網頁程式設計
JavaScript程式設計

Instructor: 馬豪尚

JavaScript

- JavaScript 是一種腳本，也能稱它為程式語言，可以讓你在網頁中實現出複雜的功能。
- JavaScript常用來完成以下任務
  - 嵌入動態文字於HTML頁面
  - 對瀏覽器事件作出回應
  - 讀寫HTML元素
  - 在資料被提交到伺服器之前驗證資料
  - 檢測訪客的瀏覽器資訊
  - 控制Cookie，包括建立和修改等

將JavaScript寫進HTML內

- 使用<script>將javascript程式寫入HTML文件內
  - 可放在<head>或<body>裡
  - 建議放在</body>前面，先讓頁面顯示出來再載入，比較不會有畫面延遲的情況

```
<body>
   <h1>歡迎光臨！</h1>
   <script>
     document.write('Hello, world!');
   </script>
         <noscript>無法使用JavaScript！</noscript>
</body>
```

將JavaScript寫進HTML內

- 透過事件屬性或超連結元素設定JavaScript
  - 寫在html元素內的"事件屬性"或使用"<a>"元素
  - JavaScript 的事件處理

```
<button onclick="javascript:window.alert('Hello, world!');">顯示訊息</button>
```

事件屬性　　事件處理程式

```
<a href="javascript:window.alert('Hello, world!');">顯示訊息</a>
```

<a> 超連結　　事件處理程式

將JavaScript程式放在外部檔案

- 撰寫外部JavaScript檔案
  - 開啟一個純文字檔，將副檔名命名為.js
- 使用<script>元素的src屬性來設定外部js的路徑
  - <script src="hello2.js"></script>

JavaScript程式碼撰寫慣例

- 英文字母有大小寫之分
  - Student/student是不同的變數
- 每一行一個敘述且結尾加上分號隔開
  - 不成文規定，非強制，藉此養成良好程式碼撰寫習慣
- 會忽略多餘的空白
- 程式碼縮排
  - 通常以兩個空白字元為縮排
- 註解的寫法為
  - 單行註解 → //
  - 多行註解 → /* 註解 */

自訂變數或函式名稱命名規則

- 和其他程式語言一樣，命名規則非強制性
- 第一個字元可以是英文字母、底線(_)或錢字符號($)，而後面的其他字元可以是英文字母、底線(_)或數字
- 不能使用JavaScript關鍵字以及內建函式、內建物件的名稱
- 變數名稱要取得有意義，建議每換一個單字就大寫開頭
  - userPhoneNumber
- 事件處理函式名稱以on開頭
  - onclick()

JavaScript基本語法 - 型別(type)

- 型別(type)指的是資料的類型，例如:數值、布林值、字串等
- Javascript屬於弱型別
  - 宣告不需要指定型別
    - var radius =10;
  - 可以隨意更改變數的型別也不會出錯
    - radius = 'hello, world!';

JavaScript基本語法 - 型別(type)

類型	型別
基本型別	數值(number)，例如: 1、3.14、-5、-0.32
	字串(string)，例如: 'Today is Monday'
	布林(boolean)，例如: true/false
	尚未定義值(undefined) ，例如有宣告變數但沒有設定變數的值，就會預設為undefined
	空值(null)，表示沒有值或沒有物件
物件型別	例如函式(function)、陣列(array)、物件(object)

數值型別

- JavaScript的數值採取IEEE 754 Double格式，是一種64位元雙倍精確浮點數
  - 數值範圍為$\pm 2.23 \times 10^{-308}$ to $\pm 1.80 \times 10^{308}$
- 可以使用科學記號
  - 3.847e-3 = 3.847×10^{-3}
- 特殊數值
  - NaN: 非數值，表示不當數值運算
  - Infinity: 正無限大
  - -Infinity: 負無限大

