ES6 語法

> ECMAScript 是 Javascript 規格標準，Javascript 隨著時間的演進，在 2015 迎來了一次重大改版，是為 (ECMAScript 6)ES6。白話來說，我們以前學的語法，由於新的函式庫興起(如 React)，已經不敷使用。ES6 新的語法可以與新的函式庫相互搭配，因此學習 ES6 式入門 React 必經之路，大家一起來學習吧!

> 這邊講的只是 ES6 的部分內容，尚未包含全部

先由變數宣告說起

`var` 再見，來認識 `let` 與 `const`

- 原本我們變數宣告都會使用 `var` ，但由於透過 `var` 宣告的區域變數會汙染到**全域變數**。像是宣告在像是 if-else 區域的變數也會汙染到全域變數

```
var a = 2;
{
    var a = 4;
    console.log(a); //4
}
console.log(a); //也是 4
```

- `let` , `const` ，其存取範圍只有本身定義的區塊中。

```
let a = 2;
{
    let a = 4;
    console.log(a); //4
}
console.log(a); //2
```

- `let` 用在可變的變數上， `const` 用在常數上

```
const c = 0;
c = 3;
//會出現以下錯誤，因為常數不可以更動
Uncaught TypeError: Assignment to constant variable.
at <anonymous>:2:7
```

    - 但也是有例外，透過 `const` 宣告的物件(Object)，由於存的是物件的參考，因此透過 `.` 選取物件的key，是可以更改該 key 的值得

```
const people = {
    name: 'Jerry',
    age: 24
}

people.age = 25;
console.log(people.age); // 25
```

> 實務上，let 通常會搭配 for 迴圈中的 i 之宣告(因為會一直更動)；而大部分的變數則會使用 const 宣告(因為通常是要選取一個 DOM 或者是固定不變的資料)。但還是要視情況來宣告，會一直更動的值就要選擇 let

再來是物件的處理

解構賦值 Destructure

> 應用陣列、物件上 把陣列、物件裡面的東西給它新的名字

陣列 ::必須對應位置::

```
let arr = [1,2,3,4];

let [first,second,third,fourth] = arr; //讓每個陣列元素都有對應的名稱

console.log(first) // 1
console.log(fourth)  //4

//也可以只賦予特定位置的值，其他可以空白(必須對應位置)
let [first, ,third ,] = arr;

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

物件 ::必須對應**key** 名稱::

```
const obj = {
    name: 'Jerry',
    age: 24,
    address: 'Taiwan'
}

let {name, age, address} = obj
console.log(name) //Jerry
```

展開與其餘

展開 Spread

> 複製物件及陣列

陣列

```
// Spread 展開運算
let arr1 = [1,2,3];
let arr2 = [4,5,6]; // 如果陣列裡面放陣列會是二維

//將 arr1 及 arr2 的物件攤開來，再放進去 arr3
let arr3 = [...arr1,...arr2];
console.log(arr3); [1,2,3,4,5,6]
```

物件

```
const obj1 = {
  a: 1,
  b: 2
}

const obj2 = {
  ...obj1,
  c: 3
}

console.log(obj2);
//出來的結果為
{
    a:1,
    b:2,
    c:3
}
```

反向展開(其餘) Rest

> 跟解構搭配 ::Rest 只能放在最後面::

```
// Rest

let arr7 = [1,2,3,4];
let [first, ...rest] = arr;
console.log(first); //1
console.log(rest); //[2,3,4]

//如果其中一個改成
✕ ✕ ✕ let [first, ...rest, last] =arr; //錯
//rest 只能放最後面
```

```
let objrest = {
  a: 1,
  b: 2,
  c: 3
}

let {a, ...restobj} = objrest;
console.log(restobj);

//結果為
{
  b: 2,
  c: 3
}
```

模版字符串（template literal）

- 以前要將字串接在一起，我們會這麼做

```
const name = "Jerry"
let old = "<h1><p>"+name+"</p></h1>"
```

- 但現在你可以使用 `` (重音符)取代 ""
  - 在`` 中可以直接寫 JS ，也可以任意斷行
  - 使用 ${} 包住JS程式

