



# 初體驗

JavaScript

# 角色定位

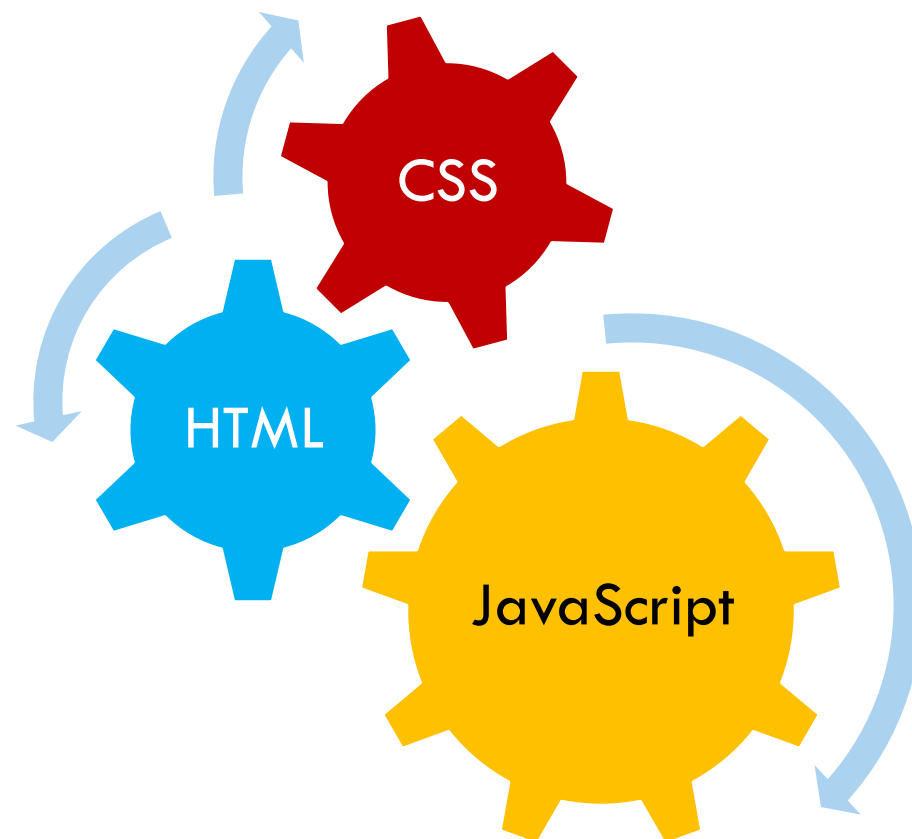
一種在瀏覽器上執行的程式語言

目的：

1. 讓使用者可以與網頁內容互動
2. 透過程式碼控制網頁內容呈現方式

線上文件

- <https://www.w3schools.com>
- <https://developer.mozilla.org>



# 程式區段

JavaScript 程式碼必須放在 `<script></script>` 區段中  
也可以透過 `<script src="外部檔案或網址"></script>` 載入

# 基本輸出（常用於除錯）

使用訊息框

- `alert("Hello, World!")`

使用 console

- `console.log("Hello, World!")`

單行註解 `//`

多行註解 `/* ... */`

補充：

確認框（`confirm`）

```
if (confirm("是否刪除資料")) {  
    console.log("資料已刪除")  
} else {  
    console.log("取消")  
}
```

輸入框（`prompt`）

```
filename = prompt("請輸入檔名")  
if (filename !== null) {  
    console.log(filename)  
}
```

# 與 HTML 互動

```
<html>
<script>
function clickme() {
    var label = document.getElementById("label")
    label.innerHTML = "Hello, World!"
}
</script>
<body>
    <button onclick="clickme()">Click Me</button>
    <div id="label"></div>
</body>
</html>
```

這行可以省略，新語法透過  
id 就可以當物件用

還有哪些 event 可以用？

[https://www.w3schools.com/jsref/dom\\_obj\\_event.asp](https://www.w3schools.com/jsref/dom_obj_event.asp)

