JavaScript 的型別主要分兩大類別，分別是原始型別 (Primitive type) 及參考型別 (Reference type)。

基本型別(Primitive) ( string、number、boolean、null、undefiend、blident、Symbol)

1. **String : 字串，表示文字型態**

let stri ="Hello',"Hello"

console.log (stri)=> 'Hello'

* **String 物件內建的屬性 (Properties)**

1. String.length表示字串的長度。

let str = “Lorem ipsum dolor”

str.length //17

* **String 物件內建的方法 (Methods)**

1. Str.charAt(index) :取得特定位置字元，預設值 0 開始。

let str = "Lorem ipsum dolor"

str.charAt(1) // o

1. **Str.charCodeAt(index) :**取得特定位置字元的 Unicode 編碼，如輸入值超出範圍則 NAN。

let str = "Lorem ipsum dolor"

str.charCodeAt(1) // 111

1. **Str.concat() :**將一個或多個字串和並於原字串中，並回傳新字串。

let str = "Lorem ipsum dolor"

str.concat( 'sit amet','consectetur adipisicing elit. ')

// "Lorem ipsum dolor sit amet , consectetur adipisicing elit."

1. **Str.includes() :**取得字串內是否相符內容，並回傳 true 或 false。

let str = "Lorem ipsum dolor"

str.includes('dolor') //true

1. Str.indexOf() :判斷字串中是否包含此字

let str = "Lorem ipsum dolor"

str.indexOf('x') // -1

1. Str.lastIndexOf() 判斷字串中是否包含此字串，但開始起點從右至左
2. let str = "Lorem ipsum dolor"

str.indexOf('r') // 16

1. Str.repeat(count) 重複原字串並回傳新字串count:重複的整數

"Lorem".repeat(2) //"LoremLorem"  
"Lorem".repeat(3) //"LoremLoremLorem"

1. Str.replace() 將原字串取代另一個字串，

let str='Lorem ipsum dolor'

str.replace('dolor', 'aaa') // "Lorem ipsum aaa"

9. Str.slice( ) 取得原字串特定位置值，並回傳新字串

let str='lorem Ipsum dolor'

str.slice(1,5) // "orem"

str.slice(-3) // "lor"

# 10 . Str.substring() 擷取部分文字

const r = "hallo work";

r.substring(1,4) // all

# 11. Str.toLowerCase() 將字串英文字母轉成小寫

let str="LOREM IPSUM DOLOR"  
str.toLowerCase() // "lorem ipsum dolor"

# 12. Str.toUpperCase() 將字串英文字母轉成大寫

let str= "lorem ipsum dolor"  
str.toUpperCase() // "LOREM IPSUM DOLOR"

# 13. Str.toString() 任何非字串的值被強制轉型為字串時

[1,2,3].toString() 會得到 "1,2,3"。

14 . Str.trim() 將字串中空白符號移除

let str = " lorem ipsum dolor "  
str.trim() // "lorem ipsum dolor

1. Number : 表示數字型態，包含整數、浮點數

其中有幾種特別的數字：Infinity、-Infinity、NaN (Not a Number)

+Infinity : 正無限值

-Infinity: 負無限值

NaN: 代表不是數字

例如123 / 0.123

1. Boolean : 布林值，只有兩種值：`True` 或 `False

var a = true;

var b = false;

var c = ( 100 > 10 ); // true

1. Null : 絕對空值，只會有一個值 null`
2. Undefined : 定義一個變數，但沒有設值時，會跑出Undefined。

一張含有 文字, 電子產品, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

var a; // undefined, 尚未給值，未定義

var b = null; // null, 明確代表此變數沒有值

1. Blgint : **BigInt** 可以讓我們任意選擇其精準度，就可以避免一些 **number** 會遇到的問題。它跟 **number** 一樣可以用 **+**, **\***, **-**, **\*\***, 與  **%** 等運算子，不過要注意不可以拿 **BigInt** 跟 **number** 型別的值交互使用，這會出現 **TypeError** 。
2. Symbol: 每一個 Symbol 的值都是獨一無二的，可以保證不會跟其他的屬性名產生衝突

定義屬性鍵名的時候一定要將 Symbol 值放在中括號 [] 裡面，否則屬性的鍵名會當做字串而不是 Symbol 值

例如:

一張含有 文字, 電子產品, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

參考型別 (Reference type)。

原始型別以外的都是物件型別(Object Type)，包括 Function、Array 都是屬於物件型別。

### 物件 Object: 使用 { } 包裹的資料，並包含一對 key 和 value，可以儲存多組

物件中，每個特性都有一組對應的名稱(key)與值(value)，要注意的是**物件中的特性並不是用;的方式區隔，而會以,的隔開每個特性。**

**/*/ name 為特性(key)，而 Bill 為值(value)。***

***var obj = {***

***name: 'Bill',***

***habbit: 'play computer'***

***};***

對於物件特性的取值方式則有兩種: 1.透過 . 的**特性存取，**2.透過[" "] 的**鍵值存取**

Ex: 透過 . 的方式**取值**

**var obj = {**

**name: 'Bill',**

**habbit: 'play computer'**

**};**

**console.log(obj.name);**

Ex2: 透過 [" "] 的方式**取值**

**var obj = {**

**name: 'Bill',**

**habbit: 'play computer'**

**};**

**console.log(obj["habbit"]);**

Ex: 透過 . 的方式**存值**

var obj = {};

obj.name = 'Bill';

console.log(obj); //*{name: 'Bill'}*

Ex2: 透過 [" "] 的方式**存值**

var obj = {};

obj["habbit"] = 'play computer';

console.log(obj); // *{habbit: 'play computer'}*

### }陣列 Array:使用 [ ] 包裹的資料，可以儲存多個資料

let drinks = ["可口可樂", "可爾必思", "珍珠奶茶"]

