- Tạo một Biểu thức chính quy: Sử dụng hàm RegExp hoặc sử dụng 2 dấu /biểu thức/, ví dụ:

Var bieuthuc = new RegExp(“bieu thuc”);

Var bieuthuc = /bieuthuc/;

- Để thêm các ký tự đặc biệt có ý nghĩa đặc biệt trong biểu thức chính quy như +, ? ta phải thêm dấu gạch ngược ‘\’, ví dụ: \+

- Biểu thức chính quy có phương thức ‘test’ trả về True hoặc False, ví dụ:

Console.log(/abc/.test('abcde'));

> True

Console.log(/abc/.test(‘abxde’));

> False

- Khớp 1 bộ ký tự: Trong 1 biểu thức chính quy, phần biểu thức có chứa dấu ngoặc vuông sẽ khớp với bất kỳ ký tự nào chứa trong dấu ngoặc vuông, ví dụ:

biểu thức sau: abc[123456]def tương đương với các biểu thức: abc1def, abc2def, abc3def…

- Trong 2 dấu ngoặc vuông, dấu ‘-’ biểu thị 1 phạm vi ký tự, ví dụ:

[0-9] tương đương [0123456789]

- Các ký hiệu trong biểu thức chính quy:

\d: ký tự số, \D: ngược lai, không phải ký tự số

\w: ký tự số và chữ, \W: ngược lại, không phải ký tự số và chữ như ‘.

\s: Bất kỳ ký tự khoảng trắng (dấu cách, tab, new line…),\S: ngược lại.

. Tất cả các ký tự trừ Newline

- Đề đảo ngược bộ ký tự ta sử dụng dấu mũ sau dấu ngoặc vuông mở: [^]

ví dụ: console.log(/[^01]/.test(‘010101010110101’));

>False

- Sử dụng ký tự + và ký tự \* để lặp các thành phần trong biểu thức chính quy,

ví dụ:

console.log(/’\d+’/.test(“‘123’”)); > True

console.log(/’\d\*’/.test(“‘123’”)); > True

+ khác \* ở chỗ: + có số lần lặp ít nhất là 1, \* cho phép số lần lặp là 0 (tức là khớp với biểu thức rỗng (‘’);

console.log(/’\d+’/.test(“‘’”)); > false

console.log(/’\d\*’/.test(“‘’”)); > true

- Dấu ? trong biểu thức chính quy cho phép 1 phần của biểu thức có thể xuất hiện 1 lần hoặc không xuất hiện. Ví dụ:

var bt = /hoangvan?tu/;

console.log(bt.test(‘hoangvantu’)); > true

console.log(bt.test(‘hoangvatu’)); > true

- Để ấn định số lần lặp lại 1 thành phần của biểu thức chính quy ta thêm cú pháp sau vào thành phần đó {a}, a là số lần lặp. Để giới hạn số lần lặp trong 1 phạm vi, sử dụng cú pháp {a,b}, lặp ít nhất a lần, nhiều nhất là b lần. Các trường hợp khác như: {,b}: lặp ít nhất là 0, nhiều nhất là b; {a,}: số lần lặp lớn hơn hoặc bằng a

- Để sử dụng toán tử + hoặc \* trên nhiều phần từ cùng 1 lúc, ta sử dụng dấu ngoặc đơn (), một phần của biểu thức chính quy được đặt trong dấu ngoặc đơn được tính là 1 phần tử duy nhất đối với các toán tử sau nó. ví dụ

var cartoonCrying = /boo+(hoo+)+/i;

console.log(cartoonCrying.test("Boohoooohoohooo")); > True

>> Ký tự i sau biểu thức chính quy để không phân biệt chữ hoa, chữ thường.

- Phương thức exec và phương thức math:

Hai phương thức trên trả về 1 mảng có 1 phần tử là 1 chuỗi con trong 1 chuỗi cho trước khớp với biểu thức chính quy.

