APRENDERAPROGRAMAR.COM

EXPRESIONES REGULARES JAVASCRIPT. REGEX. NEW. CARÁCTER ESPECIAL. NÚMERO, LETRA, ESPACIO BLANCO. (CU01154E)

Sección: Cursos
Categoría: Tutorial básico del programador web: JavaScript desde cero

Fecha revisión: 2029

Resumen: Entrega nº54 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

EXPRESIONES REGULARES JAVASCRIPT

En muchas ocasiones nos encontramos con la necesidad de hacer comprobaciones respecto a una cadena de texto. Por ejemplo comprobar si contiene letras mayúsculas, validar si es una dirección de correo electrónico correctamente formada… Estas comprobaciones también pueden servir para autocompletar cuando el usuario está escribiendo una entrada en un formulario.

Las comprobaciones sobre cadenas de texto se pueden hacer de muy distintas maneras y pueden tener muy distintas utilidades. En un nivel muy amplio, el estudio de cadenas, sus patrones, formas de construcción, etc. lo podríamos englobar dentro de la temática de "Lenguajes formales, expresiones regulares, autómatas y gramáticas".

Nosotros vamos simplemente a indicar las cuestiones más relevantes relacionadas con la posibilidad de uso de expresiones regulares dentro de JavaScript, sin entrar en su análisis desde el punto de vista formal o teórico.

Una expresión regular es un patrón: una expresión que define todas las cadenas o subcadenas que cumplen con ciertos criterios. Por ejemplo, un criterio puede ser "contener una a". Hemos de tener en cuenta que a efectos de análisis de cadenas de texto no es lo mismo a (minúscula) que A (mayúscula) ó á (con tilde). Es decir, una letra no es lo mismo que esa letra en mayúsculas o esa letra con tilde, diéresis o cualquier otro signo de puntuación. De hecho, quienes estén habituados a trabajar con códigos de caracteres como ASCII ó Unicode ya sabrán que una letra minúscula tiene distinto código numérico que esa misma letra mayúscula.

Una forma inicial de expresar un patrón podría ser la indicada en la siguiente tabla:

Patrón	Cadena a probar	¿Cumple el patrón?
Todas las cadenas que comienzan por una letra distinta de a y contienen al menos una a	Camión	Sí
	Abeto	No, dado que no contiene una a minúscula
	UEFA	No, ya que aunque contiene la A no cumple el requisito: contener una a minúscula.
	Abeja	Sí, ya que comienza con una letra A mayúscula y contiene una a minúscula
	BATHa	Sí, cumple los requisitos
	"apis mellifera sativa"	No, no cumple los requisitos*.
	\"apis mellifera sativa\"	Sí, cumple los requisitos*.

Hemos indicado "apis mellifera sativa" entrecomillada para indicar que la cadena está compuesta por varias palabras separadas por espacios en blanco. En este caso, la cadena contiene dos espacios en blanco. Hemos escrito \"apis mellifera sativa\" para representar una cadena donde las comillas forman parte de la cadena (pero no las barras inclinadas, que son el símbolo de escape que nos sirve para indicar que las comillas deben leerse literalmente). Esta cadena comienza con una comilla y contiene al menos una a minúscula, por lo que cumple con los requisitos.

Tener en cuenta que en una expresión regular los espacios en blanco cuentan. Por tanto no es lo mismo "apis mellifera sativa" que "apis mellifera sativa". Un solo espacio en blanco hace que dos cadenas se consideren distintas.

Las expresiones regulares son formas estandarizadas de expresar un patrón siguiendo ciertas reglas para definirlo. A un computador no podemos indicarle con una frase cuál es el patrón que queremos comprobar, tendremos que indicárselo con una sintaxis normalizada.

Es frecuente encontrar código donde se hace alusión a los siguientes términos:

regex: abreviatura de regular expression (expresión regular)

pattern: patrón

match: coincidencia (cadena que contiene el patrón).

Una expresión regular como “Todas las cadenas que comienzan por una letra distinta de a y contienen al menos una a” es relativamente compleja. Las expresiones regulares más sencillas son las que hacen alusión a si una cadena contiene un patrón. Por ejemplo: ¿Contiene una cadena la subcadena rio? Serían cadenas que contienen la subcadena rio y hacen match para la expresión regular indicada: “armariote”, “canario”, “rioboo”, “ariofonte”, etc. En cambio no contienen la subcadena y no hacen match cadenas como “CABRIOLET” (por estar en mayúsculas), “rico”, “dañino” ni “río”, en este último caso por llevar tilde.

EL OBJETO PREDEFINIDO REGEX

Para crear expresiones regulares JavaScript provee de una serie de símbolos especiales que permiten definir los patrones, como veremos más adelante.

Para introducir el concepto de símbolo especial y expresión regular vamos a usar primeramente sólo dos caracteres especiales: el carácter “punto” . , que es un simple punto, y el carácter ? que sirve para indicar que el carácter a continuación del símbolo es opcional, es decir, que puede aparecer 0 ó 1 vez.

En JavaScript el uso de expresiones regulares se basa en el uso de un objeto predefinido (existente) del lenguaje que podemos usar para construir nuestros scripts: el objeto RegExp. Este es uno de los objetos predefinidos de JavaScript, ya que existen otros (como Math, Date, etc.). RegExp nos provee de métodos útiles para trabajar con expresiones regulares.

JavaScript permite crear expresiones regulares de dos maneras:

a) En forma de literal: var miExpresionRegular = /as?.a/ representa a todas las cadenas que contienen una subcadena con la primera letra de la subcadena una a, seguida de una s, opcionalmente seguida de cualquier letra, y seguida de una a.

b) Instanciando el objeto RegExp: var miExpresionRegular = new RegExp("as?.a") representa lo mismo.

Una cadena como “casamentero” contiene el patrón: contiene una a, seguida de s, el carácter opcional no está presente, y seguida una a. c-**a-s-a**-m-e-n-t-e-r-o

Una cadena como “castaño” contiene el patrón: c-**a-s-t-a**-ñ-o contiene el patrón, siendo el carácter opcional la letra t.

También contienen el patrón asa, casta, masa, castañuela, casiarina, kaspamina ó asma.

Cadenas como “sabina”, “casualidad” ó “as” no contienen el patrón.

Escribe este código, guárdalo como archivo html y comprueba sus resultados. Observa cómo el método test aplicado a un objeto de tipo RegExp pasándole una cadena devuelve true si la cadena contiene el patrón definido por la expresión regular o false si la cadena no contiene el patrón.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
function ejemplo() {
var miExpReg = /as?.a/
var cadena = 'casamentero';
var msg = 'Patrón: as?.a \n';
msg = msg + '¿Contiene casamentero el patrón? : '+ miExpReg.test(cadena) +'\n';
msg = msg + '¿Contiene castaño el patrón? : '+ miExpReg.test('castaño') +'\n';
msg = msg + '¿Contiene sabina el patrón? : '+ miExpReg.test('sabina') +'\n';
msg = msg + '¿Contiene asa el patrón? : '+ miExpReg.test('asa') +'\n';
msg = msg + '¿Contiene as el patrón? : '+ miExpReg.test('as') +'\n';
alert(msg);
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemplo()"> Probar </div>
</body>
</html>
```

El resultado esperado es: Patrón: as?.a >> ¿Contiene casamentero el patrón? : true

¿Contiene castaño el patrón? : true >> ¿Contiene sabina el patrón? : false

¿Contiene asa el patrón? : true >> ¿Contiene as el patrón? : false

CARACTERES ESPECIALES EN EXPRESIONES REGULARES

Los principales caracteres especiales para construir expresiones regulares JavaScript están reflejados en la siguiente tabla. Los paréntesis tienen un significado especial que veremos en la siguiente entrega.

Carácter	Significado	Ejemplo aprenderaprogramar.com
****	Backslash o carácter de escape. Sirve para anular un carácter especial y hacer que se considere como si fuera un carácter normal.	Ver más abajo
a	Donde a es una letra cualquiera. Indica que el patrón incluye una a en el orden especificado.	barco Cadenas que tienen a 'barco' como subcadena (incluido barco mismo)
{n}	El carácter anterior aparece exactamente n veces, siendo n un entero positivo.	ca{3}t{3}e Hace match si una subcadena es caaattte.
{n,}	El carácter anterior aparece n o más veces, siendo n un entero positivo.	ca{3,}te Hace match con caaate, caaaate, etc. pero no con cate
{n,m}	El carácter anterior aparece un mínimo de n veces y un máximo de m veces, siendo n y m enteros positivos.	ca{2,5}te Hace match con caate, caaate, caaaate y caaaaate
^	El símbolo ^ (denominado "exponente") indica comienzo de la cadena por ese símbolo.	/^a/ Cadenas que empiezan por a minúscula.
.	El símbolo punto indica existencia de cualquier carácter.	ca.e Cadenas que contienen subcadenas con caxe donde x es cualquier carácter.
$	El símbolo dólar indica que la letra anterior ha de ser obligatoriamente última letra de la cadena	^a..e$ Cadenas que empiezan por a y terminan con e, y que contienen exactamente cuatro caracteres (la a inicial, dos intermedios, más la e final)
*****	El símbolo asterisco indica que la subcadena contiene el símbolo al que precede cero o más veces.	^at*e$ Cadenas que empiezan por a, terminan por e, y entre ambos tienen cero, una o muchas t´s (pero no ningún otro carácter).
{0,}	Equivalente al símbolo asterisco	^at{0,}e$ Igual que el anterior
+	El símbolo más indica que la subcadena contiene el símbolo al que precede una o más veces.	^at+e$ Cadenas que empiezan por a, terminan por e, y entre ambos tienen una o muchas t´s (pero no ningún otro carácter).
{1,}	Equivalente al símbolo +	^at{1,}e$
?	El símbolo interrogación indica opcionalidad: el carácter que lo precede puede aparecer 0 ó 1 vez en la subcadena.	^at?e$ Cadenas que empiezan por a, terminan por e, y entre ambos tienen una ninguna t.
{0,1}	Equivalente al símbolo ?, es decir, el carácter anterior puede aparecer 0 ó 1 vez.	^at{0,1}e$
a|b	El símbolo or genera un match si se encuentra a ó b siendo a ó b dos caracteres cualesquiera que se indiquen.	^a(t|g)e$ Cadenas que empiezan por a, terminan por e, y entre ambos hay una t ó una g.

Carácter	Significado	Ejemplo aprenderaprogramar.com
sub1(?=sub2)	Hace match sólo si existe una subcadena donde la subcadena sub1 está antes de la subcadena sub2	mahatma(?= ghandi) Hace match con mahatma ghandi pero no con mahatma prasha
sub1(?!sub2)	Hace match en todas las subcadenas donde la subcadena sub2 no está después de la subcadena sub1	mahatma(?! ghandi) Hace match con mahatma para, mahatma gon, etc. pero no con mahatma ghandi
[abc]	Hace match con cualquiera de los caracteres indicados dentro de los corchetes (conjunto de caracteres)	^[aeiou] Cadenas que empiezan por una vocal.
[a-z]	Rango de caracteres. Hace match con cualquier carácter comprendido entre el inicial y el final, en orden alfabético. Se pueden definir varios conjuntos uno detrás de otro, por ejemplo: [A-Za-z] indica “letra mayúscula o minúscula”.	^[c-k] Cadenas que empiezan por c, d, e, f, g, h, i, j ó k.
[^abc] ó [^a-z]	Complementario o negado de un conjunto de caracteres. Hace match con cualquier carácter distinto de los definidos en el conjunto de caracteres	[^au]$ Hace match con cualquier cadena que no termina en a ni en u
\d	Hace match con cualquier número entre 0 y 9	^\d*\d$ Hace match si la cadena es un número seguido del símbolo * y seguido de otro número. * no es aquí especial por estar precedido del escape \.
[0-9]	Equivalente a \d	^[0-9]* [0-9]$
\D	Hace match con cualquier carácter que no sea un dígito. Equivale a [^0-9]	^ [^0-9] Cadena que no empieza por un número.
\s	Hace match con un espacio en blanco (incluye tabuladores y saltos de línea entre otros).	\s\s Hace match si encuentra dos espacios en blanco
\S	Hace match con cualquier carácter que no sea espacio en blanco, tabulador, salto de línea...	[\s\S] Hace match con cualquier carácter
\w	Hace match con cualquier letra mayúscula, minúscula, número o guión bajo. Equivale a [A-Za-z0-9_]. Tener en cuenta que las letras con tilde quedan fuera y habría que añadirlas si queremos.	\w{3} Hace match si existe una subcadena con 3 caracteres válidos seguidos.
\W	Hace match con cualquier carácter que no sea letra mayúscula, minúscula, número o guión bajo. Equivale a [^A-Za-z0-9_]	\W{2,3} Hace match si existe una subcadena con 2 ó 3 caracteres seguidos que no son letras ni números.
Otros	Existen otros símbolos especiales y combinaciones de símbolos especiales para lograr determinados fines.	

EJERCICIO

Dada la expresión regular de JavaScript /^[A-C]\w+\ses\s\w+/ indicar cuáles de las siguientes cadenas hacen match con la expresión regular, y en caso de hacer match, qué parte o partes son las que hacen match:

a) Juan es guapo

b) Adriano no es feo

c) Adriano deja de ser guapo

d) Adriano ya es guapo

e) No es ahora

f) Ahora es no

g) Adriano es guapo

Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01155E

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206