JavaScript 程式設計新手村

單元10 JavaScript 基礎介紹

@kdchang

Outline

1. 變數
2. 基本資料型別
3. 運算子

變數

變數

變數可以視為一容器，可以在程式碼暫存資料，也可以更改所存的資料 (看成裝東西的盒子)

- JS 的基本型別(Primitives) 有 `String(文字)` 、 `Number(數字)` 、 `Boolean(true/false)` 、 `undefined` 、 `null`
- JS 中變數為寬鬆型別，不用事先宣告型別（type）
- 區分大小寫、不用保留字、不可以用數字開頭、不可用

指定敘述

在宣告變數的同時也可以給定初始值，給值的同時也決定了變數的型別（type），給定值使用等號（`=`），變數只是暫存，可以重新再指定值。等號並非數學相等而是賦值的意思，在JS中另外有（`===`）符號來比較相等

var 全域變數

少用全域變數，容易重複命名或是修改到產生污染。生存域看函數不是看區塊，沒有用 var 或在函數外宣告的變數，就屬全域範圍

```
var name = 'kdchang';
var blog = 'blog.kdchang.cc';

function funcName() {
        // 沒加 var 往上找到全域變數 name
        name = "cdchang";
        var blog = 'blog.cdchang.cc';
}

funcName();
console.log(name);
console.log(blog);
```

let 區域變數

在 ES6 後新增 let 區域變數可以使用，參考了 C 語言為區塊 `{}` 生存空間。不允許重複宣告、宣告前使用

```
function func() {
  let x = 1;
  console.log(x);
  for(let x = 0; x < 10; x++) {
   console.log(x);
  }
  let x = 1;
  console.log(x);
}

func();
```

let 區域變數

全域範圍以 let 宣告的變數，不會成為全域個體（global object）的屬性（window）。但以 var 宣告的變數同時也會是全域個體的屬性

```
let v = 1;
var w = 1;
console.log(v);
console.log(window.v);
console.log(w);
console.log(window.w);
```

const 常數

常數賦值後不能修改，不允許重複宣告、宣告前使用

```
function func() {
  console.log(y);
  for(let x = 0; x < 10; x++) {
   const y = 1;
   console.log(y);
  }
  const y = 3;
  console.log(y);
}

func();
```

延伸閱讀：學習 ECMAScript 6 - var, let 和 const

const 常數

全域範圍以 const 宣告的常數，不會成為全域個體（global object）的屬性（window）。但以 var 宣告的變數同時也會是全域個體的屬性

```
const x = 7;
console.log(x);
console.log(window.x);
```

延伸閱讀：學習 ECMAScript 6 - var, let 和 const

練習二

請分別賦值給 `var` 、 `let` 、 `const` ，並說明它們之間的差別？

基本資料型別

基本資料型別

基本上 JS 有五大基本資料型別(Primitives Data Types)，`Number(數字)`、`String(字串)`、`Boolean(布林值)`、`undefined`、`null` (又可視為 object)。 任何不屬於基本資料型別的都是 Object (物件)，可以使用 typeof 觀察變數資料型別

Number 數字

在 JS 中，沒有分所謂的整數、浮點數，一律都是Number。數字可以儲存：正負整數、浮點數、十六進位、八進位、指數、非數字(NaN)、Infinity、-Infinity

Number 數字

特殊值：

1. NaN(非數字)
   亦即 `Not a Number`，當運算結果為不正確時顯示(EX. 字串轉數字，數字和undefined做運算)，任何數跟 NaN 運算都變 NaN
2. Infinity(正無限)
   當數字太大超過JS最大值範圍
3. -Infinity(負無限)
   當數字太小超過JS最小值範圍

String 字串

字串是用兩個單引號 ' ' 或雙引號 " " 所包裹的文字型別 (EX. 0 或多個Unicode 字元，包含文字、標點符號和數字)，和 C 語言不同，並未支援如char() 這種單一字元，只要引號中為單一字元即可。推薦用單引號，將雙引號保留給 HTML 屬性值

Boolean 布林值

常用於條件判斷，決定要進入程式的哪個區塊，只有`true`和`false`兩種值

以下六種falsy值會自動轉為false ：

1. 空字串 " "
2. null
3. undefined
4. 數字 0
5. 非數字 NaN(仍屬Number Type)
6. false

（除以上六種外，其餘包括字串"0"、"false"皆視為true）

Undefinded

主要有兩種情形會產生`undefined`值：

1. 宣告變數後未初始值，會自動將其指定為undefined
2. 訪問物件不存在屬性或未宣告變數

NULL

null 值不像 undefined 會在未初始化自動給值，只能由我們給定，意思代表變數沒有值或是不是個物件，和undefined 不一樣，在轉換成數字時會變成 0