字串型別

- 字串是由一連串字元所組成，可以包含文字、數字、符號等，在字串的前後必須加上單引號或雙引號來做為標示，兩者不能混用
- 跳脫字元

跳脫字元	意義	跳脫字元	意義
\’	單引號	\b	倒退鍵(backspace)
\”	雙引號	\f	換頁(formfeed)
\\	反斜線(\)	\r	歸位(carriage return)
\n	換行字元	\t	Tab鍵

布林型別、未定義型別、空值

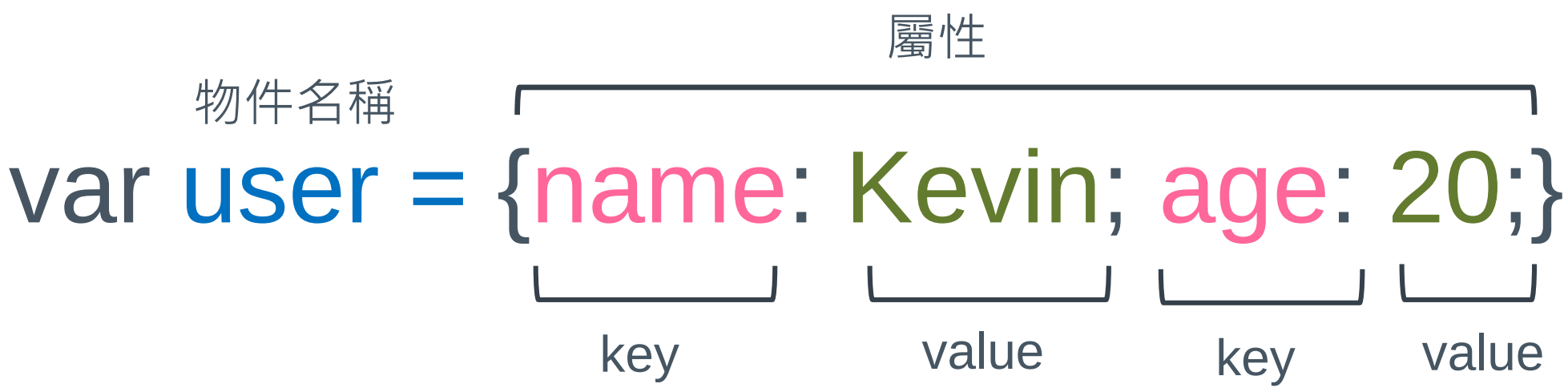
- 布林型別，只有兩種值true/false
  - 通常用在判斷式或運算式是否成立
- 未定義型別
  - 有宣告變數但沒有設定變數的值
  - 存取到尚未定義的屬性會傳回undefined
  - 沒有宣告傳回值的函式會回傳undefined
- 空值
  - 表示沒有值或沒有物件

陣列

- 陣列可以用來儲存多個資料，每個資料在陣列中稱為元素(element)在陣列中有各自的索引(index)和值(value)
- 索引用來識別元素在陣列內的位置
  - 例如第一個元素的索引為0, 第二個元素的索引為1
- 宣告一般陣列 → var A = [10, 20, 30]
  - A[0] = 10, A[1] = 20, A[2] = 30
- 宣告巢狀陣列 → var B = [10, [21, 22], 30]
  - B[0] = 10, B[1] = [21, 22], B[1][0] = 21, B[1][1] = 22

物件

- JavaScript的物件是一種關聯陣列(associative array)
- 陣列儲存的資料稱為元素，物件儲存的資料為屬性
- 物件是用鍵(key)和值(value)的一對屬性來儲存資料



物件

- 存取物件的屬性方法
  - 物件名稱.key
  - 物件名稱.['key']

物件名稱　key

user.age

物件名稱　key

user.['age']

變數

- 在程式的執行過程中，常常需要儲存一些資料，我們就可以使用變數來儲存
- 宣告變數
  - var
- Example
  - var user;
  - var user = “Kevin”
  - var x, y, z;