# 選取元素

直接使用元素 id：新語法很炫麗

`document.getElementById()`：很常見

`document.querySelector()`：使用選取器，傳回第一個符合

`document.querySelectorAll()`：使用選取器，傳回所有符合

`Document.getElementByClassName()`

`Document.getElementByName()`

`document.getElementsByTagName()`

可以忽略了

# 與 CSS 互動

```
<script>
function setColor(color) {
  let x = document.getElementById("demo")
  x.style.color = color
}
</script>
<body>
  <div id="demo">Hello, World!</div>
  <hr>
  <button onclick="setColor('blue')">Blue</button>
  <button onclick="setColor('green')">Green</button>
  <button onclick="setColor('red')">Red</button>
</body>
```

也可以寫成這樣  
x.style = `color: \${color}`

# 操作 CSS 的 CLASS

增加 class

```
let x = document.getElementById('myid')  
x.classList.add(name)
```

移除 class

```
x.classList.remove(name)
```

Toggle

```
x.classList.toggle(name)
```

查詢

```
console.log(x.classList) // 陣列  
console.log(x.className) // 字串
```



# 要注意網頁上的元件是否誕生了

```
<html>
<script>
  var label = document.getElementById("label")
  label.innerHTML = "Hello, World!"
</script>
<body>
  <div id="label"></div>
</body>
</html>
```

會找不到 label 元件

其中一種解法：將 JS 移到下方

# 另一種寫法1

```
<html>
<script>
function clickme() {
    var label = document.getElementById("label")
    label.innerHTML = "Hello, World!"
}

window.onload = function() {
    document.getElementById("bn").onclick = clickme
}
</script>
<body>
    <button id="bn">Click Me</button>
    <div id="label"></div>
</body>
</html>
```

匿名函數

```
window.onload = function() {
    document.getElementById("bn").onclick = function() {
        document.getElementById("label").innerHTML = "Hello, World!"
    }
}
```

匿名函數

也可寫成 `<body onload="some_function()">`

# 另一種寫法2 – 事件監聽器

```
<html>
<script>
function clickme() {
    let label = document.getElementById("label")
    label.innerHTML = "Hello, World!"
}

window.onload = function() {
    document.getElementById("bn").addEventListener("click", clickme)
}
</script>
<body>
    <button id="bn">Click Me</button>
    <div id="label"></div>
</body>
</html>
```

移除事件監聽器

`removeEventListener("click", clickme)`

# 註冊兩個監聽器

**alert** 是訊息框，提醒使用者注意某些事情時使用  
也常用於除錯

```
<html>
<script>
window.onload = function() {
  document.getElementById("bn").addEventListener("click", function() {
    alert("Hello, World!")
  })
  document.getElementById("bn").addEventListener("click", function() {
    alert("Hi Everyone")
  })
}
</script>
<body>
  <button id="bn">Click Me</button>
</body>
</html>
```


# 變數與常數


JavaScript 變數 **可以** 不用宣告隨叫隨用

若要宣告，可以使用 `var`、`let` 與 `const`

`let` 宣告的變數，有效範圍僅止於某個區段

`var` 可重複宣告，`let` 與 `const` 不行

```
var a = 10;  
var a = 20; 
```

```
var a = 10;  
let a = 20; 
```

```
a = 10 // 相當於 var  
var b = 20  
let c = 30  
const d = 40
```

# USE STRICT 模式

當設定 `use strict` 模式後，變數一定要宣告才能使用  
要放在全域或區域的第一行才有作用

```
'use strict'  
a = 10 // throws ERROR
```



# 資料型態

# 為何資料要有型態？

資料型態決定該資料有哪些功能

不同型態的資料**原則上**必須轉成同一型態才能後續處理

資料型態分為兩種：純量型態與非純量型態（e.g., 物件、結構、陣列...）

JS 的型態屬於推論型態，由初始化內容決定其型態

特殊值

- NaN：通常在數學運算失敗時出現，例如 `parseInt("abc")`
- null：在 JS，此值通常來自於資料庫或是 JSON 中的資料
- undefined：當變數沒有初始化時



