



String & Regex

Module: Advanced Programming with JavaScript



Kiểm tra bài trước



Mục tiêu

- Giải thích về biến chuỗi và hằng chuỗi
- Trình bày cách khai báo biến chuỗi
- Trình bày các thuộc tính và phương thức thao tác với chuỗi
- Trình bày được khái niệm Regular Expression
- Giải thích các quy tắc làm việc với Regular Expression

Thảo luận

Chuỗi

Chuỗi



- Chuỗi là cách nói ngắn gọn của "chuỗi ký tự"
- Một hằng chuỗi có thể bao gồm bất cứ ký tự nào nằm trong cặp dấu nháy đơn hoặc nháy kép
- Ví dụ:

"Hello world"

"B"

"This is 'another string' "

'Bò kêu "moo" '

- Các hằng chuỗi được sử dụng để gán cho các biến chuỗi

Khai báo biến Chuỗi



- Cú pháp

- Khai báo như khai báo biến

```
let str = "This is a string";
```

- Sử dụng đối tượng chuỗi

```
let str = new String("This is a string");
```

- Sử dụng ký tự \ để thể hiện các ký tự nháy đơn hoặc nháy kép, ví dụ:

```
let message = 'CodeGym \'2017\' !';
```

```
let domain = "CodeGym - \"2017\" !";
```

Các ký tự đặc biệt



Tổ hợp ký tự	Ký tự hiển thị
\'	Dấu nháy đơn
\"	Dấu nháy kép
\\	Dấu sổ chéo ngược
\n	Xuống dòng
\t	Dấu tab

Nối chuỗi



- Sử dụng dấu + để ghép hai chuỗi hoặc biến chuỗi
- Ví dụ:

```
let message = "Welcome to" + "CodeGym!";
```

Hoặc

```
let message1 = "Welcome to";
```

```
let message2 = "CodeGym";
```

```
let message = message1 + message2;
```




Thuộc tính length

- Thuộc tính **length** cho biết độ dài của chuỗi

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
var sln = txt.length;
```

26

Chỉ số của ký tự trong chuỗi



- Chỉ số của ký tự trong chuỗi bắt đầu từ 0

Chuỗi	C	o	d	e	G	y	m
Chỉ số	0	1	2	3	4	5	6

Phương thức thao tác chuỗi



Phương thức	Mô tả
toUpperCase()	Chuyển toàn bộ ký tự của chuỗi về dạng chữ in hoa
toLowerCase()	Chuyển toàn bộ ký tự của chuỗi về dạng chữ thường
concat()	Nối các chuỗi lại với nhau
indexOf()	Trả về chỉ số của chuỗi trùng khớp được tìm thấy đầu tiên
lastIndexOf()	Trả về chỉ số của chuỗi trùng khớp được tìm thấy sau cùng
charAt()	Trích xuất <u>một ký tự</u> trong chuỗi
substring()	Trích xuất <u>một chuỗi con</u> trong chuỗi
substr()	Trích xuất <u>một chuỗi con</u> trong chuỗi
replace()	Thay thế một nội dung nào đó trong chuỗi bằng nội dung mới

indexOf()



- indexOf() trả về chỉ số (vị trí) của một chuỗi nằm ở trong một chuỗi khác. Trả về -1 nếu không tìm thấy.
- Nếu chuỗi con xuất hiện nhiều lần thì sẽ trả về vị trí lần xuất hiện đầu tiên
- Ví dụ:

Vị trí lần xuất hiện đầu tiên

```
var str = "Please locate where 'locate' occurs!";  
var pos = str.indexOf("locate");
```

7

lastIndexOf()



- lastIndexOf() trả về vị chỉ số (vị trí) của một chuỗi nằm trong một chuỗi khác, tính từ lần xuất hiện cuối cùng
- Trả về -1 nếu không tìm thấy
- Ví dụ:

Vị trí lần xuất hiện cuối cùng

```
var str = "Please locate where 'locate' occurs!";  
var pos = str.lastIndexOf("locate");
```

21

slice()



- slice() tách một phần của chuỗi tính từ một vị trí bắt đầu cho đến vị trí kết thúc
- Chuỗi con bao gồm ký tự tại vị trí bắt đầu, nhưng không bao gồm ký tự tại vị trí kết thúc
- Ví dụ:

```
var str = "Apple, Banana, Kiwi";  
var res = str.slice(7, 13);
```

Vị trí 7 Vị trí 13

Banana

substr()



- substr() tách một phần của chuỗi tính từ một vị trí bắt đầu và một độ dài mong muốn
- Ví dụ:

```
var str = "Apple, Banana, Kiwi";  
var res = str.substr(7, 6);
```

Vị trí 7

6 ký tự

Banana

replace()



- replace() thay thế một phần của chuỗi bằng một chuỗi khác
- replace() không thay đổi chuỗi gốc mà trả về một chuỗi mới
- Nếu chuỗi con xuất hiện nhiều lần thì replace() chỉ thay thế lần xuất hiện đầu tiên
- Ví dụ:

```
var str = "Hello World";  
  
var txt = str.replace("World", "CodeGym");
```

↑
Hello CodeGym



toUpperCase()

- toUpperCase() chuyển một chuỗi thành chữ viết hoa
- Ví dụ:

```
var text1 = "Hello World!";  
var text2 = text1.toUpperCase();
```

HELLO WORLD!



toLowerCase()

- toLowerCase() chuyển một chuỗi thành chữ viết thường
- Ví dụ:

```
var text1 = "Hello World!";  
var text2 = text1.toLowerCase();
```

hello world!

trim()

- trim() loại bỏ các khoảng trắng ở hai đầu của chuỗi
- Ví dụ:

```
var str = "      Hello World!      ";  
var text = str.trim();
```

↑
"Hello World!"

charAt()



- charAt() trả về một ký tự tại vị trí một vị trí
- Ví dụ:

Vị trí 0

↓

```
var str = "HELLO WORLD";  
var c = str.charAt(0)
```

↑

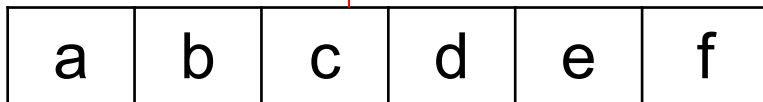
H

split()



- Split chuyển một chuỗi thành mảng dựa trên một dấu hiệu phân tách
- Ví dụ:

```
var str = "a,b,c,d,e,f";  
var arr = str.split(",");
```



a	b	c	d	e	f
---	---	---	---	---	---

Dấu hiệu phân tách





Demo

Các phương thức thao tác với chuỗi



Thảo luận

Regular Expression

Regular Expression



- Biểu thức chính quy là một đối tượng mô tả một mẫu các ký tự
- Được sử dụng trong các thao tác tìm kiếm và thay thế (search-and-replace)
- Ví dụ:
 - Tìm kiếm những khách hàng có họ hoặc tên chứa một chuỗi nhất định
 - Kiểm tra tính hợp lệ của một email
 - Thay thế một chuỗi trong một tài liệu văn bản

Khai báo Regular Expression



- Cú pháp:

/pattern/modifiers;

Trong đó:

- pattern là một mẫu chuỗi
- modifiers là các ký hiệu quy định hành vi tìm kiếm (phân biệt hoa/thường, tìm kiếm toàn cục)

- Ví dụ:

```
var patt = /CodeGym/i
```



Demo

Sử dụng regular expression



Thảo luận

Sử dụng các phương thức của String

search()



- search() tìm kiếm một chuỗi và trả về chỉ số của chuỗi đó
- Ví dụ:

```
var str = "Visit CodeGym";  
var n = str.search(/codegym/i);
```

Vị trí số 6



6



replace()



- replace() thay thế một phần của chuỗi bằng một chuỗi khác
- Ví dụ:

```
var str = "Visit Microsoft!";  
var res = str.replace(/microsoft/i, "CodeGym");
```

Visit CodeGym



Sử dụng đối tượng RegExp

test()



- test() tìm kiếm một mẫu chuỗi, trả về giá trị **true** nếu tìm thấy, trả về giá trị **false** nếu không tìm thấy
- Ví dụ:

```
var patt = /@/;  
var result = patt.test( 'info@codegym.vn' );
```

↑
true

```
var patt = /@/;  
var result = patt.test( 'infocodegym.vn' );
```

↑
false

exec()



- exec() tìm kiếm một mẫu chuỗi và trả về chuỗi tìm được
- Ví dụ:

```
var patt = /o+/;  
var result = patt.exec( 'Helloooooo CodeGym' );
```

↑
oooooo

Các modifier



- Modifier thay đổi hành vi của việc tìm kiếm và thay thế
- Các modifier bao gồm:
 - i: Không phân biệt chữ hoa và chữ thường
 - g: Tìm kiếm toàn cục (thay vì chỉ tìm kiếm lần xuất hiện đầu tiên)
 - m: Tìm kiếm trên nhiều dòng
- Ví dụ:

```
var str = "Is this all there is?";  
var patt = /is/g;  
var result = str.match(patt);
```

↑
is,is

Dấu ngoặc



- Các dấu ngoặc quy định các khoảng ký tự
- Ví dụ:

	Biểu thức	Ý nghĩa
→	[abc]	Tìm một trong số các ký tự a hoặc b hoặc c
→	[^abc]	Tìm các ký tự không phải là a hoặc b hoặc c
→	[0-9]	Tìm một giá trị nằm trong khoảng từ 0 đến 9
→	[^0-9]	Tìm một giá trị không nằm trong khoảng từ 0 đến 9
→	(red green)	Tìm kiếm chuỗi 'red' hoặc 'green'

Dấu ngoặc: Ví dụ



- Tìm kiếm các ký tự nằm trong khoảng từ 'a' đến 'h'

```
var str = "Is this all there is?";  
var patt1 = /[a-h]/g;  
var result = str.match(patt1);
```

↑
h,a,h,e,e

Các ký tự mô tả

- Các ký tự mô tả (metacharacter) là các ký tự đặc biệt, có ý nghĩa riêng được quy định trước

Ký tự mô tả	Ý nghĩa
\d	Tìm một số
\D	Tìm một ký tự không phải là số
\s	Tìm một ký tự khoảng trắng
\S	Tìm một ký tự không phải là khoảng trắng
\b	Tìm ở vị trí bắt đầu hoặc kết thúc chuỗi
\B	Tìm ở vị trí không phải ở bắt đầu hoặc kết thúc chuỗi
\uxxxx	Tìm một ký tự theo mã Unicode
\n	Tìm ký tự xuống dòng
\t	Tìm ký tự tab



Các ký tự mô tả: Ví dụ

- Tìm kiếm ký tự không phải là số

```
var phone = "098787878a";  
var patt = /\D/;  
var result = patt.test(phone);
```

↑
true

- Có thể sử dụng: `[^0-9]`

Quantifier



- Quantifier quy định hình thức thực hiện việc tìm kiếm dựa trên số lượng các ký tự
- Các quantifier:

Quantifier	Ý nghĩa
n^+	Tìm một chuỗi có chứa ít nhất 1 ký tự n
n^*	Tìm một chuỗi có chứa ít nhất 0 ký tự n
$n?$	Tìm một chuỗi có chứa 0 hoặc 1 ký tự n
$n\{X\}$	Tìm một chuỗi chứa chính xác X lần chữ số n liên tiếp nhau
$n\{X, Y\}$	Tìm một chuỗi chứa từ X đến Y lần chữ số n liên tiếp nhau
n	Tìm một chuỗi bắt đầu bằng ký tự n
$n\$$	Tìm một chuỗi kết thúc bằng ký tự n

Quantifier: Ví dụ



- Kiểm tra tính hợp lệ của tên: Chỉ bao gồm chữ viết hoa, chữ viết thường và ký tự khoảng trắng

```
var name = 'John @ Dewey';  
var patt = /^[a-zA-Z\s]+$/;  
  
var result = patt.test(name);  
      ↑  
false
```



Demo

Sử dụng các cách khác nhau để tạo regular expression

Tóm tắt bài học



- Chuỗi
- Regular Expression

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: *Thuật toán cơ bản*