Escape 跳脫/逸出序列

Escape Sequence是以 \ 開頭的字元，可以在字串中表示需用特殊方式表示的字元或鍵盤無法輸入的字元，這邊只列最常用三個

```
\' -> '
\" -> "
\\ -> \
```

運算子

運算子

JS 的敘述運算式由運算元（operands）和運算子（operators）兩者組成。JS 中有算術、指定、位元、邏輯等運算子

```
const a = 1;
const b = 3;
let c = a + b
// a, b, 7為運算元; +, =為運算子
```

算術運算子

JS 提供了一般常用的數學運算符號：

1. 四則運算：+(加)、-(減)、*(乘)、/(除)、%(取餘數)
   其中加號也用於連結兩字串，減號於數字前表負數，所以使用上需注意

2. `++`(遞增運算 `x = x + 1`)、`--`(遞減運算 `x = x - 1`)使用上需注意置於變數前後結果可能不同，置於變數前為先運算再指定值

邏輯運算子

邏輯運算子常和布林值一起使用，並返回布林值。
主要有三種邏輯運算子：

1. `!` 表非(取相反)
2. `&&` 表邏輯且
3. `||` 表邏輯或

```
let a = true; let b = false;
!a -> false
a && b -> false
a || b -> true
```

位元運算子

JS 支援位元運算，可以進行二進位運算(EX. 向右移動幾位元、執行NOT、AND、XOR、OR運算)，唯課程中較不會使用到，這裡只列出參考資料

指定運算子

指定運算子可以簡化運算是，寫出更簡潔程式，同樣的這邊只列出常用指定運算子

```
x += y  (x = x + y)
x -= y  (x = x - y)
x *= y  (x = x * y )
x /= y  (x = x / y)
x %= y (x = x % y)
```

比較運算子

比較運算子用於比較運算元的大小，並根據比較結果返回布林值

運算元可以是數值、字串、布林或物件的值。字串是以 Unicode 的值作為標準的字典順序來比較。需要注意的是，如果兩個運算元不是同樣的類型 (type)，JS 會為了比較而嘗試把運算元轉換為適當的類型。等於 `(==)`、 不等於 `(!=)` 、嚴格等於 `(===)` 、嚴格不等於 `(!==)` 、 `(>)` 、 `(>=)`、 `(<)` 、 `(<=)`

比較運算子

強烈建議使用嚴格相等性`(===)`、`(!==)`最為比較方式，因為不會自動轉換型別，避免發生錯誤

```
const a = 1;
const b = 3;

console.log(a === b);
```

型別轉換

Type Conversions 表示在同一個運算式中會有不同的型別，所以需要轉換成同一型別方能運算。而 JS 為鬆散型別的語言，在指定值時才決定變數型別，基本上 JS 會強制轉換型別

數值 + 字串 (數值會強制轉字串)
布林 + 字串 (布林會強制轉字串)
布林 + 數值 (布林會強制轉數值)

(布林轉字串 "true" / 轉數值 = 1; 轉字串為 "false" / 轉數值 = 0)

用typeof()可以觀察值的型別，有、Number、String、Boolean、undefined、object、function

強制型別轉換

- parseInt()
  返回由第二個參數所指定的 radix （基數）的整數。如果 parseInt 遇到不是在指定基數之內的字元，就會直接忽略這個字元及其隨後字元，並返回在此之前已經分析出來的整數值。如果連第一個字元也不可以轉換為指定基數之內的字元，返回 NaN(非數字) 。parseInt 函數會切除字串以取得整數
- parseFloat()
  返回浮點數。如果遇到正負符號 (+ 或 -)、數字 (0-9)、小數點、指數以外的字元，會返回在此之前的數值，並忽略那些字元。如果連第一個字元也不可以轉換為數字，就會返回NaN(非數字)

運算子優先順序

運算式中通常有多個運算子，為了讓運算式可以有相同運算解果，運算式會有預設的優先順序。其實最簡單的就是我們耳熟能詳的『括號內先運算，先乘除後加減』

練習三

運用運算子和運算元計算將 (1 + 3) * 9 + (23 / 5)，並用 parseInt()將結果轉成整數和數字 24 比較大小，將比較結果使用 JSBin.com console 顯示

總結

在這個單元中我們學會了：

1. JavaScript 基礎介紹