js 宣告命名有三種

第一種 (camel 是一個單字 Case 也是一個單字 第二個單字開頭要大寫)

camelCase

第二種 (在兩個單字之間加一個下滑線)

camel_case

第三種(都打大寫 專門用在 const 代表不變函數)

PI = 3.14159

例如:

const PI = 3.14159;

console.log(PI);

js 註解三種方法

第一種

// 1. one line comment

//123456789

//456789123

第二種

/*

132465789

12456789

*/

第三種

/**

*123456 案 enter 會自動加星號不用用手打

*564321

*/

Nan 代表 not a number js 不知道這是捨麼數字

concat 代表串接

Strings 代表字串

常用函數

console.log("123456")

alert()跳出框

prompt() 跳出輸入框

若字串與數字加在一起會變成 concat

strings + number = concat

number = > 第一步會先把 number 變成 strings

+= > 第二步會把 會把 number 與 strings 串接在一起

例如:let n1 = 20;

let n2 = 30;

let name2 = "johnson";

let n3 = 10;

let n4 = 15;

console.log(n1 + n2 + name2 + n3 + n4); "50johnson1015"

變數: (= let const var)

const 表示不變的數

若在 js 裡面

const x = 6 ;

x = 7 ; js 裡面會顯示 error 因為 const 是不變的

let 的話表示

(1)

let x= 8;

x=9;

若

(2)

let x = 9 ;

let x = 10;

(1)可以 (2)不行

運算符號

remainder 取餘方法

5/2=2...1

5%2=2....1

17%3=2

poerator 取次方方法

(** 符號 為取次方)

console.log(7**4)

++遞增 - - 遞減(跟遞增一樣)

let x = 0 ;

x++;

x++;

x++;

x = 3

因為 x++會增加 1 所以三個 x++ 所以會等於=3

+= , -+ , *+ , /+(原理都一樣)

let x = 0;

/x = x + 10;/

x += 10;

x = 10

/x = x + 10;/

x += 10;

(x = x + 10; 把 x 去掉 + = 倒過來 就變成 x += 10;)

Primitive Data Types (js 當中最基本"6 種"js 儲存資料的方式)

number 數字

第一個 toSrting 用法

let age = 27

toSrting 表示把 number 改變型態 變成字串

console.log(age.toSrting()+10); 測試 若 age 有變成字串 輸出會變成 2710 若沒會變成 37

第二個 toFixed 用法

let pi = 3.1415926533;

console.log(pi.toFixed(3));toFixed 表示要取小數點後面幾個數字若我打 3 會取 3.141

若打 5 會取 3.14159

注意!pi.toFixed 要寫在 console.log 裡面否則若打在外面 一樣會變成 3.1415926533 因為 toFixed 不會改變 pi 裡面的數值

let pi = 3.1415926533;

pi.toFixed(3);

console.log(pi)

---

string 字串

Length

let name = "hello lin";

console.log(name.length) ;

取得字串的長度

hello lin 因為 hello lin 中間有加空格 所以也會算進去 hello lin 有 9 個字 加空格變成 10 個字 所以 log 裡面會顯示 10 表示 10 個字

Index

let name = "hello";

console.log(name[4]);

因為 index[ ] 是從 0 計算

h 是 0 e 是 1 l 是 2 l 是 3 o 是 4

所以會顯示 o

注意!

length 計算是從 1 開始

index 計算是從 0 開始

Slice

let name = "abcdefg";

console.log(name.slice(1,3))

取得字串裡面的字

name.slice(1,3)代表取得字串裡面的 1 跟 3 的字

會取得 b 跟 c 不會取得到 d

第一個為 Begin index Inclusive

所以會取得 b 的字 代表起始

第二個為 End Index Exclusiver

所以會取得 c 的字 代表結束

為什麼不會取得 d 因為 End Index 代表

排除 所以會排除 d 所以會得到前一個的字為 c

例如 1:

let name = "numvvvv;

console.log(name.slice(1,3))

所以會取得 u 跟 m

3 也代表 v 字 所以 v 會被排除

例如 2:

let name = "abcdefghijklmn";

console.log(name.slice(4,7));

所以會取得 e 跟 g

7 代表 h 所以會 h 會被排除掉

所以會取得 e 跟 g

index of

let name = "hellooo wordl";

console.log(name.indexOf("e"));

indexOf 代表取得字串位置

e 在 1 的位置

若是我打 o 的話

會取得 4 因為 o 在 4 的位置

若是我直接打 console.log(name.indexOf("ooo"));

