函式的基本觀念

函式 Function

包裝好的程式區塊

- 函式的使用流程
  - 建立函式
  - 呼叫函式

建立並呼叫函式語法

函式內部的程式碼不會立即執行

- 建立函式

```
function 函式名稱(參數名稱列表){
        函式內部的程式區塊
}
```

- 呼叫函式

```
函式名稱(參數資料列表);
```

無參數的範例

透過函式呼叫，執行函式內部的程式碼

- 建立函式

```
function test(){
        console.log("Hello");
}
```

- 呼叫函式

```
test();
test();
```

單一參數的範例

透過函式呼叫，執行函式內部的程式碼

- 建立函式

```
function show(msg){
        console.log(msg);
}
```

- 呼叫函式

```
show("Hello");
show("你好");
show(true);
```

多個參數的範例

參數之間用逗號隔開

- 建立函式

```
function add(n1, n2){
        let result=n1+n2;
        console.log(result);
}
```

- 呼叫函式

```
add(3, 4);
add(10, -5);
add("Hello", "World");
```

建立函式的另一種寫法

建立函式

常見的另一種函式寫法

- 原來的寫法

```
function 函式名稱(參數名稱列表){
        函式內部的程式區塊
}
```

- 另一種寫法

```
let 函式名稱=function(參數名稱列表){
        函式內部的程式區塊
}
```

另一種寫法的範例

隱含的意義：函式名稱就是變數名稱

- 建立函式

```
let show=function(msg){
        console.log(msg);
}
```

- 呼叫函式

```
show("Hello");
show("你好");
show(true);
```

練習：函式的建立、呼叫與參數

函式的學習重點

談到函式就必須注意的事情

- 建立函式、呼叫函式的基本流程
- 參數的運用
- 回傳值的運用

函式的結束與回傳值

函式結束的語法

使用 return 結束函式，並帶出回傳值

- 函式結束，帶出回傳值 undefined

```
function 函式名稱(參數名稱列表){
        函式內部的程式區塊
        return;
}
```

- 函式結束，帶出自訂的回傳值

```
function 函式名稱(參數名稱列表){
        函式內部的程式區塊
        return 回傳值;
}
```

函式自然結束

函式內的程式碼執行完畢，自然結束

- 函式自然結束，帶出回傳值 undefined

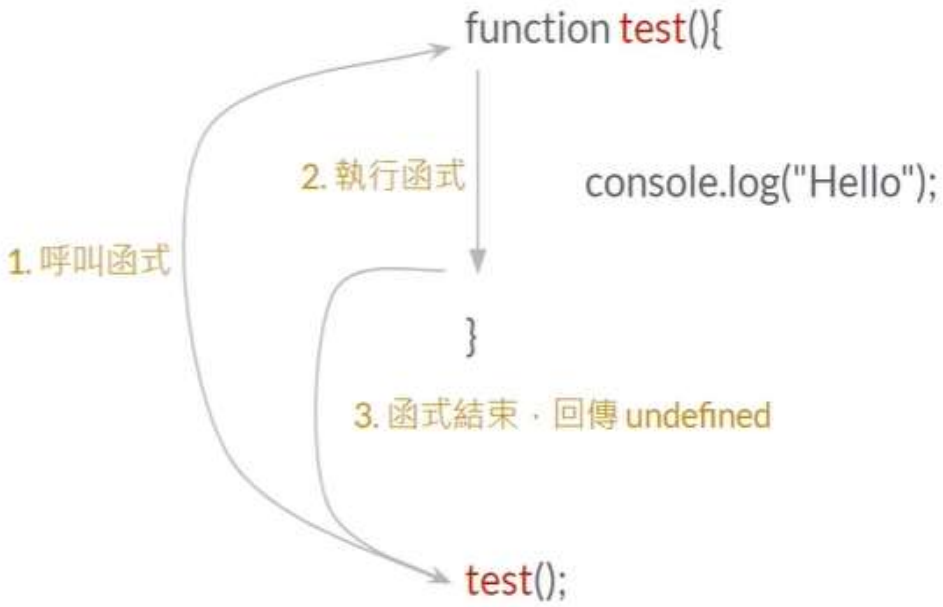
```
function 函式名稱(參數名稱列表){
        函式內部的程式區塊
}
```

