**Biểu thức chính quy RegEx trong Javascript**

Trong bài này chúng ta sẽ tìm hiểu về biểu thức chính quy trong javascript, hay còn gọi là RegEx trong javascript. Đây là thư viện được dùng để xử lý chuỗi nâng cao nên rất khó học.

Như ta biết, **regular expression** hoạt động theo quy tắc so khớp, va lấy kết quả của sự so khớp đó thông qua những quy tắc mà thư viện RegEx đưa ra. Dựa vào những quy tắc này, lập trình viên sẽ viết thành những biểu thức và áp dụng vào bài toán của mình.

**Để tiện cho việc giải thích thì mình xin có một vài lưu ý như sau**:

* Biểu thức chính quy viết trong tiếng Anh là regular expression
* Regular expression là một thuật ngữ, và chữ viết tắt của nó là RegEx.
* => Vì vậy, khi nhắc đến ba cụm từ này thì bạn phải hiểu nó là một nhé.

**Mục lục**

* [1. RegEx trong Javascript là gì?](https://freetuts.net/regex-trong-javascript-405.html#goto-h2-0)
* [2. Hàm test() trong javascript RegEx](https://freetuts.net/regex-trong-javascript-405.html#goto-h2-1)
* [3. Ký tự bắt đầu và kết thúc chuỗi RegEx trong javascript](https://freetuts.net/regex-trong-javascript-405.html#goto-h2-2)
  + [Ký tự bắt đầu chuỗi pattern](https://freetuts.net/regex-trong-javascript-405.html#goto-h3-0)
  + [Ký tự kết thúc chuỗi pattern](https://freetuts.net/regex-trong-javascript-405.html#goto-h3-1)
* [4. Bảng các biểu thức RegEx trong js căn bản](https://freetuts.net/regex-trong-javascript-405.html#goto-h2-3)
* [5. Bảng các quy tắc RegEx trong javascript khác](https://freetuts.net/regex-trong-javascript-405.html#goto-h2-4)
  + [Modifiers](https://freetuts.net/regex-trong-javascript-405.html#goto-h3-2)
  + [Brackets](https://freetuts.net/regex-trong-javascript-405.html#goto-h3-3)
  + [Metacharacters](https://freetuts.net/regex-trong-javascript-405.html#goto-h3-4)
  + [Quantifiers](https://freetuts.net/regex-trong-javascript-405.html#goto-h3-5)
* [6. Dùng hàm string.match lấy kết quả toàn bộ chuỗi regex](https://freetuts.net/regex-trong-javascript-405.html#goto-h2-5)
* [7. Dùng hàm regex.exec lấy kết quả từng group chuỗi RegEx](https://freetuts.net/regex-trong-javascript-405.html#goto-h2-6)
* [8. Lời kết](https://freetuts.net/regex-trong-javascript-405.html#goto-h2-7)

**1. RegEx trong Javascript là gì?**

RegEx (*regular expression*) là tập hợp những quy tắc, dựa vào những quy tắc này ta sẽ viết ra những biểu thức so khớp giữa các chuỗi với nhau. Đây là một thư viện nâng cao được tích hợp sẵn trong đối tượng RegExp của Javascript.

*Bài viết này được đăng tại [free tuts .net]*

Không chỉ ở javascript, thư viện RegEx còn được tích hợp ở hầu hết các ngôn ngữ lập trình hiện nay. Vì vậy, khi bạn đọc xong bài này và hiểu hết bài rồi, thì qua những ngôn ngữ khác sẽ rất dễ học.

Mình sẽ lấy một ví dụ cực kì đơn giản thế này. Để so sánh hai chuỗi có bằng nhau hay không thì ta sử dụng toán tử bằng =. Để kiểm tra chuỗi con có xuất hiện trong chuỗi cha hay không thì ta có thể sử dụng hàm **substring**.

Nhưng để tìm một chuỗi bắt đầu bằng chữ c và tiếp theo là 5 chữ cái in thường thì làm thế nào? Gặp trường hợp này thì hãy nghĩ đến ngay biểu thức chính quy regular expression nhé.

Trong Javascript thì regular expression là một chuỗi, nhưng nó không được bao quanh bởi cặp dấu nháy đơn ' hoặc nháy kép " mà nó được bao quanh bởi cặp dấu /.

