變數、判斷式、迴圈及函式

JavaScript 基本四要素

變數宣告

基本介紹

var (variable)

var 是宣告的意思 (variable 的簡寫)，後面接自定義的變數名稱，並於等號後方賦予自定義數值，可為：

布林值（Boolean）

數字（Number）、字串（String）

物件（Object）、陣列（Array）

空值（Null）、未定義（undefined）

等形式

使用原則

1. 盡量命名較語意化的名稱，日後維護才看得懂之前寫了什麼
2. 變數名稱不能是數字開頭
3. 字母的大小寫是有區別的，因此變數 box 和變數 Box 是不同的變數
4. 如果變數賦予的值型別是 number，可以使用變數來做運算

實際用例

```
var status = false    var text = 'Hello!'    var number = 0
var list = []         var price = null       var todo = undefined
```

變數宣告（ES6）

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
```

判斷式

If...else

```
if ( 條件 ) {
  成立時執行的動作
} else if ( 其他條件 ) {
  成立時執行的動作
} else {
  前面都不成立時執行的動作
}
```

三元判斷式

```
條件 ? 成立時執行的動作 : 不成立時執行的動作

實際用例：
let score = 70
const level = score >= 60 ? 'good' : 'bad'
console.log(level)
// good
```

比較運算子

等於	不等於	大於、小於	大於等於、小於等於	並且、或者
==	!=	>	>=	&& （多條件需全部成立）
=== 較嚴謹（例：'1' === 1 // false）	!== 較嚴謹	<	<=	|| （多條件只需其中一個成立）

迴圈

while

```
初始條件
while ( 條件範圍 ) {
  執行指定動作
  迴圈動作
}

實際用例：
let i = 0
while (i < 10) {
  console.log(i)
  i++  // i = i + 1
}
// 0-9
```

for

```
for ( 初始條件; 條件範圍; 迴圈動作 ) {
  執行指定動作
}

實際用例：
for (let i = 0; i < 10; i++) {
  console.log(i)
}
// 0-9
```

函式

函式常用於處理資料或程式碼共用

陳述式（具名函式）

```
function 函式名稱(參數) {
  執行指定動作
  return 回傳結果
}

實際用例：
function outputPrice(number) {
  return number * number
}
console.log(outputPrice(5))
// 25
```

表達式（匿名函式）

```
const 變數名稱 = function(參數) {
  執行指定動作
  return 回傳結果
}

實際用例：
const outputPrice = function(number) {
  return number * number
}
console.log(outputPrice(5))
// 25
```

this 指向（取決於函式情境）

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

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