他會直接取得第一個字的位置所以會取得 4

注意! 他會區分大小寫,若是打錯或是大小寫寫錯會顯示-1

若是不區分大小寫可以寫成這樣

let name = "ABcdefg";

console.log(name.toLowerCase().indexOf("b"));

因為:toLowerCase 代表把字串的大寫全換成小寫,所以就可以搜尋

toLowerCase

代表把全部的大寫換成小寫

let name = "ABcdefg";

console.log(name.toLowerCase())

所以會變成 abcdefg

toUpperCase

代表把全部的小寫換成大寫

let name = "ABcdefg";

console.log(name.toUpperCase())

所以會變成 ABCDEFG

split

把字串換成 array

let name = "ABCDEFG abcde abbbbbb"; console.log(name.split(" "));

會取得 Array(3) ["ABCDEFG", "abcde", "abbbbbb"]

若是("") ""裡面沒打空白會變成

Array(21) ["A", "B", "C", "D", "E", "F", "G", " ", "a", "b", …]

把每一個字變成 array\

---

boolean

true 為真

false 為假

---

undefined 表示未定義

例如:

let x ;

console.log(x);

會顯示 undefined

因為未定義 x 的值

null

let x = null;

console.log(x);

會顯示 null

因為 null 代表甚麼都沒有 undefined 是表示在等待一個值給他

---

typeof

若想查詢資料狀態,可以使用 typeof 查詢

例如

let x = "true";

let y = true;

console.log(typeof x) 會顯示 string 因為有加 " "

console.log(typeof y) 會顯示 boolean

Logical Operator

== , ===

==代表比較左右是否有相等

例如

console.log(3=="3");

會顯示 true

因為只會看本身的值是否有一樣

不會去看本身的型態是否一樣

===代表比較左右資料類型和本身的值是否一樣

例如

console.log(3 == "3");

會顯示 false 因為"3"是字串

若改成(3 == 3) 會顯示 true

因為都是 number 型態而數字也是一樣

! = , ! ==

<true 代表等於 false 代表不等於>

! = 代表不等於

console.log(4 ! = "4");

會顯示 false

因為 4 等於 4

所以會顯示 false

因為不會去比較型態只會比較本身的值是否不等於

! = = 代表比較左右資料類型和本身的值是否不等於

console.log(4! = "4");

會顯示 true

因為本身會去比較型態與本身的值是否不等於

而"4" 是字串不是 number 所以會顯示 true

< , > , > = , < =

:代表大於 <:代表小於

console.log(4>3);

會顯示 true 因為 4 大於 3

&& ‖

&& 代表 中文的<和> 代表左右的值都要 boolen

‖ 代表 中文的<或> 代表左右只要一個 boolen 有成立就可以

let x = true;let

y = false;

console.log(x ‖ y);true

console.log(x && y);false

---

if else 用法

若 if 條件不成立 會跳到 else

if(條件){

}else{

}

(1)

let name = true;

if(name == true){

console.log("hello!!")

}else{

console.log("not ")

}

(2)

若在 if(name)不打 == 或者其他比較符號的話,會自動判斷 true 或者 false

第一行會判斷是否 true

let name = false;

if(name){

console.log("hello!!")

}else{

console.log("not ")

}

因為 name 是 false 會顯示"not"

---

isnNaN

表示值會確認是不是 NaN

例如:

let age = prompt("請輸入年齡");

age = Number(age); 若在 js 裡面 promp 所輸入的數字都是字串,所以要先改變成 number

if(isNaN(age)){

上面那段意思就是雖然我打中文(二十)可是 age 已改成 number 所以會變成 Nan

alert("請輸入數字");

}else{

alert("是數字沒錯");

}

若我打(二十)會顯示(請輸入數字),因為 age 改成 number 就算打國字也會自動轉成 NaN

若我打 20 會顯示("是數字沒錯")因為 20 是 number

而不能使用 age == NaN or age ===NaN 否則還是會顯示("是數字沒錯")

Truthy and Falsy Values

在 boolean 裡面若遇到一律都是 false

false 0 " " null undefined NaN

Array

index 表示搜尋 array 裡面數組

let name = ["a","b","c","d"];

console.log(name[2]);

上面代表搜尋 name 裡面的第二個數組

0 代表 a 1 代表 b 2 代表 c 所以會顯示 c

---

length

let name = ["a","b","c","d"];

console.log(name.length);

搜尋 name 裡面有幾個數組 ""為一組,"a","b","c","d" 所以有四組

---

push()

let name = ["a","b","c","d"];

name.push("e");

console.log(name);