**Cú pháp để tạo một chuỗi RegEx như sau**:

|  |  |
| --- | --- |
| 1 | /pattern/modifiers |

**Trong đó**:

* pattern là chuỗi Regular Expression
* modifiers là thông số cấu hình cho chuỗi pattern, và nó có các giá trị sau:
  + i : so khớp không quan tâm đến chữ hoa chữ thường
  + g : so khợp toàn bộ chuỗi cần tìm
  + m : so khớp luôn cả các dữ liệu xuống dòng (multiline)

**Ví dụ**: Pattern kiểm tra chuỗi có tồn tại chữ "*freetuts*" không, không phân biệt chữ hoa chữ thường và tìm toàn bộ tài liệu.

|  |  |
| --- | --- |
| 1 | var pattern = /freetuts/igm; |

Trong ví dụ này thì:

* Pattern là freetuts
* Modifiers là igm, tức là so khớp toàn bộ kể cả xuống dòng và không quan tâm đến chữ in hoa hay in thường

**2. Hàm test() trong javascript RegEx**

Để kiểm thử một biểu thức regular expression thì chúng ta sẽ sử dụng hàm test. Cú pháp của nó như sau:

|  |  |
| --- | --- |
| 1 | /pattern/.test(string); |

Trong đó pattern chính là biểu thức chính quy, string chính là chuỗi cần tìm.

Giá trị của hàm này sẽ trả về true nếu tìm thấy, và false nếu không tìm thấy.

**Ví dụ**: Kiểm tra chuỗi xem trong chuỗi "*chào mừng bạn đến với freetuts*" có xuất hiện chữ "*freetuts*" không thì ta viết như sau.

|  |  |
| --- | --- |
| 1  2 | let result = /freetuts/.test("chào mừng bạn đến với freetuts");  alert(result); // True |

Kết quả trả về là true vì trong chuỗi trên có xuất hiện chữ *freetuts*.

**3. Ký tự bắt đầu và kết thúc chuỗi RegEx trong javascript**

Cấu trúc của một chuỗi luôn có ký tự bắt đầu và kí tự kết thúc. Và trong RegEx cũng có một số ký tự giúp quy ước nơi bắt đầu chuỗi và nơi kết thúc chuỗi. Cấu trúc của nó sẽ như sau:

|  |  |
| --- | --- |
| 1 | /^pattern$/ |

Trong đó ký tự ^ là khai báo bắt đầu chuỗi, còn $ là khai báo kết thúc chuỗi.

**Ký tự bắt đầu chuỗi pattern**

**Ví dụ**:**Bạn cần kiểm tra một chuỗi có bắt đầu bằng chữ techtuts hay không?**

Giả sử chuỗi cần dò tìm là "*Website techtuts.net là blog công nghệ*".

Nếu ta giải bài toán này như cách giải ở phần 2 thì chuỗi pattern sẽ như sau:

|  |  |
| --- | --- |
| 1  2 | let result = /techtuts/.test("Website techtuts.net là blog công nghệ");  console.log(result); // True |

Rõ ràng kết quả không như ta mong muốn, bởi chữ techtuts không phải là chuỗi bắt đầu. Để giải quyết vấn đề này ta sẽ thêm ký tự ^ đặt đằng trước chuỗi pattern.

|  |  |
| --- | --- |
| 1  2 | let result = /^techtuts/.test("Website techtuts.net là blog công nghệ");  console.log(result); // False |

Tóm lại, nếu muốn quy ước tìm từ đầu chuỗi thì bạn sử dụng ký tự ^ nhé.

**Ký tự kết thúc chuỗi pattern**

Quay lại yêu cầu bài toán trên, ta cần kiểm tra xem chuỗi đó có phải kết thúc bằng chữ techtuts hay không. Ta sẽ thêm dấu $ vào vị trí cuối cùng của chuỗi pattern.

|  |  |
| --- | --- |
| 1  2 | let result = /techtuts$/.test("Website techtuts.net là blog công nghệ");  console.log(result); // False |

Kết quả false là chính xác.

**Vậy ta có kết luận như sau**: Nếu không khai báo bắt đầu chuỗi hoặc kết thúc chuỗi thì javascript sẽ so khớp tại bất kì vị trí nào bên trong chuỗi, miễn là có xuất hiện giá trị cần tìm.