### 陣列就是一個「有序」、「有索引」、「有值」的一個「集合」。

常見陣列方法

1. push : 在陣列最後方加入新元素

var array = ['小明', '杰倫', '漂亮阿姨', '小美']

array.push('老媽');

console.log(array); // ["小明", "杰倫", "漂亮阿姨", "小美", "老媽"]

1. pop : 移除最後一位元素

var array = ['小明', '杰倫', '漂亮阿姨', '小美']

array.pop();

console.log(array); // ["小明", "杰倫", "漂亮阿姨"]

1. shift() 則可以移除第一個陣列值。

var array = ['小明', '杰倫', '漂亮阿姨', '小美']

array.shift();

console.log(array); // ["杰倫", "漂亮阿姨", "小美"]

1. unshift() 如果需要加入在第一個則可以使用

var array = ['小明', '杰倫', '漂亮阿姨', '小美']

array.unshift('老媽');

console.log(array); // ["老媽", "小明", "杰倫", "漂亮阿姨", "小美"]

1. splice( 索引位置, 數量) 可以移除指定位置、指定數量的陣列值

var array = ['小明', '杰倫', '漂亮阿姨', '小美']

array.splice(1, 1);

console.log(array); // ["小明", "漂亮阿姨", "小美"]

1. reverse() 會反轉陣列的本身。

var array = ['小明', '杰倫', '漂亮阿姨', '小美']

array.reverse();

console.log(array); // ["小美", "漂亮阿姨", "杰倫", "小明"]

1. concat 是用於串接兩個陣列，串接的結果會透過回傳值取得，並不會套用至原有的陣列

var array = ['小明', '杰倫', '漂亮阿姨', '小美']

var newArray = array.concat(['老爸', '老媽']);

console.log(newArray); // ["小明", "杰倫", "漂亮阿姨", "小美", "老爸", "老媽"]

1. include 可以用來檢查陣列中是否包含特定值，結果會透過回傳的方式取得。

var array = ['小明', '杰倫', '漂亮阿姨', '小美']

console.log(array.includes('漂亮阿姨')); // true

console.log(array.includes('阿明')); // false

1. indexOf 則是回傳該值的索引位置。

var array = ['小明', '杰倫', '漂亮阿姨', '小美']

console.log(array.indexOf('漂亮阿姨')); // 2

console.log(array.indexOf('阿明')); // -1

1. join()可以將陣列中的值轉變為字串，並且加入特定字元作為值相間的符號。

var array = ['小明', '杰倫', '漂亮阿姨', '小美']

console.log(array.join(', ')); // 小明, 杰倫, 漂亮阿姨, 小美

Function

當我們透過 typeof 去檢查一個「函式 (function) 」的時候，雖然你會得到 "function" 的結果，讓你以為 function 也是 JavaScript 定義的一種型別，但實際上它仍屬於 Object 的一種。

「函式」指的是將一或多段程式指令包裝起來，可以重複使用，也方便維護。

宣告函式的方法有好幾種，但不管是什麼方式，通常一個函式會包含三個部分：

* 函式的名稱 (也可能沒有名稱)
* 在括號 ( ) 中的部分，稱為「參數 (arguments) 」，參數與參數之間會用逗號 , 隔開
* 在大括號 { } 內的部分，內含需要重複執行的內容，是函式功能的主要區塊。

例如：

一張含有 文字, 螢幕擷取畫面, 字型, 陳列 的圖片

自動產生的描述

定義函式的方式:

常見定義函式的方式有這幾種：

1函式宣告（Function Declaration）

2函式運算式（Function Expressions）

1函式宣告（Function Declaration）

一張含有 文字, 螢幕擷取畫面, 字型, 名片 的圖片

自動產生的描述

2 函式運算式（Function Expressions）

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

資料轉換型別

JavaScript是**弱型別**的程式語言，對型別的要求並沒有像JAVA、C#如此嚴謹，相對地，彈性許多。  
當我們給的值不是JavaScript預期想要的型別，它會想辦法轉成它要的型別。

強制轉型只有三種類型：

* 轉為字串
* 轉為布林值
* 轉為數字

1.轉字串（string）

String(123); // '123'

2.轉數字（number）

1.用 Number()

Number('123'); // 123

Number(' 123123123 '); // 123123123

Number(' 12313213231231hhhhh'); // NaN

Number(' 12313213231231hhhhh1232123132'); // NaN

Number('aaa'); // NaN

Number('12 213'); // NaN

1. 用 parseInt()

parseInt('123'); // 123

parseInt(' 123123123 ');

parseInt(' 12313213231231hhhhh');

parseInt(' 12313213231231hhhhh1232123132'); // 12313213231231

parseInt('aa'); // NaN

parseInt('12 213'); // 12

3 . 轉 Boolean

Boolean 值不管再怎麼轉型，最後都只會有兩種結果，其中有一些值一定會被轉型為 false ，JS 裡面用 falsy value 來描述那些必定會被轉型為 false 的數值。

Boolean('') // false

Boolean(0) // false

Boolean(-0) // false

Boolean(NaN) // false

Boolean(null) // false

Boolean(undefined) // false

Boolean(false) // false

怎麼判斷哪些值會被轉型為 true 呢？ 很簡單：除了 falsy value 以外的值就會被轉為 true ，所以只要搞清楚上述幾種情況即可，以下舉例被轉型為 ture 的情況（ Trusy Value ）:

Boolean({}) // 空物件也是 true !

Boolean(function() {}) // function 也是物件喔！

Boolean([]) // 空陣列也是物件喔!

Boolean(Symbol()) // true

!!Symbol() // true