字串處理

1. length 字串長度

傳回字串長度「計算幾個字」空字串的 length 為 0。JavaScript 使用 UTF-16 編碼用來表示大部分常見的字元，使用兩個代碼單元表示不常用的字元。因此 length 返回值可能與字串中實際的字元數量不相同。獨角碼一個字可能有兩個或四個位元組。(但尚未找到)

```
stringObject.length;
```

```
let myStr = "myString";
stringObject.length; /* 8 */
let myStr = "基本字串";
stringObject.length; /* 4 */
let myStr = "《基本(符號)字串》";
stringObject.length; /* 10 */
let myStr = "基本的な文字列";
stringObject.length; /* 7 */
```

2. substr() 提取字串中的幾個字

取字串中的幾個字 JavaScript 的起始位置是 0。如果省略了 length 那麼返回從 stringObject 的開始位置到結尾的字串。-1 指字串中最後一個字元、-2 指倒數第二個字元，以此類推。

```
stringObject.substr(start, length)
```

```
let myStr = "myString";
myStr.substr(2,3); /* Str */
myStr.substr(-3,3); /* ing */
myStr.substr(2); /* String */
```

3. substring() 提取字串中兩個指定索引號之間的字元

取字串中的幾個字，起始位置是 0 從指定的位置 start 到指定「位置 index」的字串。substring() 不接受負的參數小於 0 則為 0。

```
stringObject.substring(start, index)
```

```
let myStr = "myString";
myStr.substring(2,5); /* Str */
myStr.substring(2,25); /* String (超出 length 只取到字串的結尾) */
```

4. slice() 取得部份字串

slice() 比 substring() 差異是可以使用「負數」為參數。另與 substr() 不同是因為它用兩個字元的「位置」來指定子串，而 substr() 則用字元位置和長度來指定子串。

```
stringObject.slice(start, end)
```

start 要取得部份字串的起始。如果是負數，則該參數規定的是從字串的尾部開始算起的位置。

也就是說 -1 指字串的最後一個字元，-2 指倒數第二個字元，以此類推。

end 接著要取得部份字串的結尾。若未指定此參數，則要提取的子串包括 start 到原字串結尾的字串。

如果該參數是負數，那麼它規定的是從字串的尾部開始算起的位置。

```
let myStr = "myString ABCDEF";
myStr.slice(5); /* ing ABCDEF */
myStr.slice(2,7); /* Strin */
myStr.slice(-6,-3); /* ABC */
myStr.slice(-3); /* DEF */
```

5. charAt() 取字串中的一個字

取字串中的一個字 index 索引指標從 0 算起、如果參數 index 不在 0 與字串長度之間，將返回一個空字串。

```
stringObject.charAt(index)
```

```
let myStr = "myString";
myStr.charAt(0); /* m */
myStr.charAt(2); /* S */
myStr.charAt(10); /* 返回一個空字串 */
```

6. 連接字串

在 **JavaScript** 中，使用加號(**+**)將字串連接；這也是我們做數字相加的方式。示範：

```
let one = "Hello, ";
let two = "how are you?";
let joined = one + two;
joined;
```

7. indexOf() 字串尋找

字串尋找「由左至右尋找」將從頭到尾地檢索字串 **stringObject** 是否包含有 **searchvalue**。開始檢索的位置在字串的 **fromIndex** 處或字串的開頭**(**沒有指定 **fromIndex** 時**)**。如果找到一個 **searchvalue** 則返回 **searchvalue** 的第一次出現的位置。**stringObject** 中的字元位置是從 **0** 開始的。

```
stringObject.indexOf(searchvalue,fromIndex)
```

```
let myStr = "myString String";
myStr.indexOf("Str",1); /* 2 */
myStr.indexOf("Str",5); /* 9 */
myStr.indexOf("Str",10); /* -1 沒有找到 */
```

8. lastIndexOf() 字串反向尋找

```
stringObject.lastIndexOf(searchvalue,fromIndex)
```

9. match() 傳回尋找到的字串

傳回尋「找到的字串」**regexp** 為待尋找的字串，可加 **/g** 代表總體尋找，可加 **/i** 代表不分大小寫。

```
stringObject.match(regexp)
```

```
let myStr = "myString";
myStr.match("Str"); /* Str */
myStr.match(/Str/); /* Str */
myStr.match(/str/); /* null */
myStr.match(/str/ig); /* Str,Str (總體尋找) */
```

10. split() 分隔成字串陣列

用於把字串分隔成「字串陣列」(**separator** 為分隔字串 **length** 為分隔數目)。該陣列是通過在 **separator** 指定的邊界處將字串 **stringObject** 分割成子串創建的。返回的陣列中的字串不包括 **separator** 自身。如果分隔字串 **separator** 為空字串 **("")** 那麼 **stringObject** 中的每個字元之間都會被分割。

```
stringObject.split(separator, length);
```

```
let myStr = "What plan for Weekend?"
myStr.split(""); /* W,h,a,t, ,p,l,a,n, ,f,o,r, ,W,e,e,k,e,n,d,? */
myStr.split("",8); /* W,h,a,t, ,p,l,a (分隔數目 8) */
```

```
myStr.split(" "); /* What,plan,for,Weekend? */
myStr.split(/\s+/); /* What,plan,for,Weekend? */
myStr.split(" ",3); /* What,plan,for (分隔數目 3) */
myStr.split(/\s+/,3); /* What,plan,for (分隔數目 3) */
```

/\s+/ 相等於多次空白字元。

運算子

1. 比較運算子

- 等於：`==`
- 不等於：`!==`
- 大於：`>`
- 小於：`<`
- 大於等於：`>=`
- 小於等於：`<=`

= 跟 == 的區別
=:賦予變數一個值(最常見的賦值運算子)
==: 比較兩個值是否相同(比較運算子)

嚴格等於(===)
先判斷型別，再判斷內容。

```
console.log(1 === '1');
//
console.log('1' === '1');
//
```

2. 邏輯運算子

通常在判斷式裡與布林值一起使用。
包含以下三種符號：

- **&&**：and，連接兩個以上的條件，全部條件成立才會回傳 true
- ||：or，連接兩個以上的條件，只要其中一個條件成立就會回傳 true
- **!**：not，可以把回傳**值**轉成相反的，例如下面的程式碼

```
console.log(2>3);
//
console.log(!(2>3));
//
```

==========

pixel

display block inline

Object

Array

var let const

Flex