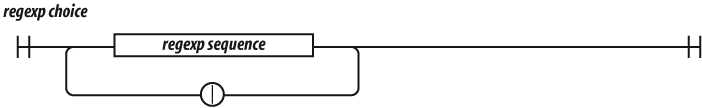
Elements

# Elementos

Veamos más detenidamente los elementos que componen las expresiones regulares.

## Elección Regexp

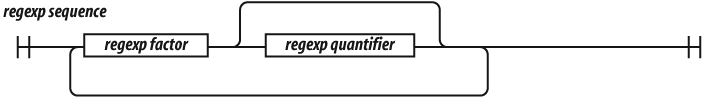


A *regexp choice* contiene uno o varios *regexp sequences*. Las secuencias están separadas por el carácter | (barra vertical). La elección coincide si alguna de las secuencias coincide. Intenta hacer coincidir cada una de las secuencias en orden, por lo que:

"into".match(/in|int/)

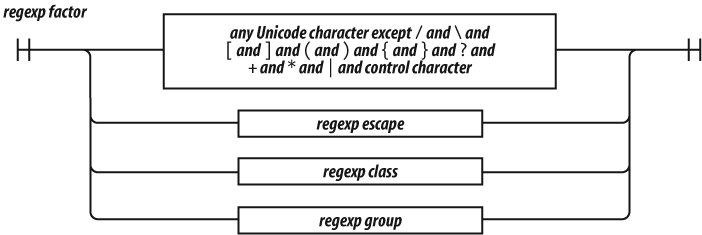
coincide con in en into. No coincidiría con intporque la coincidencia de in tuvo éxito.

## Secuencia Regexp



A *regexp sequence* contiene una o más secuencias *regexp factors*. Cada factor puede ir seguido opcionalmente de un cuantificador que determina cuántas veces se permite que aparezca el factor. Si no hay cuantificador, el factor se comparará una vez.

## Factor Regexp



A *regexp factor* puede ser un carácter, un grupo entre paréntesis, una clase de caracteres o una secuencia de escape. Todos los caracteres se tratan literalmente, excepto los caracteres de control y los caracteres especiales:

\ / [ ] ( ) { } ? + \* | . ^ $

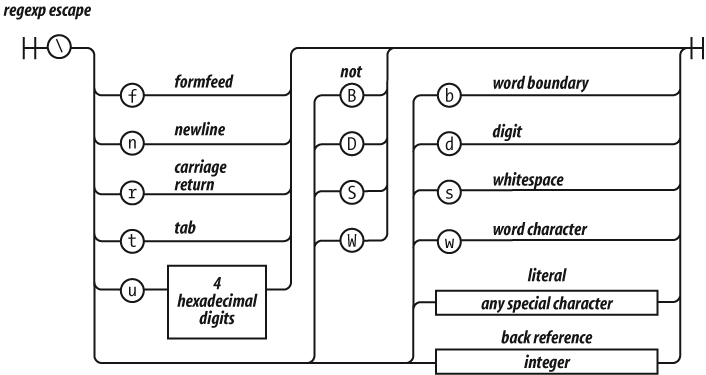
que deben escaparse con un prefijo \ si se quiere que coincidan literalmente. En caso de duda, cualquier carácter especial puede recibir un prefijo \ para hacerlo literal. El prefijo \ *does not* hace que las letras o los dígitos sean literales.

Un . sin mayúsculas coincide con cualquier carácter excepto con un carácter de final de línea.

Un ^ sin esconder coincide con el principio del texto cuando la propiedad lastIndex es cero. También puede coincidir con caracteres de final de línea si se especifica la bandera m.

Un $ sin mayúsculas coincide con el final del texto. También puede coincidir con caracteres de final de línea si se especifica la opción m.

## Escape Regexp



El carácter barra invertida indica escape tanto en los factores regexp como en las cadenas, pero en los factores regexp funciona de forma un poco diferente.

Como en las cadenas, \f es el carácter de salto de línea, \n es el carácter de nueva línea, \r es el carácter de retorno de carro, \t es el carácter de tabulación y \u permite especificar un carácter Unicode como una constante hexadecimal de 16 bits. En los factores regexp, \b*is not* es el carácter de retroceso.

\d es igual que [0-9]. Coincide con un dígito. \D es lo contrario: [^0-9].

\s es lo mismo que [\f\n\r\t\u000B\u0020\u00A0\u2028\u2029]. Es un conjunto parcial de caracteres Unicode de espacio en blanco. \S es lo contrario: [^\f\n\r\t\u000B\u0020\u00A0\u2028\u2029].

\w es lo mismo que [0-9A-Z\_a-z]. \W es lo contrario: [^0-9A-Z\_a-z]. Se supone que representa los caracteres que aparecen en las palabras. Por desgracia, la clase que define es inútil para trabajar con prácticamente cualquier lenguaje real. Si necesitas emparejar una clase de letras, debes especificar tu propia clase.

Una clase de letra sencilla es [A-Za-z\u00C0-\u1FFF\u2800-\uFFFD]. Incluye todas las letras de Unicode, pero también miles de caracteres que no son letras. Unicode es grande y complejo. Una clase de letras exacta del Plano Básico Multilingüe es posible, pero sería enorme e ineficaz. Las expresiones regulares de JavaScript proporcionan un soporte extremadamente pobre para la internacionalización.

\b pretendía ser un anclaje de límites de palabras que facilitara la búsqueda de coincidencias de texto en los límites de las palabras. Por desgracia, utiliza \w para encontrar los límites de las palabras, por lo que es completamente inútil para las aplicaciones multilingües. Esta parte no es buena.

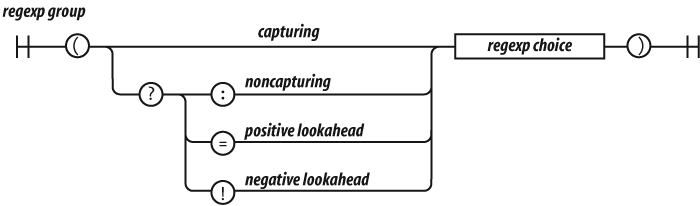
\1 es una referencia al texto que fue capturado por el grupo 1, de modo que se pueda cotejar de nuevo. Por ejemplo, podrías buscar palabras duplicadas en el texto con:

var doubled\_words = /([A-Za-z\u00C0-\u1FFF\u2800-\uFFFD]+)\s+\1/gi;

doubled\_words busca apariciones de palabras (cadenas que contienen 1 o más letras) seguidas de espacios en blanco seguidos de la misma palabra.

\2 es una referencia al grupo 2, \3 es una referencia al grupo 3, y así sucesivamente.

## Grupo Regexp



Hay cuatro tipos de grupos:

*Capturing*

Un grupo de captura es una elección regexp envuelta entre paréntesis. Los caracteres que coincidan con el grupo serán capturados. A cada grupo de captura se le asigna un número. La primera captura ( de la expresión regular es el grupo 1. El segundo ( capturador de la expresión regular es el grupo 2.

*Noncapturing*

Un grupo de no captura tiene un prefijo (?:. Un grupo no capturador simplemente coincide; no captura el texto coincidente. Los grupos no capturadores no interfieren con la numeración de los grupos capturadores.

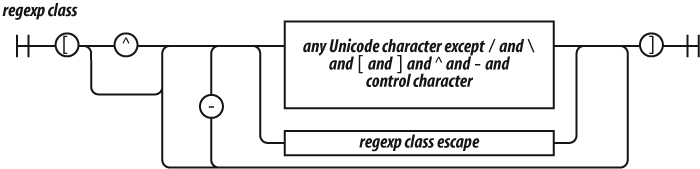
*Positive lookahead*

Un grupo de búsqueda positiva tiene un prefijo (?=. Es como un grupo de no captura, excepto en que después de que el grupo coincida, el texto se rebobina hasta donde empezó el grupo, sin que coincida nada. Esta parte no es buena.

*Negative lookahead*

Un grupo de búsqueda negativa tiene un prefijo (?!. Es como un grupo de búsqueda positiva, salvo que sólo coincide si no coincide. Esta parte no es buena.

## Clase Regexp



A *regexp class* es una forma cómoda de especificar uno de un conjunto de caracteres. Por ejemplo, si quisiéramos que coincidiera con una vocal, podríamos escribir (?:a|e|i|o|u), pero es más cómodo escribirlo como la clase [aeiou].

Las clases ofrecen otras dos ventajas. La primera es que se pueden especificar rangos de caracteres. Así, el conjunto de 32 caracteres especiales ASCII:

! " # $ % & ' ( ) \* +, - . / :  
; < = > ? @ [ \ ] ^ \_ ` { | } ˜

podría escribirse como

(?:!|"|#|\$|%|&|'|\(|\)|\\*|\+|,|-|\.|\/|:|;|<|=|>|@|\[|\\|]|\^|\_|` |\{|\||\}|˜)

pero es ligeramente más agradable escribirlo como:

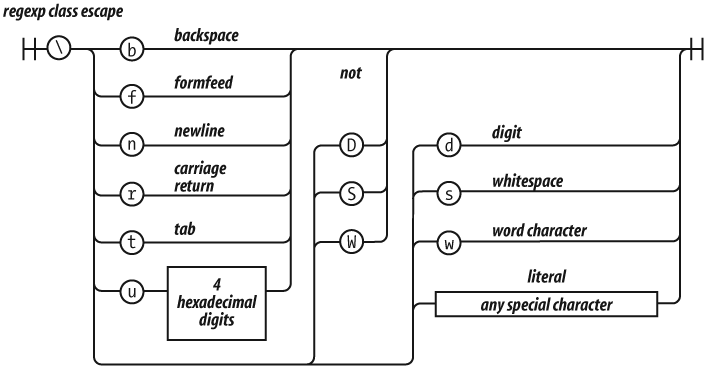
[!-\/:-@\[-`{-˜]

que incluye los caracteres de !a / y de : a @ y de [ a ` y de { a ˜. Sigue teniendo un aspecto bastante desagradable.

La otra comodidad es la complementación de una clase. Si el primer carácter después de [ es ^, entonces la clase excluye los caracteres especificados.

Así que [^!-\/:-@\[-`{-˜] coincide con cualquier carácter que sea *not* uno de los caracteres especiales ASCII.

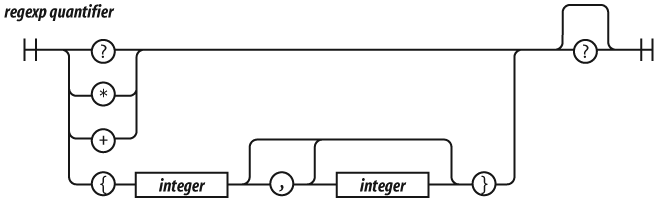
## Escape de clases regexp



Las reglas de escape dentro de una clase de caracteres son ligeramente diferentes a las de un factor regexp. [\b] es el carácter de retroceso. Estos son los caracteres especiales que deben escaparse en una clase de caracteres:

- / [ \ ] ^

## Cuantificador Regexp



A *regexp factor* puede tener un *regexp quantifier* que determina cuántas veces debe coincidir el factor. Un número entre llaves significa que el factor debe coincidir ese número de veces. Así, /www/ coincide igual que /w{3}/. {3,6} coincidirá 3, 4, 5 ó 6 veces. {3,} coincidirá 3 o más veces.

? es igual que {0,1}. \* es igual que {0,}. + es igual que {1,}.

La concordancia tiende a ser avariciosa, haciendo coincidir tantas repeticiones como sea posible hasta el límite, si lo hay. Si el cuantificador tiene un sufijo extra ?, la concordancia tiende a ser perezosa, intentando concordar el menor número posible de repeticiones. Por lo general, es mejor seguir con la concordancia codiciosa.