

# JavaScript 程式語言

---



# JavaScript 簡介

製作動態網頁的基礎

- 主要用在網頁前端開發
- Java 和 JavaScript 完全不一樣



# JavaScript 快速開始

將程式寫在 `<script>` 標籤中即可

- 簡單的範例

```
<script>
```

```
    console.log("Hello JavaScript");
```

```
</script>
```

---

# 練習：撰寫、測試 JavaScript 程式

### 3. 資料、資料型態



# 註解

會被電腦忽略的文字，寫給人看的

- 單行註解使用 `//`  
`//` 這是單行註解
- 多行註解使用 `/*` 和 `*/` 包裹  
`/*`  
這是多行註解  
第二行  
`*/`



# 什麼是資料


程式中最基本的運作單位

- 所有程式都是由最基本的資料組成
- 資料分成很多種型態 (種類)

# 資料型態

—






# 數字

表達數字，整數或是小數 (浮點數)

- 數字型態的資料
  - 3
  - -50
  - 3.1415926
  - -1.244



# 數字

表達數字，整數或是小數 (浮點數)

- 數字型態的資料
  - 3
  - -50
  - 3.1415926
  - -1.244



# 字串

表達任意的文字

- 用雙引號、或單引號包裹的文字
  - "Hello"
  - "你好"
  - '單引號也沒問題'



# 布林值

表達正確或是錯誤的二分概念


- 兩個特殊的關鍵字
  - true
  - false



# 空值

表達空的概念

- 兩個特殊的關鍵字
  - undefined
  - null



# 資料型態總覽

資料型態在運算中扮演關鍵角色

- 以下是 JavaScript 的主要資料型態
  - 數字
  - 字串
  - 布林值
  - 空值
  - 物件 Object

---

練習：印出各式各樣不同的資料

## 4. 變數與常數





# 什麼是變數

可存放資料、可命名的空間

- 變數的運作流程
  - 宣告變數
  - 使用變數



# 宣告變數

建立、命名一個空間

- 宣告變數語法

`let` 變數名稱

- 程式範例

`let x;`

`let name;`



## 使用變數

根據變數名稱取得存放的資料

- 使用已經宣告過的變數名稱
- 程式範例

```
let x;  
console.log(x);
```



## 指定變數中的資料

將新的資料放進變數中

- 指定資料語法  
let 變數名稱=資料  
宣告過的變數名稱=資料
- 程式範例  
let x;  
console.log(x);  
x=5;  
console.log(x);  
x="Hello";  
console.log(x);



## 指定變數中的資料

將新的資料放進變數中

- 指定資料語法

let 變數名稱=資料

宣告過的變數名稱=資料

- 程式範例

```
let x;
```

```
console.log(x); //undeinfed
```

```
x=5;
```

```
console.log(x); //5
```

```
x="Hello";
```

```
console.log(x); //Hello
```



## 常數 Constant

不能夠更動資料的變數，稱為常數

- 宣告常數語法  
`const` 常數名稱=資料
- 程式範例  
`const x=3;`  
`x=100;` // 錯誤的程式

---

## 練習：透過變數操作資料

## 5. 運算符號 Operator





# 什麼是運算符號

可以對資料進行某種操作的符號


- 算術運算
- 指定運算
- 比較運算
- 單元運算
- 邏輯運算



# 算術運算

基本的加減乘除、模運算


- 加法： $+$
- 減法： $-$
- 乘法： $*$
- 除法： $/$
- 模運算： $\%$



## 指定運算

將資料存放進變數中


- 基本： $=$
- 搭配算術運算使用：
  - $+=$
  - $-=$
  - $*=$
  - $/=$
  - $\%=$



## 比較運算

比較資料的大小，得到布林值

- 大於： $>$
- 小於： $<$
- 大於等於： $>=$
- 小於等於： $<=$
- 是否相等： $==$



# 邏輯運算

針對布林值的運算

- 且 (AND) : &&

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

- 或 (OR) : ||

A	B	A    B
true	true	true
true	false	true
false	true	true
false	false	false

---

## 練習：各種運算符號的操作

## 6. 流程控制：判斷式



# 什麼是流程控制

讓程式的運作流程更多變

- 判斷式  
建立邏輯判斷與對應的處理
- 迴圈  
可重複執行的程式區塊





## 程式區塊

用大括號 { 和 } 包裹起來的程式碼

- 範例如下

```
{  
    console.log("Hello");  
    console.log("World");  
}
```

判斷式



# if 判斷式

根據條件決定是否要執行程式區塊

- 基本語法

```
if(布林值){
```

如果布林值為 true，執行這個程式區塊

```
}
```



## if ... else 判斷式

條件成立與否，都有對應的程式區塊

- 基本語法

```
if(布林值){
```

如果布林值為 true，執行這個程式區塊

```
}else{
```

如果布林值為 false，執行這個程式區塊

```
}
```



## if ... else if ... else 判斷式

多個條件對應多個程式區塊

- 基本語法

```
if(布林值){
```

如果對應布林值為 true，執行這個程式區塊

```
}else if(布林值){
```

如果對應布林值為 true，執行這個程式區塊

```
}else if(布林值){
```

如果對應布林值為 true，執行這個程式區塊

```
}...{
```

如果對應布林值為 true，執行這個程式區塊

```
}else{
```

如果上方布林值都是 false，執行這個程式區塊

```
}
```

---

## 練習：判斷式的基本操作和運用

## 7. 流程控制：迴圈

# 流程控制：迴圈





# while 迴圈

## while 迴圈

根據條件決定是否要重複執行程式區塊

- 基本語法

```
while(布林值){
```

如果布林值為 true，執行這個程式區塊

執行完畢後，跳到迴圈開頭，重新再來一次

```
}
```



## while 迴圈範例

搭配變數和判斷式運作

- 基本範例

```
let n=0;
while(n<3){
    console.log(n);
    n++;
}
```

# for 迴圈

## for 迴圈

while 迴圈的囉哩吧說版

- 基本語法

```
for(初始化程式;布林值;每次轉圈時執行){
```

如果布林值為 true，執行這個程式區塊

執行完畢後，跳到迴圈開頭，重新再來一次

```
}
```



# for 迴圈範例

搭配變數和判斷式運作

- while 迴圈基本範例

```
let n=0;
while(n<3){
    console.log(n);
    n++;
}
```

- for 迴圈基本範例

```
for(let n=0;n<3;n++){
    console.log(n);
}
```

## 練習：迴圈的基本操作和運用

## 8. 流程控制：迴圈指令

# 流程控制：迴圈指令



## 迴圈中的指令

只能寫在迴圈中，否則產生錯誤

- 強迫中斷迴圈 `break`
- 強迫進入下一圈 `continue`



# break 指令

強迫中斷迴圈

- 使用範例

```
let n=0;
while(n<3){
    if(n==1){
        break;
    }
    console.log(n);
    n++;
}
```





## continue 指令

強迫跳到迴圈開頭，進入下一圈

- 使用範例

```
for(let n=0;n<3;n++){  
    if(n==1){  
        continue;  
    }  
    console.log(n);  
}
```

只印出

0

2

## 練習：迴圈指令的基本操作