```
const name = "Jerry"
let new = `<h1>
<p> ${name} </p>
</h1>
`
```

函式也有重大改變

箭頭函式 Arrow function

```
// 過去宣告的方法
var old = function (n){
  return n;
}

//或是匿名函式
function(n) {
    return n;
}
```

箭頭函式的方法

```
// 箭頭函式
const arrow = (n) => {
  return n;
}

// ‼如果只有一個參數 括弧可以省略
const oneargs = n => {
  return n;
}
```

好處

```
// 搭配 array
let arr8 = [1,2,3,4,5];
console.log(
  arr8
    .filter(function(value){
      return value > 1;
    })
    .map(function (value) {
      return value * 2;
    })
)

//改寫
console.log(
  arr8
    .filter(value => {
      return value > 1;
    })
    .map(value => {
      return value * 2;
    })
);

// 再簡寫 //‼如果只要回傳一行 {}可以省略
console.log(
  arr8
  .filter(value => value > 1)
  .map(value => value * 2)
);
```

JSX 語法

> JSX 是一種像是在"在JavaScript 中直接寫 HTML" 的語法

以前我們寫 JS 的事件

- 例如，點擊按鈕彈出 alert 視窗，我們可能會撰寫以下的程式碼

```
<button onclick="alert('hello')">點我</button>
```

- 或者會把 JS 獨立出來，透過 id 綁定相對應的DOM (事件監聽法)

```
<button id="btn">點我</button>
<script>
    document.getElementById('btn').addEventListener('click',function(){
    alert('hello')
    },false)
</script>
```

以前由於不想讓 html 跟 JS 程式碼混雜在一起，會較推崇後者的作法，但在 React 的世界裡面，我們將焦點關注於"元件"本身。因此在 React 上，在撰寫 JS 的時候會與 UI(html) 綁定在一起

JSX

- 可以在html tag中使用 JS 任何語法，只要用 `{}` 包住即可，可以傳入變數、加減乘除，甚至是表達式
    - 傳入變數

    ```
    const name = 'Jerry'
    const hello = <h1>hello, {name}</h1>
    ```

    - 加減乘除

    ```
    const year = 2020
    return ( //return 是 React 渲染函數的返回值
        <div>
            <p> 今年是民國 {year-1911} 年</p>
        </div>
    )
    ```

    - 三元表達式

    ```
    const isDone = true
    return (  //return 是 React 渲染函數的返回值
        isDone ? <h1>isDone 為 true 時才顯示</h1> : ''
    )
    ```

- 在 React ，我們會將元件寫成　JSX　來表示
    - 假如我們有個元件名稱為：Menu，那它會長以下這樣

```
<Menu />
```

    - 像是一個HTML的 tag，只是名稱是自訂義

- 這個元素可能有相對應的事件動作

```
<Menu onClick={()=>{
 alert('hello')
}} />
```

    - 這裡的寫法就和以前將 click 寫在 html Tag 上一樣
        - "" 變成 {} ，包住JS 語法(此處是箭頭函式)
    - 但這裡這樣寫 onClick 其實是 React 事先處理好的方法，透過撰寫成元件的 attribute 將 JS 程式碼傳進去

- 除事件之外，元件還有其他 attribute

    - 加入 class 名稱
        - 在 html tag 中我們使用 `class` ，但在 React 的元件上， `class` 是保留字，要使用 `className`
        - 另外，使用 htmlFor 來替代 for，因為 for 也是保留字

```
<Menu className="menu" onClick={()=>{
 alert('hello')
}} />
```

    - 自訂義的屬性 Props -> 這個我們會在上課提到
    - 更多 attribute 可以參考這裡
    - React 的元件也可以寫成，前後兩個tag 包起來

```
<Menu> </Menu>
```

- JSX 是如何做到讓瀏覽器看得懂，可以參考這裡