在數組最後一個新增一個數組

本來是 abcd 四個數組 而加了 name.push("e"); 代表在數組再增加一個

所以會顯示 abcde

---

pop()

let name = ["a","b","c","d"];

name.pop();

console.log(name);

刪除數組最後一個數組

本來是 abcd 而加了 name.pop();代表會刪除最後一個數組

所以會顯示 abc

---

shift()

let name = ["a","b","c","d"];

name.shift();

console.log(name);

刪除數組前面第一個數組

本來是 abcd 而加了 name.shift();代表會刪除前面第一個數組

所以會顯示 bcd

---

unshift()

let name = ["a","b","c","d"];

name.unshift("aaa")

console.log(name);

在數組前面增加一個數組

本來是 abcd 而加了 name.unshift("aaa");代表會在第一個數組

前面再增加一個數組所以會變成"aaa","a","b","c","d"

---

function 函數

表示含有一系列的程式碼,例如常用的 console.log(),alert()....

兩個一定要認識的參數

parameter

例一

在 array 命名一個 friends

friends 裡面本來的值有"a1","a2","a3"

然後新增 friends.push("a4")

這個剛新增的 a4 就教函數

例二

function saiHi(name,age){

console.log("Hi");console.log("my name is"+name+".")

console.log("age:"+age+".")

}

saiHi(" lin","21")

saiHi 裡面的 name 跟 age 就是我們把值丟進去就是一個函數

只要在 saiHi(" lin","21") 括號裡面輸入任意一個值("lin",21)

他就會接收到把值傳進 name 跟 age 裡面

所以會顯示 my name is lin .

age : 21

return

很像"丟出來"概念

例如:

function convert(oc){

return oc * 1.8 +32;

}

上段意思是說 return 會把 oc * 1.8 +32;丟出來

let of = convert(0) 然後要用 let of 去接 return 出來的 oc * 1.8 +32 * 0

covert(32) 然後 32 會去跟 1.8 +32 * 0 做加減乘除

範例:

```
function convert(oc){

return oc * 1.8 +32; }

let input = prompt("請輸入溫度");

let result = convert(input);  去接住 return 出來的 oc * 1.8 +32;做處理

alert(result+"度");
```

return 裡面的 oc * 1.8 +32;丟出來 讓 result 去接 然後 prompt 裡面輸入的數字去讓 result 去處理

---

object 物件 (object 是用{ } array 適用[ ])

property

第一個為使用[ ] 搜尋 object 物件名子 獲得物件裡面的資料

[ ] ,dot notation

```
let isName ={

first_name:"wlison",

last_name:"ren",

age:32,

is_married:"false",

spouse:null}

console.log(isName["spouse"]);
```

第二個為用 (. 點) 搜尋 object 物件名子 獲得物件裡面的資料

```
let isName ={
```

first_name:"wlison",

last_name:"ren",

age:32,

is_married:"false",

spouse:null}

console.log(isName.age)

---

method

let isName = {first_name:"wlison",

last_name:"ren",

age:32,

is_married:"false",

spouse:null,

sayHi(){

console.log("hello");

},sayHiii(){

console.log("sayhi hello");

}

}isName.sayHi();

isName.sayHiii();

在 isName 裡面新增一個 sayHi() 跟 sayHiii()

裡面放置"hello","sayhi hello"文字

然後在外部輸入一個 isName.sayHi(); 和 isName.sayHiii();

程式然後會去搜尋

isName 裡面的 sayHi() 和 isName.sayHiii() 所以會顯示"hello"與

sayhi hello

也可以使用

let isName = {first_name:"wlison",

last_name:"ren",

age:32,

is_married:"false",

spouse:null,

say(world){

console.log("my name is :" + world +"。")

}}

isName.say("lin")

上段程式碼我在 say()裡面加入一個 world 變成 say(world)

然後在外部 打上 isName.say("lin"),然後"lin"會傳入 world

裡面,所以會顯示 my name is :lin。

---

this

let isName = {

first_name:"wlison",

last_name:"ren",

age:32,

is_married:"false",

spouse:null,

sayage(){

console.log(this.age);

} }

isName.sayage();

this 代表會指向本身的 object

而 sayage()本身就是 isName object 裡面的

所以

sayage(){

console.log(this.age); 這段 this.age 會自動找 isName 物件的 age

} }

所以會顯示 32

迴圈

for loop

for(let i = 0; i<10;i++){

console.log(i)

