Manipulate Strings in JavaScript

< [regular express](https://code.tutsplus.com/tutorials/8-regular-expressions-you-should-know--net-6149) >

# 基本操作

字串在程式語言中，一般表示的是多個字元的組合，可以是數字也可以是字母。

## Creating and Viewing the 結果 of Strings

3種方式可以定義字串

(' '), 雙引號 (" "), 反引號 (` `).

'This string uses single quotes.';

"This string uses double quotes.";

`This string uses backticks.`;

第3種又稱為template literal後面討論

輸出用的函數：

顯示在主空螢幕尚可以利用

console.log():

或者以視窗形式表示 alert():

alert("This is a string in an alert.");

## 指派給變數

儲存或指派的方式可以利用關鍵字 var, const or let。例如

**const** newString = "This is a string assigned to a variable.";

console.log(newString);

結果

This is a string assigned to a variable.

對程式設計而言，利用變數來表示一個字串，可以不用每次都要打入一次。

## 字串連結（String Concatenation）

**Concatenation**

直接利用 「＋」例如

"Sea" + "horse";

結果

Seahorse

如果要有空白，則

"Sea " + "horse";

結果

Sea horse

連結符號＋可以用於變數和常數。例如，

**const** poem = "The Wide Ocean";

**const** author = "Pablo Neruda";

**const** favePoem = "My favorite poem is " + poem + " by " + author ".";

結果

My favorite poem is The Wide Ocean by Pablo Neruda.

## Variables in Strings with Template Literals

這裡我們來看一下模版的使用方式，注意符號 ${} 代表，我們要插入一個變數。

**const** poem = "The Wide Ocean";

**const** author = "Pablo Neruda";

**const** favePoem = `My favorite poem is ${poem} by ${author}.`;

結果

My favorite poem is The Wide Ocean by Pablo Neruda.

## String Literals and String Values

在程式碼撰寫中，用雙引號刮起來的部分，為**string literal**。但實際利用console.log(),alert()輸出的時候，卻看不到雙引號，前者就是**string literal**後者為**string value**。

"Beyond the Sea"; //literal

結果

Beyond the Sea //value

|  |
| --- |
| 一些怪名詞的翻譯 token: 代幣，投了錢，然後通關。在compiler中，讀入使用者程式碼的過程，一個字元一個字元的讀入，當讀入一個token，就到達一個狀態，這個情況也類似投幣然後通關。  literal  A literal is a token in a program text that denotes a value. There are string literals like "123", character literals like 'a' and numeric literals like 7.  int contestants[] = {1, 2, 3};  In the program fragment above there are three literals 1 2 and 3 and no others. In particular, neither contestants nor {1, 2, 3} are literals |

## Escaping Quotes and Apostrophes in Strings

特殊情況，例如字串中帶有單引號，或雙引號，會讓上述的用法產生問題，例如，

**const** brokenString = 'I'm a broken string';

console.log(brokenString);

結果

unknown: Unexpected token (1:24)

解決辦法，為外圍部分用雙引號。類似的解決辦法,大概有下面幾種:

* 雙引號，和單引號交叉混用。
* 逃逸字元(Escape characters)
* Template literals

後面探索一下上面三個方法:

##### 雙引號，和單引號交叉混用。

雙引號，和單引號交叉混用。例如，

"We're safely using an apostrophe in double quotes."

或者

'Then he said, "Hello, World!"';

##### Using the Escape Character (\)

利用 逃逸字元(\)，例如

'We\'re safely using an apostrophe in single quotes.'

或者

"Then he said, \"Hello, World!\"";

##### Using Template Literals

`We're safely using apostrophes and "quotes" in a template literal.`;

## Long Strings and Newlines

利用逃逸字元表示的特殊符號，例如

 \n 或者or \r 都是換行。

**const** threeLines = "This is a string\nthat spans across\nthree lines.";

結果

This is a string

that spans across

three lines.

程式撰寫的時候，可以利用如下的方式，讓debug容一些：

**const** threeLines = "This is a string\n" +

"that spans across\n" +

"three lines.";

要表示多行，也可以用template literal ，例如

**const** threeLines = `This is a string

that spans across

three lines.`;

結果

This is a string

that spans across

three lines.

It's important to be aware of all the ways of creating newlines and strings that span across multiple lines, as different code bases may be using various standards.

# 字串函數

A **string** 由多個characters 組成，像是letters, numbers, or symbols。繼續探討字串相關函數之前，先區分string primitive 和 string object。.

## String Primitives and String Objects

Javascript 對字串的處理，分成兩種type 一種是string,一種是物件。區別起見，前者 **string primitive**，後者 **String object**。

// Initializing a new string primitive

**const** stringPrimitive = "A new string.";

// Initializing a new String object

**const** stringObject = **new** **String**("A new string.");

利用typeof 檢視到底是哪種型態：第一個，只是指派了字串給一個變數。

**typeof** stringPrimitive;

結果

string

第二個則建立了物件，然後指派給變數。

**typeof** stringObject;

結果

object

事實上，當我們在字串後面打一點，的時候，JavaScript 會自動幫我們轉成字串物件。例如

"a pen".toUpperCase()

## How Strings are Indexed

每個字串中的字元對應一個從0開始的索引號碼。例如，

| **H** | **o** | **w** |  | **a** | **r** | **e** |  | **y** | **o** | **u** | **?** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

利用號碼，可以操作子字串。

## 存取字串中的字元 Characters

"How are you?";

由於字串也是以陣列的方式存於記憶體，因此可以利用中刮號，找出索引所對應的字元。

"How are you?"[5];

結果

r

也可以利用 charAt()

"How are you?".charAt(5);

結果

r

或 indexOf() 傳回字元所對應的索引。

"How are you?".indexOf("o");

結果

1

雖然有兩個o但是只有傳回第1個O所在的位置。

找最後一個 lastIndexOf()

"How are you?".lastIndexOf("o");

結果

9

也可以找多個字元

"How are you?".indexOf("are");

結果

4

分割： slice() 兩個參數，開始和結束。

"How are you?".slice(8, 11);

結果

you

8到11有4個，但是只傳回you 而不是「you?」

如果沒有第二個參數，則之後的字串會被傳回。

"How are you?".slice(8);

結果

you?