# 資料型態

共有八種型態

- String : "Hello, World!"
- Number : 10、3.14
- BigInt : 超極大的整數
- Boolean : true / false
- Undefined : 尚未定義的變數
- Null : 空值
- Symbol
- Object :
  - object
  - array
  - date

判斷使用者在文字框是否輸入數字

```
s = "abc"
n = parseInt(s)
if (isNaN(n)) {
  console.log("不是數字")
}
```

object 型態

```
obj = {
  "name": "David",
  "age": 30
}
console.log(obj.name)
```

# 型態檢查

使用 `typeof` 檢查型態

回傳

- string
- number
- boolean
- object
- function

判斷 NaN 用 `isNaN()`

判斷 null 用 `=== null`

```
console.log(typeof "hello") // Prints "string"
console.log(typeof 123)     // Prints "number"
console.log(typeof [10])    // Prints "object"
console.log(typeof function(){} ) // Prints "function"
```

# 型態轉換

## 字串轉數字

- `var n = parseInt("10")`
- `var f = parseFloat("3.14")`

## 數字轉字串

- `var s = String(10)`

## 轉布林

- `var b = Boolean(0) // false`
- 除 0 以外其他都是 `true`

# 練習

網頁上有兩個輸入框讓使用者輸入數字，當按鈕按下後將這兩個數字相加後顯示到網頁上

提示：字串轉數字函數為 `parseInt()`、`parseFloat()`

例如：`n = parseInt("10")`



# 流程控制 - 條件判斷與迴圈

# IF

if 語句，用來決定什麼情況要執行哪些程式碼

condition 位置為「邏輯運算式」，運算結果只有兩種 `true` / `false`，稱為布林值

```
if (condition1) {  
  
} else if (condition2) {  
  
} else {  
  
}
```

邏輯運算子	
and	&&
or	
not	!

比較運算子	
大於	>
大於等於	>=
小於	<
小於等於	<=
一般等於	==
嚴格等於	===
一般不等於	!=
嚴格不等於	!==

```
a = 10  
b = "10"  
console.log(a == b) // true  
console.log(a === b) // false
```

x	y	==	===
undefined	undefined	true	true
null	null	true	true
true	true	true	true
false	false	true	true
'foo'	'foo'	true	true
0	0	true	true
+0	-0	true	true
+0	0	true	true
-0	0	true	true
0	false	true	false
" "	false	true	false
" "	0	true	false
'0'	0	true	false
'17'	17	true	false
[1, 2]	'1,2'	true	false
new String('foo')	'foo'	true	false
null	undefined	true	false

a === b

a == b

	true	false	1	0	-1	"true"	"false"	"1"	"0"	"-1"	" "	null	undefined	Infinity	-Infinity	()	{}	[[ ]]	[0]	[1]	NaN
true	■																				
false		■																			
1			■																		
0				■																	
-1					■																
"true"						■															
"false"							■														
"1"								■													
"0"									■												
"-1"										■											
" "											■										
null												■									
undefined													■								
Infinity														■							
-Infinity															■						
()																■					
{}																	■				
[[ ]]																		■			
[0]																			■		
[1]																				■	
NaN																					■

	true	false	1	0	-1	"true"	"false"	"1"	"0"	"-1"	" "	null	undefined	Infinity	-Infinity	()	{}	[[ ]]	[0]	[1]	NaN
true	■		■					■												■	
false		■		■					■								■		■	■	
1			■					■												■	
0				■					■										■	■	
-1					■					■											
"true"						■															
"false"							■														
"1"								■													■
"0"									■											■	
"-1"										■											
" "											■										
null												■	■								
undefined													■	■							
Infinity														■							
-Infinity															■						
()																■					
{}																	■				
[[ ]]																		■			
[0]																			■		
[1]																				■	
NaN																					■

# SWITCH CASE

```
let weather = "rainy"
switch (weather) {
  case "rainy":
    console.log("rain")
    break
  case "sunny":
    console.log("sunny")
    break
  default:
    console.log("others")
}
```