}

i 一開始先設定初始值為 0

i 若小於 10 的話

i++代表會一值增加

例如 i = 1 有大於 10 嗎 沒有

然後 1+1=2 有大於一嗎 沒有

然後一直加到大於 10 為止

會一直加是 i++的關係

若沒打 i++會一直無限循環

---

while loop

let j = 0;

while(j<=10) {

console.log(j)

j++

}

跳脫出來

---

continue

for(let i = 0; i<10;i++)

{

if(i == 5){

continue;

}

console.log(i)}

continue 意思就是說 if(i == 5)的話會直接跳

脫 5 的數字意思就是說略過 5

所以會顯示 1,2,3,4,6,7,8,9

---

break

for(let i = 0; i<10;i++){

if(i == 5){

break;

}

console.log(i) }

break 的意思是說 if(i == 5)會直接跳脫不會再下去了

跟 continue 很像 continue 是跳過 5 再繼續加下去

可是 break 會直接跳脫不會再繼續加下去

所以會顯示 1,2,3,4

(重要!)

let a1 =["a1","a2","a3"];

for(let x =0 ; x<a1.length;x++){

console.log(a1[x]);

}

第一段先 array a1 裡面有"a1","a2","a3" 3 個值

第二段為 x<a1. length;x++ 而為什麼要<a1.length

因為 array 是從 0 開始 而 array 有 3 個值

而 x 因為設為 0,for 會跑三次 (0.1.2)

除非你用 x<=a1.length 才會跑四次(0.1.2.3)

所以 for 會跑 0 1 2 剛好對應 array 的 0 1 2

Math object

Math.PI

console.log(Math.PI);

算出圓周率

Math.pow(a,b)

let a1 = Math.pow(2,10);

console.log(a1)

算出次方

Math.rendom()

for(let x = 0 ; x<10;x++){

console.log(Math.random())

}

若是想要 1~100 的話可以變成

console.log(Math.random() * 100)

若是想去除後面小數點可以改成

console.log(Math.floor(Math.random() * 100));

產生隨機亂數

Math.sqrt()

let x = Math.sqrt(64)

console.log(x)

算出平方根

所以會顯示 8

Math.abs()

let x = Math.abs(-666)

console.log(x)

算出絕對值就算是負值也會變正值,正值還是正值不會變負值

所以會顯示 666

Math.floor

let x = Math.floor(666.3333) ;

console.log(x)

會去除數字後的小數點 666.3333 變成 6666

ceil

let x = Math.ceil(666.1)

console.log(x)

只要數字後面有小數點就會進一位

例如 666.1 後面有小數點會變成 667

668.3 為變成 669

若是 661 後面沒小數點還是一樣 661

```
let answer = Math.floor(Math.random()*100);

let n1 = 0;
let n2 = 99;

while(true){
let guess = prompt("請輸入" + n1 +"到" + n2);
if(guess < n1 || guess > n2){
         alert("數字 0~99");
         continue;
}
if(guess == answer){
        alert("恭喜你猜對了數字是" + answer + ".");
        break;
    }else if(guess <= answer){
            n1 = guess;


}else if(guess >= answer){
        n2 = guess;
    }


}
```

解說

先讓 answer 產生亂數 1~99 floor 表示不要亂數後面小數點

//若 guess 小於 n1 或 guess 大於 n2 會顯示 數字 0~99//

/而 continue 代表就是 會一直 run 下去直到輸入的數字正確為止 而不用 break 原因是會直接跳脫迴圈,而不能再繼續猜

/n1 = guess 表示說若 prompt 輸入的文字大於 answer(亂數)會直接把輸入的數字傳入 n1 而 n1 這個值 並不會還是 0/

/n2 原理跟 n1 依樣/

經典演算法

```
let friends = ["h","Harry", "Ron", "Snap"];
let reversed_friends = []
for(let i = friends.length-1;i>=0;i--){
/*為什麼 friends.length-1 因為 array 是 012 length 是 0123 所以 length 要 -1 才會對樣 array 的 012*
let friend = friends[i]
/*讓 i 的數字填入 friends 裡面 array 會倒過來*/
reversed_friends.push(friend);
}
console.log(reversed_friends);
```

```
function finbiggest(arr){ let biggesNumber = -100000; for(let i = 0 ; i<arr.length;i++){ if(arr[i] > biggesNumber){ biggesNumber = arr[i] }} return biggesNumber; } console.log(finbiggest([1,2,3,4,6,9,5,2,1100,200 ]))
```