Ví dụ:

var ex = /\d+/.exec(‘hoang van tu 123 tu hoang van 15’);

console.log(ex); >[“123”]

console.log("one two 100".match(/\d+/));

// . ["100"]

- Khi biểu thức chính quy chứa các biểu thức con được nhóm bằng dấu ngoặc đơn thì văn bản khớp với nhóm đó cũng được hiển thị trong mảng. Phần tử Khớp với toàn bộ biểu thức luôn được hiển thị đầu tiên, sau đó là phần tử khớp với nhóm thứ nhất, rồi thứ hai. Ví dụ:

var quotedText = /'([^']\*)'/;

console.log(quotedText.exec("she said 'hello'"));

> ["'hello'", "hello"]

- Khi nhóm cuối cùng không khớp với chuỗi, kết quả sẽ sinh ra undefiled, ví dụ

console.log(/bad(ly)?/.exec(“bad”)); > [“bad”,undefiled]

- Tương tự, khi 1 nhóm được khớp nhiều lần, chỉ có khớp cuối cùng được xuất hiện trong mảng. Ví dụ:

console.log(/(\d)+/.exec(“123”)); > [“123”, “3”];

- Dấu mũ ^ và $ thể hiện bắt đầu và kết thúc 1 chuỗi, ví dụ:

console.log(/^\d+$/.exec(“123”));>[“123”] khớp với chuỗi bắt đầu là 1 số, kết thúc là 1 số, /^!/ khớp với chuỗi bắt đầu bằng !. Cú pháp sau là sai: /x^/ vì không thể có x trước sự bắt đầu của 1 chuỗi.

- \b là điểm đánh dấu danh giới, tương tự như ^ và $, cú pháp /\bcat\b/

- Ký tự ống (|) cho phép lựa chọn các giá trị, tương tự toán tử hoặc, cú pháp:

var animacount = /\b\d+ (chicken|pig|cow)s?\b/;

console.log(animacount.test(“12 cows”)); > true

console.log(animacount.test(“15 cow”)); > true

- Phương thức replace: tìm kiếm trong chuỗi khớp với 1 biểu thức chính quy rồi thay thế bởi một ký tự khác. Ví dụ:

console.log("Borobudur".replace(/[ou]/, "a")); //Barobudur

console.log("Borobudur".replace(/[ou]/g, "a")); // . Barabadar

Khi một toán tử g được gán như ví dụ trên, toàn bộ các phần khớp trong chuỗi sẽ được thay thế, không phải chỉ phần tử đầu tiên.

- Replace 1 hàm:

var stock = "1 lemon, 2 cabbages, and 101 eggs";

function minusOne(match, amount, unit) {

amount = Number(amount) - 1;

if (amount == 1) // only one left, remove the 's'

unit = unit.slice(0, unit.length - 1);

else if (amount == 0)

amount = "no";

return amount + " " + unit;

}

console.log(stock.replace(/(\d+) (\w+)/g, minusOne));

Chú ý: match là toàn bộ biểu thức chính quy, amount là nhóm thứ nhất, unit là nhóm thứ 2 (Đây là quy tắc hàm thay thế nhận biến để xử lý)

- các toán tử + \* ? {} là người tham lam vì nó cố gắng khớp nhiều nhất có thể và quay lại từ đó. Bằng cách thêm ? vào các toán tử trên, chúng sẽ trở thành khớp ít nhất có thể, chỉ khớp nhiều hơn khi không khớp được ít hơn. (+?, \*?, ??, {}?)

- Sử dụng hàm RegExp để tạo biểu thức chính quy khi không biết cụ thể biểu thức chính quy mà chỉ biết tên biến (Tức là biểu thức chính quy động). Ví dụ:

var name = “hary”;

var text = “hary is a suspicious character”;

var reg = new RegExp(“\\b(” + name + “)\\b”, “gi”);

console.log(text.replace(reg, \_$1\_));

Lưu ý: Hai gạch ngược (\\) khi viết biểu thức trong hàm RegExp tương đương với một gạch ngược (\) khi viết biểu thức trong hai dấu / …/

Bên ngoài biến phải có dấu ngoặc đơn ()

$& là nhóm nào ứng với nhóm đó, $1, $2 là viết cụ thể các nhóm

- Phương thức Seach: phương thức indexOf của chuỗi không thể gọi được bằng biểu thức chính quy, nhưng có một phương thức khác, seach, có thể sử dụng biểu thức chính quy, cũng giống như indexOf, nó trả về index đầu tiên mà biểu thức tìm thấy, nếu không tìm thấy trả về -1. Ví dụ:

console.log(" word".search(/\S/)); // . 2

console.log(" ".search(/\S/)); // . -1

- lastindex là vị trí của biểu thức chính quy khi bắt đầu khớp. nếu khớp được, lastindex sẽ tự động cập nhật giá trị, nếu không khớp được lastindex sẽ trở về 0.

- Phương thức exec khác phương thức match ở cú pháp: biểu thức.exec(“String”);

“string”.match(Biểu thức); nếu biểu thức có đặt biến toàn cục g, thì phương thức match sẽ trả về 1 mảng gồm tất cả các giá trị khớp được.