



BIỂU THỨC CHÍNH QUY REGEXP



Biểu thức chính quy - Regular expression

- **Biểu thức chính quy** là một nhóm các ký tự, ký hiệu được sử dụng để tìm kiếm văn bản (text).
- Một biểu thức chính quy là một mẫu tương đồng quy luật với một chuỗi từ trái qua phải.
- Trong lập trình nó được dùng với các hàm xử lý chuỗi, xử lý văn bản với các tác vụ cụ thể như: *tìm và thay thế chuỗi, kiểm tra tính hợp lệ của dữ liệu, trích xuất chuỗi* con từ một chuỗi cho trước

Biểu thức chính quy - Regular expression

- Ví dụ: ứng dụng yêu cầu người dùng phải tuân thủ quy tắc đặt tên:
 - **Tên** gồm các ký tự thường, các số, gạch dưới, gạch nối.
 - **Tên** phải có độ dài từ 3 đến 15 ký tự
- Biểu thức chính quy biểu diễn quy tắc trên có dạng:

`/^[a-z0-9_-]{3,15}$/`

Biểu thức chính quy - Regular expression

- Ý nghĩa ký tự trong biểu thức chính quy:
 - **^** Ký hiệu cho biết bắt đầu một dòng
 - **[a-z0-9_-]** Cho phép tên chứa ký tự a-z, số từ 0 - 9, ký tự - , ký tự _
 - **{3,15}** Tên dài 3 đến 15 ký tự
 - **\$** Điểm kết thúc dòng
- Với biểu thức chính quy trên thì các tên: congngheweb, congnghe_web, congngheweb_123 được chấp nhận

Biểu thức RegExp cơ bản

- **Biểu thức RegExp cơ bản**: là một mẫu các ký tự dùng để tìm kiếm trong text (chuỗi).
- Biểu thức chính quy Regex phân biệt chữ hoa chữ thường

Các ký tự biểu diễn - Meta

Ký tự Meta	Mô tả
.	Biểu diễn bất kỳ ký tự nào ngoài trừ ký tự xuống dòng
[]	Tập hợp ký tự. Phù hợp nếu có bất kỳ ký tự nào trong dấu []
[^]	Tập hợp ký tự phủ định. Phù hợp nếu không có ký tự nào trong []
*	Lặp lại 0 đến nhiều lần.
+	Lặp lại 1 hoặc nhiều lần
?	Tùy chọn có hay không cho mẫu phía trước

Các ký tự biểu diễn - Meta

Ký tự Meta	Mô tả
{n,m}	Độ dài tối thiểu là n tối đa là m
(xyz)	Biểu diễn một nhóm.
 	Biểu diễn thay thế, phép toán or
\	Biểu diễn ký tự đặc biệt [] () { } . * + ? ^ \$ \
^	Điểm bắt đầu của dòng.
\$	Điểm kết thúc của dòng

Ký hiệu tắt cho tập hợp

Viết tắt	Diễn tả
.	Bất kỳ ký tự nào ngoại trừ xuống dòng
\w	Chữ, số, và _, tương đương với: [a-zA-Z0-9_]
\W	Ngoài bảng chữ cái, tương đương với: [^\w]
\d	Các số: [0-9]
\D	Không phải số: [^\d]
\s	Là ký tự trắng, tương đương với: [\t\n\f\r\p{Z}]
\S	Không phải ký tự trắng: [^\s]

Biểu thức **?= lookahead**

- **Lookahead** **?=** để lọc kết quả.
 - Ký hiệu **?=**. Phần đầu của biểu thức phải được tiếp nối bởi biểu thức **lookahead**.
- Ví dụ
 - **(T|t)he(?=\sfat)** thì **lookahead** là **(?=\sfat)** - nghĩa là **T** hoặc **t** theo sau là **he** vậy tìm được 2 kết quả.
 - Nhưng do có biểu thức lookahead, điều này thì kết quả phù hợp là chỉ lấy khi theo sau nó là chuỗi fat
 - **(T|t)he => The fat cat sat on the mat.**
 - **(T|t)he(?=\sfat) => The fat cat sat on the mat.**

Biểu thức ?! phủ định lookahead

- **Ký hiệu: ?!**, lấy kết quả đi sau nó không có chuỗi lookahead
- Ví dụ:
 - (T|t)he(?!sfat) => The fat cat sat on the mat.

Biểu thức (?<=...) và (?<!...)

- **Biểu thức (?<=...) Lookbehind:** lấy các phù hợp mà đi trước là một mẫu cũ thể.
 - Ví dụ: (?<=(T|t)he\s)(fat|mat) lấy tất cả các từ fat hoặc mat sau các từ The hoặc the
 - (?<=(T|t)he\s)(fat|mat) => The fat cat sat on the mat.
- **Biểu thức (?<!...) phủ định Lookbehind:** lấy các phù hợp mà đi trước không có một mẫu lookbehind chỉ ra.
 - Ví dụ: (?<!(T|t)he\s)(cat) => The cat sat on cat.