**4. Bảng các biểu thức RegEx trong js căn bản**

Trước khi làm việc với những ví dụ thì hãy tham khảo bảng tóm tắt dưới đây. Đây là những biểu thức chính quy regular expression căn bản nhất, được sử dụng nhiều nhất trong lập trình javascript.

| **Biểu thức** | **Ý nghĩa** |
| --- | --- |
| [abc] | Tìm các chữ cái a,b hoặc c |
| [0-9] hoăc [a-z] | Tìm các ký tự từ 0-9 hoặc từ a-z |
| (x|y) | ìm ký tự x hoặc y |
| \d | Tìm các chữ số |
| \s | Tìm khoảng trắng |
| n+ | Tìm 1 hoặc nhiều chữ n liên tiếp nhau |
| n\* | Tìm 0 hoặc nhiều chữ n liên tiếp nhau |
| n? | Tìm 0 hoặc 1 chữ n |

**Ví dụ 1: Kiểm tra trong chuỗi có xuất hiện chữ *Cường* hay không?**

Ví dụ này ta chỉ cần truyền nguyên chuỗi Cường vào pattern là được.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var patt = /Cường/;  if (patt.test("Tác giả Cường")) {      document.write('Trong chuỗi có chữ Cường');  }  else {      document.write('Trong chuỗi không có chữ Cường');  } |

Bạn cũng có thể sử dụng trực tiếp ngay chuỗi pattern như ví dụ sau>

|  |  |
| --- | --- |
| 1 | /freetuts/.test("Chào mừng bạn đến với freetuts.net") |

**Ví dụ 2: Kiểm tra chuỗi có ít nhất một chữ n**

Ví dụ này hơi đặc biệt nên ta sử dụng dấu + trong bảng RegEx căn bản trên, nó có tác dụng kiểm tra một chuỗi có ít nhất 1 ký tự cần tìm kiếm.

**Demo**[**RUN**](https://freetuts.net/editor.html?id=415)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (/n+/.test("hello")) {      document.write('Trong chuỗi có chữ n');  }  else {      document.write('Trong chuỗi không có chữ n');  } |

**Ví dụ 3: Kiểm tra trong chuỗi có xuất hiện ký tự số hay không?**

Với ví dụ này ta có hai cách giải. Cách thứ nhất sử dụng cặp dấu ngoặc [0-9], và cách thứ hai là dùng ký hiệu \d.

Mình sẽ giải bằng cặp dấu ngoặc [0-9].

**Demo**[**RUN**](NULL)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (/[0-9]/.test("hello123")) {      document.write('Trong chuỗi có xuất hiện số');  }  else {      document.write('Trong chuỗi không xuất hiện số');  } |

**Ví dụ 4: Kiểm tra trong chuỗi không hoặc có xuất hiện số**

Bài này ta sẽ kết hợp cặp dấu ngoặc [0-9] và ký hiệu \* trong bảng trên. Rõ ràng trường hợp này thì chuỗi RegEx luôn đúng bởi vì ta có 2 trường hợp XUẤT HIỆN hoặc KHÔNG XUẤT HIỆN.

**Demo**[**RUN**](https://freetuts.net/editor.html?id=417)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (/[0-9]\*/.test("freetuts")) {      document.write('Luôn luôn chạy');  }  else {      document.write('Không bao giờ chạy');  } |

**Ví dụ: Kiểm tra trong chuỗi có chữ H hay không?**

Bài này khá đơn giản là ta chỉ cần truyền chữ H vào như sau:

**Demo**[**RUN**](https://freetuts.net/editor.html?id=418)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (/H/.test("hello")) {      document.write('Có chữ H');  }  else {      document.write('Không có chữ H');  } |

Rõ ràng trong ví dụ có xuất hiện chữ h mà tại sao nó báo không xuất hiện? Lý do là trong chuỗi pattern chúng ta kiểm tra chữ in HOA nhưng trong chuỗi cần kiểm tra thì lại có chữ in thường nên nó trả về **false**. Để giải quyết thì ta thêm chữ i đằng sau chuỗi pattern.

**Demo**[**RUN**](https://freetuts.net/editor.html?id=419)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (/H/i.test("hello")) {      document.write('Có chữ H');  }  else {      document.write('Không có chữ H');  } |

Các bạn xem phần **modifiers** ở trên để biết thêm thông tin.

**5. Bảng các quy tắc RegEx trong javascript khác**

Ở trên mình đã cung cấp các quy tắc thường gặp của biểu thức chính quy rồi, thì phần này mình sẽ tổng hợp lại để các bạn dễ dàng ôn tập nhé.