- 相當於

```
function 函式名稱(參數名稱列表){
        函式內部的程式區塊
        return;
}
```

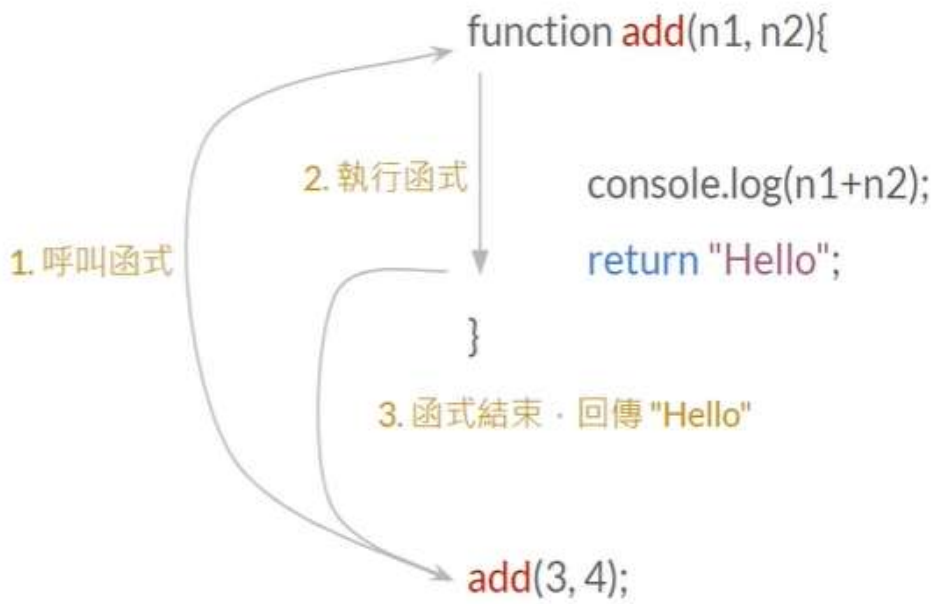
回傳值的運作

將資料從函式內部傳回呼叫函式的位置



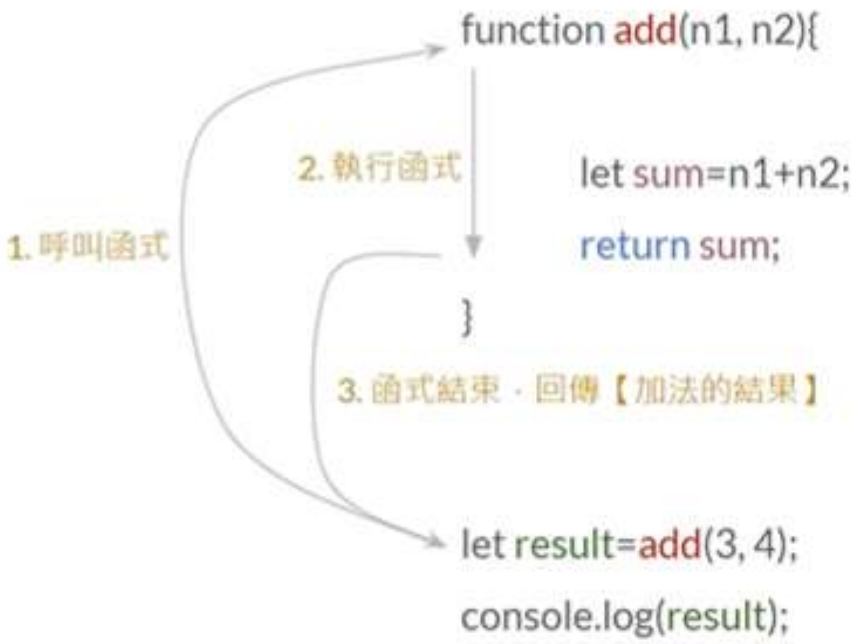
回傳值的運作

函式回傳值只和 return 命令有關



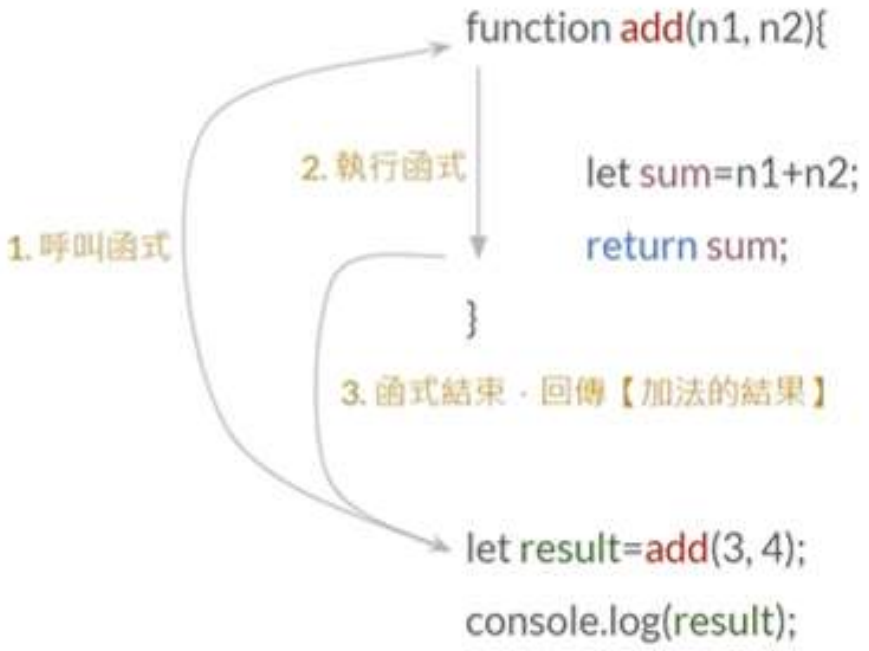
回傳值的運用

設計有意義的回傳值



回傳值的運用

設計有意義的回傳值



練習：熟悉、運用函式的回傳值