常數

- 常數和變數一樣可以用來儲存資料，差別在於常數不能重複宣告，也不能重複設定值，可以用來儲存一些不會隨著程式的執行而改變的資料
- 宣告常數
  - const
- Example
  - const PI = 3.14159;
  - const ID1 = 1, ID2 = 2;

算術運算子

運算子	描述	範例
加法 (+)	運算元相加	x+y, x+3
減法 (-)	運算元相減	x-y, x-3
乘法(*)	運算元相乘	x*y, x*3
除法(/)	運算元相除	x/y, x/3
增加 (++)	運算元之前 (++x)，會回傳運算元增加 1 後的值; 在運算元之後(x++)， 會回傳運算元加 1 前的值。	假如 x是 3，那 ++x 將把 x 設定為 4 並回傳 4，而 x++ 會回傳 3 ， 接著才把 x 設定為 4。
減少 (--)	將運算元減少 1。回傳值的情況與增加運算元 相同。	假如 x是 3，那 --x 將把 x 設定為 2 並回傳 2，而 x-- 會回傳 3 ， 接著才把 x 設定為 2。
取餘數 (%)	回傳兩個運算元相除後的餘數。	12 % 5 回傳 2.
指數運算子 (**)	計算以 a 為底的 b 次方， 也就是, a^b	2 ** 3 回傳 8. 10 ** -1 回傳 0.1.

賦值運算子

名稱	簡化的運算子	意義
賦值	x = y	x = y
加法賦值	x += y	x = x + y
減法賦值	x -= y	x = x - y
乘法賦值	x *= y	x = x * y
除法賦值	x /= y	x = x / y
餘數賦值	x %= y	x = x % y
指數賦值	x **= y	x = x ** y

比較運算子

運算子	描述	會回傳 True 的例子
等於 (==)	假如運算元等價就回傳 True。	3 == var1, "3" == var1, 3 == '3'
不等於 (!=)	假如運算元等價就回傳 True。	var1 != 4, var2 != "3"
嚴格等於 (===)	假如運算元具有相同型態且等價則回傳 True。	3 === var1
嚴格不等於 (!==)	假如運算元具有相同型態但不等價，或是具有不同型態，回傳 True。	var1 !== "3“, 3 !== '3'
大於 (>)	假如左方運算元大於右方運算元，回傳 True。	var2 > var1, "12" > 2
大於或等於 (>=)	假如左方運算元大於或等於右方運算元，回傳 True。	var2 >= var1, var1 >= 3
小於 (<)	假如左方運算元小於右方運算元，回傳 True。	var1 < var2, "2" < 12
小於或等於 (<=)	假如左方運算元小於或等於右方運算元，回傳 True。	var1 <= var2, var2 <= 5

邏輯運算子

運算子	使用方法	描述
邏輯 AND (&&)	運算式1 && 運算式2	只有在兩個運算元都是 True 時才會回傳 True，否則回傳 false。
邏輯 OR (||)	運算式1 || 運算式2	||在兩個運算元有任一個是 True 時就會回傳 True，否則回傳 false。
邏輯 NOT (!)	!運算式	假如單一個運算元能被轉換成 True 時，回傳false ， 不然回傳 true。

位元運算子

運算子	用法	描述
位元 AND	a & b	回傳兩個運算元對於每個 bit 做 AND 的結果。
位元 OR	a | b	回傳兩個運算元對於每個 bit 做 OR 的結果。
位元 XOR	a ^ b	回傳兩個運算元對於每個 bit 做 XOR 的結果。
位元 NOT	~ a	將運算元中的每個 bit 反轉(1->0,0->1)
左移	a << b	將 a 的每個 bit 向左移動 b 個 bits，空餘的位數以 0 填滿。
有號右移	a >> b	將 a 的每個 bit 向右移動 b 個 bits，空餘位數以最高位補滿。
以 0 填充的右移	a >>> b	將 a 的每個 bit 向右移動 b 個 bits，空餘的位數以 0 填滿。