# 迴圈

## For LOOP

`for( ; ; )`

- `for(let i = 0; i < 10; i++) { ... }`

`for(let index in array)`

- `for (let index in ["a", "b", "c"]) { ... }`

`for(let el of array)`

- `for (let el of ["a", "b", "c"]) { ... }`

## While LOOP

`i = 0`

```
while (i < 10) {  
  console.log(i)  
  i += 1  
}
```

無窮迴圈 ( infinite loop )

`for(;;) { ... }`

`while (true) { ... }`

# 迴圈的半途而廢

`continue`：這回到此為止，立刻進入下一回合

`break`：直接離開迴圈不玩了

可用在 `for` 與 `while` 迴圈

```
for(let i = 0; i < 10; i++) {  
  if (i === 3) {  
    continue  
  }  
  console.log(i)  
}
```

```
for(let i = 0; i < 10; i++) {  
  if (i === 3) {  
    break  
  }  
  console.log(i)  
}
```

# 巢狀迴圈

泡泡排序 ( bubble sort )

```
a = [5, 7, 2, 3, 1]
for (let i = 0; i < a.length - 1; i++) {
  for (let j = i; j < a.length; j++) {
    if (a[i] > a[j]) {
      tmp = a[i]
      a[i] = a[j]
      a[j] = tmp
    }
  }
}
console.log(a) // [1, 2, 3, 5, 7]
```

# 練習

1. 使用者在網頁上輸入兩個數字做除法運算，必須檢查分母是否為 0，為 0 時必須告訴使用者分母不可輸入 0
  - 提示：`type="number"`
2. 隨機產生 5 組 1 ~ 40 間 6 個不重複的整數數字
  - 提示：`Math.random()`、`Math.ceil()`、`Math.floor()`
3. 九九乘法表
  - 提示：巢狀迴圈



# 字串

# 字串

字串長度 `length`

- `"hello".length`

單引號、雙引號都可以

跳脫字元 \

- `\n`、`\\`、`\`

字串中夾帶變數

```
name = 'David'  
str = `Good morning ${name}`  
console.log(str);  
// Prints "Good morning David"
```

backtick

# 多行字串

使用 backtick

```
s = `
  第一行
    第二行
      第三行
`
console.log(s);
```

# 常用函數

## 子字串

- `substr(from, len)`
- `substring(from, to)`

## 字串取代

- `replace`：只換第一個
- `replaceAll`：全部換
- 範例：`"hello".replace("ll", "zz")`

## 大小寫

- `toUpperCase()`
- `toLowerCase()`

## 頭尾去空白

- `trim()`

## 分割

- `split(char)`
- 例如：`arr = "David,Tom,Mei".split(",")`

## 搜尋

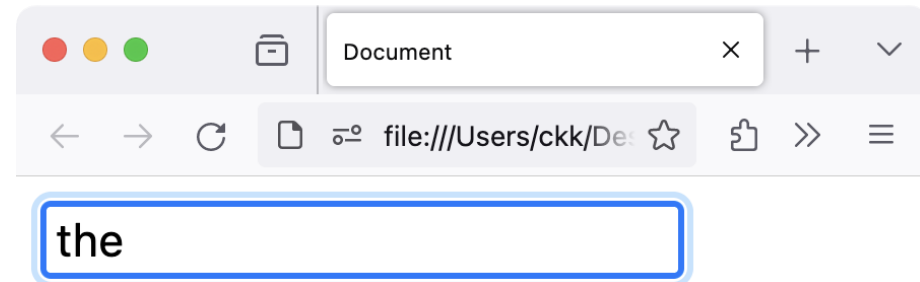
- `indexOf(str, [start])`



# 練習

畫面上有一段文章以及一個文字輸入框用來做文字搜尋

如果文章中有符合使用者輸入的文字時，該文字會高亮度顯示



While scouring Argentina's La Colonia Formation for new dinosaur fossils, paleontologists noticed **the** single toe bone sticking out of **the** ancient rock. When **they** dug in, **they** found a new dinosaur—a carnivore that roamed prehistoric Patagonia several million years before an asteroid impact brought **the** Cretaceous to a fiery close.