**Modifiers**

Bảng này gồm 3 ký tự, dùng để khai báo cấu hình của chuỗi pattern.

| **Modifier** | **Description** |
| --- | --- |
| i | So sánh không phân biệt chữ hoa chữ thường (case-insensitive) |
| g | So sánh toàn bộ chuỗi dù trong chuỗi có xuống hàng (global) |
| m | So sánh nhiều dòng (multiline) |

**Brackets**

Bảng các quy tắc liên quan đến dấu ngoặc vuông và ngoặc nhọn.

| **Expression** | **Description** |
| --- | --- |
| [abc] | Tìm các ký tự a, b hoặc c |
| [^abc] | Tìm các ký tự không phải a, b và c |
| [0-9] | Tìm các ký tự là chữ số từ 0-9 |
| [^0-9] | Tìm các ký tự không phải chữ số từ 0-9 |
| (x|y) | Tìm ký tự x hoặc y |

**Metacharacters**

Bảng quy tắc các ký tự đơn.

| **Expression** | **Description** |
| --- | --- |
| . | Tìm ký tự bất kì |
| \w | Tìm ký tự chữ cái |
| \W | Tìm các ký tự không phải là chữ cái |
| \d | Tìm ký tự là chữ số |
| \D | Tìm ký tự không phải là chữ số |
| \s | Tìm ký tự là khoảng trắng |
| \S | Tìm ký tự không phải khoảng trắng |
| \b | Tìm so khớp bắt đầu hoặc kết thúc chuỗi |
| \B | Tìm so khớp không phải bắt đầu hoặc kết thúc chuỗi |
| \0 | Tìm ký tự NULL |
| \n | Tìm ký tự xuống hàng |
| \t | Tìm ký tự tab |

**Quantifiers**

Bảng quy tắc đến số lượng các ký tự.

| **Expression** | **Description** |
| --- | --- |
| + | Kiểm tra ký tự xuất hiện một hoặc nhiều lần |
| \* | Kiểm tra ký tự xuất hiện không hoặc nhiều lần |
| ? | Kiểm tra ký tự xuất hiện không hoặc một lần |
| {X} | Kiểm tra ký tự xuất hiện đúng X lần |
| {X,Y} | Kiểm tra ký tự xuất hiện tối thiểu X lần và tối đa Y lần |
| {X,} | Kiểm tra ký tự xuất hiện ít nhất X lần |
| ^ | Kiểm tra ký tự bắt đầu chuỗi |
| $ | Kiểm tra ký tự kết thúc chuỗi |

Để lấy hết các ví dụ trên thì bài viết sẽ rất dài, nên phần này mình chỉ làm 4 ví dụ cơ bản thôi nhé. Các bạn hãy thực hành theo để suy luận và hiểu được ý nghĩa của từng quy tắc.

**Ví dụ 1: Kiểm tra chuỗi xuất hiện ít nhất 10 chữ N liên tiếp và không phân biệt chữ hoa chữ thường**

Ta sử dụng cú pháp {10,} để kiểm tra ít nhất 10 chữ N liên tiếp.

**Demo**[**RUN**](https://freetuts.net/editor.html?id=420)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var pattern = /n{10,}/i;  if (pattern.test("10 chu n la nnnnnnnnnn")) {      document.write('Chuỗi có NHIỀU hơn 10 chữ n');  }  else {      document.write('Chuỗi có ÍT hơn 10 chữ n');  } |

**Bài 2: Kiểm tra chuỗi có phải là 'freetuts.net' hay không**

Để kiểm tra chính xác chuỗi là '*freetuts.net*' thì bắt buộc ta phải thêm ký tự bắt đầu ^ và kết thúc $ để cố định chuỗi lại.

Ký tự . trong chuỗi '*freetuts.net*' là ký tự đặc biệt, vì nó có trong danh sách các biểu thức của RegExp nên bắt buộc ta phải thêm dấu \ đằng trước nó.

**Demo**[**RUN**](https://freetuts.net/editor.html?id=421)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var pattern = /^freetuts\.net$/i;  if (pattern.test("freetuts.net")) {      document.write('Chuỗi freetuts.net');  }  else {      document.write('Không phải chuỗi freetuts.net');  } |

**Bài 3: Kiểm tra chuỗi là các chữ số và dài 8 ký tự**

Để kiểm tra các chữ số ta dùng [0-9] và chiều dài 8 ký tự nên ta dùng {8}.

