域為「函式作用域」；於函式內宣告並作用，如於函式外宣告則為全域變數。
缺點：容易污染全域變數

```
function varTest() {
  var a = 1
  if (true) {
    var a = 2
    console.log(a) // 2
  }
  console.log(a) // 2
}
```

let

作用域為「區塊作用域」；於 { } 內宣告並作用，離開範圍後將不被存取。

```
function letTest() {
  let b = 1
  if (true) {
    let b = 2
    console.log(b) // 2
  }
  console.log(b) // 1
}
```

const (constant)

必須賦予值，且禁止重新賦值

```
const constTest = 2

console.log(constTest) // 2

constTest = 4
// 直接報錯
```

字串模板

用法說明

插入表達式

```
function product(name, price, discount) {
  // ES5
  console.log('Product-' + name + ' is on sale!')

  // ES6
  console.log(`Product-${name} is now NT${price * discount}`)
}
```

多行字串

```
const product_info = `
  <div class="box">
    <div class="info">${item.name}</div>
    <div class="info">${item.price}</div>
    <div class="info">${item.stock}</div>
  </div>
`
```

箭頭函式

用法說明

語法縮寫

```
// ES5
var double = function (x) {
  return x * 2
}
// ES6
const double = (x) => {
  return x * 2
}
// 參數只有一個，且 return 只有一行時
const double = x => x * 2
```

this 指向（指向最外層）

```
const test = () => {
  console.log(this)
}
const objA = {
  name: 'Jarvis',
  test
}

test() // window
objA.test() // window
```

解構賦值

陣列解構

基礎賦值

```
const nums = [1, 2, 3]

// ES5
var first = nums[0]
var second = nums[1]

// ES6
const [first, second] = nums
```

預設值

```
const nums = [1, 2]

const [first, second, third = 0]
= nums

console.log(third) // 0
```

忽略元素

```
const nums = [1, 2, 3]

const [, first] = nums
const [,, third] = nums

console.log(third) // 3
```

解構賦值

陣列解構

變數交換

```
let a = 1
let tempA = a
let b = 2

// ES5
a = b
b = tempA

// ES6
[a, b] = [b, a]
```

其餘運算（可淺拷貝）

```
const nums = [1, 2, 3, 4]

const [first, ...others] = nums

console.log(first) // 1
console.log(others)
// [2, 3, 4]
```

解構賦值

物件解構

縮寫賦值

```
const x = 100
const y = 100

// ES5
var newObj = { x: x, y: y }

// ES6
let newObj = { x, y }
```

基本賦值

```
const point = { x: 10, y: 20 }

// ES5
var x = point.x
var y = point.y

// ES6
const { x, y } = point
```

預設值

```
const point = { x: 10, y: 20 }

const { x, y, z = 30 } = point

console.log(z) // 30
```

解構賦值

物件解構

賦予新名稱

```
const point = { x: 10, y: 20 }

const { x: nX, y: nY } = point

console.log(nY) // 20
```

其餘運算（可淺拷貝）

```
const point = { x: 1, y: 2, z: 3 }

const { x, ...others } = point

console.log(x) // 1
console.log(others)
 // { y: 2, z: 3 }
```

前端框架概念

React / Vue / Angular

為什麼要使用框架？

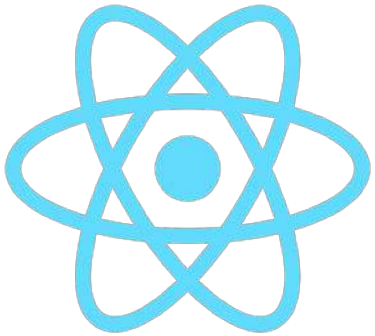
1. jQuery 程式碼大部分情況下是麵條程式碼，現代框架幫助我們進行分層，程式碼解耦更易於讀寫

3. 前端框架的根本意義是解決了**畫面與狀態同步問題**

2. jQuery 或 JS 頻繁操作 DOM，導致效能低下；步驟過多不易維護

4. 「框架」其實就是一種提升開發效率、降低維護難度的開發架構

三大框架比較



React.js

原作者	Jordan Walke
開發者	Facebook
發布時間	2013年3月
架構	Virtual DOM
著重	JavaScript

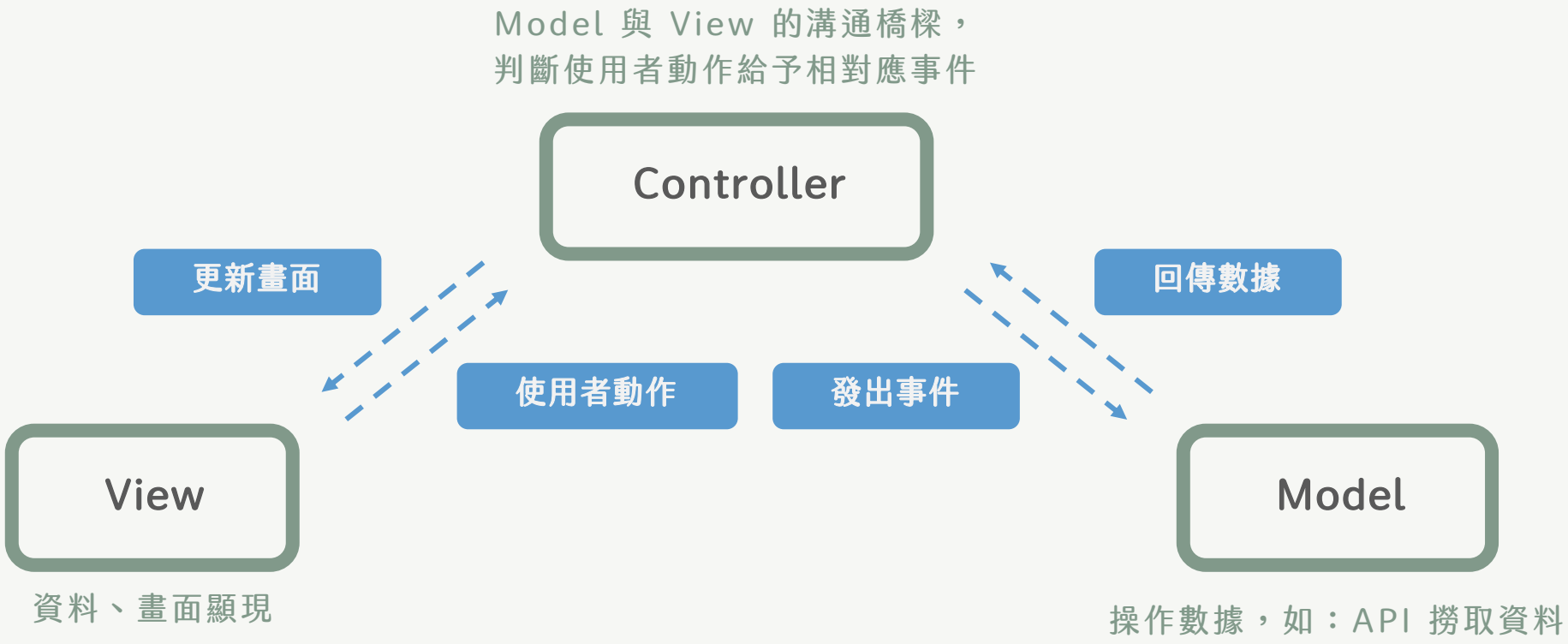
Vue.js

原作者	尤雨溪
發布時間	2014年2月
架構	MVVM
著重	Vue 語法

Angular.js

開發者	Google
發布時間	2010年10月
架構	MVVM
著重	TypeScript

MVC

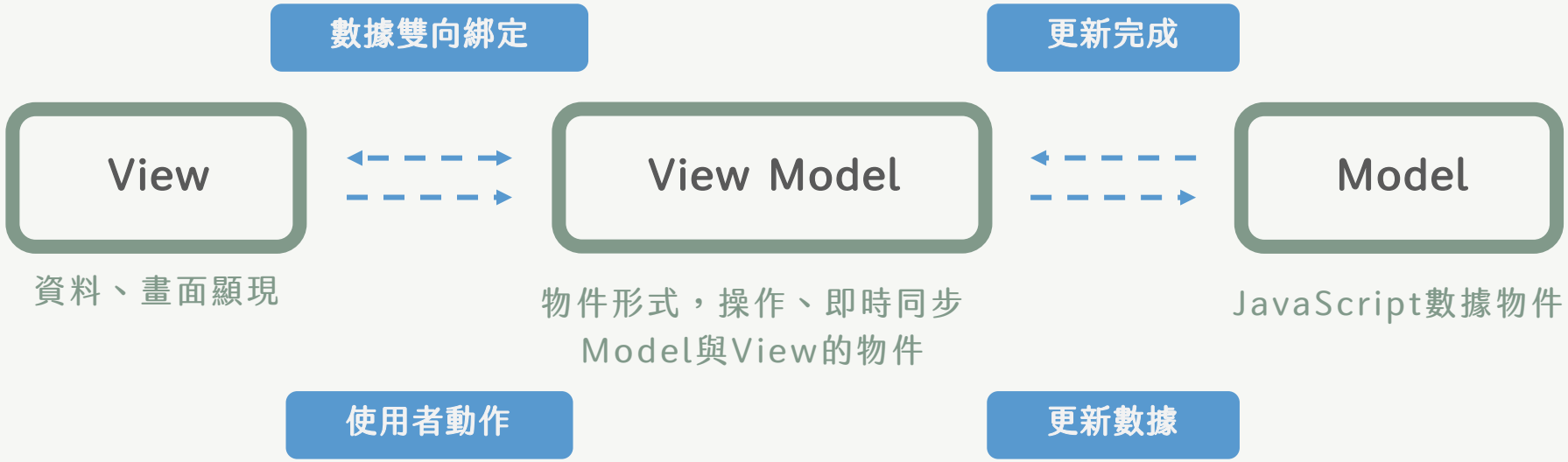


缺點：

1、開發時，大量調用相同 DOM API，處理及操作繁瑣，程式碼不易維護

2、大量操作 DOM 使頁面渲染性能降低、加載速度變面，影響用戶體驗

MVVM



缺點：

1、數據綁定使得 Bug 不易被調試

2、大的模塊，model 很大，不利於內存的釋放