字串運算子

- 除了作為比較運算子之外， 運算子 (+) 也能用於字串，將兩字串接在一起，並回傳接在一起後的結果。
  - console.log('我的 ' + '字串'); // 會印出 字串 "我的字串"。
- 也可以使用 += 的賦值運算值在字串變數上
  - var astring = '我的'
  - astring += '字串'

運算子優先級

運算子類型	屬於該類別的運算子
成員 高	. []
呼叫/建立 實例	() new
反向/增加	! ~ - + ++ -- typeof void delete
乘法/除法	* / %
加法/減法	+ -
位元移動	<< >> >>>
關係運算子	< <= > >= in instanceof
相等性	== != === !==
位元 and	&
位元 xor	^
位元 or	|
邏輯 and	&&
邏輯 or	||
條件運算子	?:
指定運算子	= += -= *= /= %= <<= >>= >>>= &= ^= |=
逗點運算子 低	,

流程控制

- 選擇結構
  - 用來檢查條件式，根據結果的值來執行不同的敘述
  - 例如: if…else、switch
- 迴圈結構
  - 用來重複執行指定的敘述
  - 例如: for、while、do…while、for…in
- break和continue指令
  - break為跳出迴圈
  - continue為在迴圈內跳過後面的敘述，直接返回迴圈的開頭

函式

- 函式是將一段具有某種功能或可以重複使用的敘述寫成獨立的程式區塊，供後續呼叫使用
- 一般函式
  - 用來執行一般動作的描述
  - 程式設計人員自行撰寫程式碼來呼叫
- 事件函式
  - 用來處理事件動作，一般是用來回應使用者或系統所觸發的事件
  - 平常處於閒置狀態，當有需要處理時才呼叫

使用者自訂函式

- JavaScript有提供一些內建函式，通常是針對常見的用途，當不夠滿足使用者需求時，使用者可以自行定義函式
- 使用function來宣告函式

Example

```
function 函式名稱(傳入參數){
    描述;
    return 傳回值;
}
```

JavaScript 內建對話視窗函式

- alert()
  - 用來跳出提示 (警告) 對話視窗。
- prompt()
  - 用來跳出一個文字輸入的對話視窗。
- confirm()
  - 用來跳出需要使用者確認的對話視窗，對話視窗中會有確定及取消兩個按鈕。
  - 會返回一個布林值 (boolean)，用來表示使用者按了確定 (true) 或是取消 (false)。

變數的有效範圍

- 全域變數
  - 在函式外面宣告的變數屬於全域變數，程式的所有描述都可以存取這些變數
- 區域變數
  - 在某一個函式裡面使用var來宣告的變數屬於區域變數，只有在該函式裡的敘述可以存去這些變數
- 區塊變數
  - 在某一個區塊(用大括號括起來的區域)內使用let來宣告的變數，只有在該區塊內的描述可以存取這些變數

變數的生命週期與遮蔽效應

- 變數生命週期
  - 全域變數：直譯器會在程式執行完全結束之後才會將它們移除
  - 區域 / 區塊變數：直譯器會在函式中建立這些變數，待函式或區塊執行結束就將它們移除
- 在函式內的有區域變數和全域變數的名稱相同時，在區域內執行描述時會參照區域變數，忽略全域變數

練習

- 建立一個基本html
- 寫一個外部的JavaScript並載入到html
- 這個javascript的功能
  - 找出小於輸入的一個正整數內所有的質數
    - 例如:輸入5, 會找到2, 3, 5三個質數
    - 判斷一個數是否能被自身小的正整數(除了1和自身)整除.如果能整除則不是質數,否則反之
    - var number = prompt(“請輸入正整數", "");
  - 將結果存在一個陣列中並用alert()呈現在網頁上