**Demo**[**RUN**](https://freetuts.net/editor.html?id=422)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var pattern = /^[0-9]{8}$/;  if (pattern.test("12345678")) {      document.write('Các số dài 8 ký tự');  }  else {      document.write('Không phải là số hoặc ngắn hơn 8 ký tự');  } |

**Bài 4: Kiểm tra chuỗi định dạng mã thẻ cào xxxx-xxxx-xxxx-xxxx và x chính là các chữ số.**

Ta sử dụng [0-9] để kiểm tra là chữ số. Và các cặp xxxx có chiều dì là 4 nên ta sử dụng {4}.

**Demo**[**RUN**](https://freetuts.net/editor.html?id=423)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var pattern = /^[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{4}$/;  if (pattern.test("1234-1232-1321-2312")) {      document.write('Mã thẻ hợp lệ');  }  else {      document.write('Mã thẻ không hợp lệ');  } |

**6. Dùng hàm string.match lấy kết quả toàn bộ chuỗi regex**

Ngoài phương thức test() ở trên thì Javascript còn một phương thức khác khá hay, đó là là phương thức match() trong đối tượng chuỗi string.

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 như sau:**

|  |  |
| --- | --- |
| 1 | string.match(pattern); |

Trong đó string là chuỗi cha, còn pattern chình là chuỗi RegEx.

Kết quả hàm này trả về một mảng các giá trị so khớp với chuỗi pattern.

**Ví dụ 1**: Lấy tất cả các chữ số trong chuỗi "*số điện thoại 0979306603*".

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | let 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, vơi tôi sinh năm 92";  result = string.match(/[0-9]+/img);  console.log(result);  // ["90", "92"] |

**Ví dụ 2**: Nếu không tìm thấy thì sẽ trả về giá trị **null**.

|  |  |
| --- | --- |
| 1  2  3 | string = "tôi đang học regular expression";  result = string.match(/[0-9]+/img);  console.log(result);  // null |

**7. Dùng hàm regex.exec lấy kết quả từng group chuỗi RegEx**

Hàm string.match() có vẻ hay, nhưng nó không thực sự hữu dụng khi làm việc với những bài toán trong thực tế. Hàm này trả về kết quả là toàn bộ của chuỗi pattern, nên nếu bạn muốn lấy kết quả của một quy tắc trong chuỗi pattern thôi thì làm thế nào? Đừng lo, hàm regex.exec() sẽ giúp bạn xử lý việc này.

Cú pháp của hàm này như sau:

|  |  |
| --- | --- |
| 1 | /pattern/.exec(string) |

Trong đó, pattern chính là chuỗi biểu thức chính quy, còn string chính là chuỗi cần tìm.

**Ví dụ**: Viết biểu thức chính quy lấu đường dẫn URL bên trong thẻ img dưới đây.

|  |  |
| --- | --- |
| 1 | <img src="<https://freetuts.net/public/logo/logo.png>" /> |

Nhìn qua cấu trúc chuỗi thì ta sẽ có chuỗi pattern như sau:

|  |  |
| --- | --- |
| 1 | /<img src="(.+)" \/>/img |

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

|  |  |
| --- | --- |
| 1  2  3 | let string = '<img src="<https://freetuts.net/public/logo/logo.png>" />';  var result = /<img src="(.+)" \/>/img.exec(string);  console.log(result); |

Kết quả là một mảng gồm hai phần tử:

Trong đó:

* Phần tử đầu tiên chính là kết quả của toàn bộ chuỗi pattern /<img src="(.+)" \/>/,
* Phần tử thứ hai là kết quả của quy tắc regex đầu tiên nằm bên trong cặp (), đó chính là (.+).

Để hiểu được ví dụ này thì bạn phải biết khái niệm **capturing group** trong regular expression.

**Capturing group** chính là tập hợp những nhóm trong chuỗi pattern. Nhóm đầu tiên là toàn bộ chuỗi, nhóm thứ hai là công thức nằm bên trong cặp () đầu tiên, nhóm thứ ba là công thức nằm bên trong cặp () thứ hai ... cứ như vậy cho đến hết chuỗi pattern.

**Ví dụ**: Chuỗi pattern /<img(.+)src="(.+)"(.\*)\/>/ sẽ có các group sau:

1. /<img(.+)src="(.+)"(.\*)\/>/ - toàn bộ chuỗi.
2. (.+) - đầu tiên
3. (.+) - thứ hai
4. (.\*) - thứ ba

Vậy, kết quả mà hàm exec trả về chính là một mảng, mỗi phần tử trong mảng là một capturing group, trong chuỗi pattern có bao nhiêu group thì nó sẽ trả về bấy nhiêu phần tử.