## charCodeAt與fromCharCode

charCodeAt(index) 方法可返回指定位置的字元的 Unicode 編碼( 0 - 65535 之間的整數)。

如果index是超出給定字串的範圍（例如負數，或大於等於字串的長度），則charCodeAt()返回NaN。

String.fromCharCode(num1,num2,...,numX)

fromCharCode() 可接受一個或多個指定的 Unicode 值，然後返回一個字串。

該方法是String的靜態方法，字串中的每個字元都由單獨的數位 Unicode 編碼指定。

**>>** x**=**"你".charCodeAt(0)

20320

**>>** String.fromCharCode(x)

"你"

## 字串長度

利用屬性length 可以得到字串長度。

"How are you?".length;

結果

12

如果長度為12，則最大索引為11，最小為0.

## 大小寫轉換

 toUpperCase() :轉成大寫

toLowerCase() :轉成小寫

"How are you?".toUpperCase();

結果

HOW ARE YOU?

"How are you?".toLowerCase();

結果

how are you?

注意：這兩個函數並不會更改原來的字串內容。

## Splitting Strings

split() 分解字串

**const** originalString = "How are you?";

// Split string by whitespace character

**const** splitString = originalString.split(" ");

console.log(splitString);

結果

[ 'How', 'are', 'you?' ]

然後利用陣列的方式，存取每個被分解的字串。例如

splitString[1];

結果

are

如果不給定分隔字元（不給參數），則內定使用comma 。

## Trimming Whitespace

The JavaScript trim() method removes white space from both ends of a string, but not anywhere in between. Whitespace can be tabs or spaces.

**const** tooMuchWhitespace = " How are you? ";

**const** trimmed = tooMuchWhitespace.trim();

console.log(trimmed);

結果

How are you?

The trim() method is a simple way to perform the common task of removing excess whitespace.

## 搜尋和代換

在字串中代換某些字，可以利用replace() 方法：第1個參數是我們要找的字串，第2個參數是我們要代換的字串：

**const** originalString = "How are you?"

// Replace the first instance of "How" with "Where"

**const** newString = originalString.replace("How", "Where");

console.log(newString);

結果

Where are you?

在replace()中，可以利用功能較為強大的 [Regular Expressions](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions) 來取代字串。例如要替換多個字串的時候，可以利用，  g (global) 旗標代表全域搜尋，而不是只有代換一次, 而i旗標代表忽略大小寫 (case insensitive)。

【例】

**const** originalString **=** "Javascript is a programming language. I'm learning javascript."

// Search string for "javascript" and replace with "JavaScript"

**const** newString **=** originalString.replace(**/**javascript**/**gi, "JavaScript");

console.log(newString);

結果

JavaScript is a programming language. I'm learning JavaScript.

【例】"http\\u00253A\\u00252F\\u00252Fexample.com" 是 unicode, escaped 字串，要轉換成看unescaped 字串：

**var** x **=** "http\\u00253A\\u00252F\\u00252Fexample.com";

**var** r **=** **/**\\u([\d\w]{4})**/**gi;

x **=** x.replace(r, **function** (match, grp) {

**return** String.fromCharCode(parseInt(grp, 16)); } );

console.log(x); // http%3A%2F%2Fexample.com

x **=** unescape(x);

console.log(x); // http://example.com

範例中，利用正規表達式找 \u0025，unicode 是0025，因此用刮號表示分組號碼，表示只要利用到這部分。

gi 表示全部，i表示不分大小寫。

##### 參考

* < [Regexr](http://regexr.com/) 網站>
* <tutorial "[How To Work with Strings in JavaScript](https://www.digitalocean.com/community/tutorials/how-to-work-with-strings-in-javascript).">

# 應用

## 字串、整數 互轉

hexString = yourNumber.toString(16);

配對

yourNumber = parseInt(hexString, 16);

## input type="textarea"

<[demo code](..\cwww\mycalculator.html)>：改進計算機的功能

<!-- 遊標的位置

遊標之前的字串：scr.value.substr(0,scr.selectionStart)

遊標之前的字串：scr.value.substr(scr.selectionStart,scr.value.length)

例如

12I345

之前：12

之後：345

-->

# 字串搜尋

var str=" I like going swimming in the sea"

var pts="sea"

var ptr=/sea/

string 物件的搜尋,可以用的函數有:

search()

match()

indexof()

includes()

regep 的搜尋,函數有

test()

exec()

但是,其中 incude()和indexof() 只能用字串。

其他函數雖然可以交替使用,但是如果是利用字串搜尋,就不能有//,例如

str.match("/sea/") 不行

str.match(/sea/) 可以 , 自行測試 /sea/I, /sea/g