Các phương thức của RegExp

- **Phương thức RegExp** test () và execute () và với các phương thức chuỗi match (), Replace (), search () và split ().

Cú pháp để tạo một chuỗi RegEx

- Cú pháp: `/pattern/modifiers`
- Trong đó
 - **Pattern:** chuỗi Regular Expression
 - **Modifiers:** thông số cấu hình cho chuỗi pattern, gồm các giá trị sau:
 - i : so khớp không quan tâm đến chữ hoa chữ thường
 - g : so khớp toàn bộ chuỗi cần tìm
 - m : so khớp luôn cả các dữ liệu xuống dòng (multiline)

Các phương thức của RegExp

Phương thức	Mô tả
<u>exec()</u>	Thực hiện tìm kiếm so sánh trong một chuỗi. trả về một mảng thông tin hoặc null khi không khớp.
<u>test()</u>	Kiểm tra so sánh trong một chuỗi. trả về true hoặc false.
<u>match()</u>	Trả về một mảng chứa tất cả các kết quả phù hợp, bao gồm cả nhóm thu thập hoặc null nếu không tìm thấy kết quả phù hợp nào.
<u>matchAll()</u>	Trả về một iterator có chứa tất cả các kết quả phù hợp, bao gồm cả các nhóm thu thập.

Các phương thức của RegExp

Phương thức	Mô tả
<u>search()</u>	Kiểm tra so sánh trong một chuỗi. trả về chỉ mục của kết quả phù hợp hoặc -1 nếu tìm kiếm không thành công.
<u>replace()</u>	Thực hiện tìm kiếm một kết quả phù hợp trong một chuỗi và thay thế chuỗi con đã so khớp bằng một chuỗi con thay thế
<u>replaceAll()</u>	Thực hiện tìm kiếm cho tất cả các kết quả phù hợp trong một chuỗi và thay thế các chuỗi con phù hợp bằng một chuỗi con thay thế.
<u>split()</u>	Sử dụng một biểu thức chính quy hoặc một chuỗi cố định để ngắt một chuỗi thành một mảng các chuỗi con.

Các phương thức của RegExp

- Ví dụ:

- Phương thức **exec** được dùng để tìm chuỗi phù hợp theo mẫu so khớp.
- `var myRe = /d(b+)d/g;`
- `var myArray = myRe.exec("cdbbdsbz");`

Các phương thức của RegExp

- **test():** kiểm tra một biểu thức regular expression

- Cú pháp: `/pattern/.test(string);`

- Trong đó

- **Pattern:** biểu thức chính quy

- **string:** chuỗi cần tìm.

- Ví dụ:

```
let result = /Javascript/.test("Hello Javascript");  
alert(result); // True
```

Các phương thức của RegExp

- Ví dụ: kiểm tra một chuỗi có bắt đầu bằng chữ **tech** hay không?

```
let result = /tech/.test("Website tech.net là blog  
công nghệ");  
console.log(result); // True
```

Các phương thức của RegExp

- **Kiểm tra ký tự bắt đầu và kết thúc chuỗi RegExp.**

- **Cú pháp**

`/^pattern$/`

- **Trong đó:** trong đó ký tự ^ là khai báo bắt đầu chuỗi, còn \$ là khai báo kết thúc chuỗi.

- **Ví dụ:**

```
let result = /^[A-Z][a-z]/.test("Technology");  
console.log(result); // true
```

Các phương thức của RegExp

- **Match()** lấy kết quả toàn bộ chuỗi regexp, Nếu test() sẽ trả về true nếu so khớp và false nếu không so khớp, thì **match()** sẽ trả về kết quả dựa vào cấu trúc của chuỗi pattern.
- Cú pháp: `string.match(pattern);`
- Trong đó
 - **string** là chuỗi cha, **pattern** là chuỗi **RegExp**.
 - Kết quả trả về một mảng các giá trị so khớp với chuỗi pattern.

Các phương thức của RegExp

- **Match()**

- Ví dụ: Lấy tất cả các chữ số trong chuỗi "số điện thoại 0979306603".

```
var string = "số điện thoại 0979306603";  
var result = string.match(/[0-9]+/img);  
console.log(result); // ["0979306603"]  
  
// Thử đổi chuỗi string sang giá trị khác  
string = "tôi sinh năm 90, vợ tôi sinh năm 92";  
result = string.match(/[0-9]+/img);  
console.log(result); // ["90", "92"]
```

Các phương thức của RegExp

- **exec()** lấy kết quả của một quy tắc trong chuỗi pattern
- Cú pháp `/pattern/.exec(string)`
- Ví dụ: Viết biểu thức chính quy lấu đường dẫn URL bên trong thẻ img dưới đây.

```

```

```
let string = '';  
1  
2  
3var result = //img.exec(string);  
console.log(result);
```

```
//img
```

Bây giờ ta sẽ thử đặt nó vào hàm **exec()** xem kết quả như thế nào nhé.

Kết quả là một mảng gồm hai phần tử: