Expresiones Regulares

Antonio Espín Herranz

Contenidos

- Expresiones regulares
  - Construcción y aplicación
  - Expresiones regulares en cadenas
  - Validación de datos

Introducción

- Una expresión regular es una secuencia de caracteres que forma un **patrón de búsqueda**.

- El patrón de búsqueda se puede utilizar para operaciones de **búsqueda y reemplazo** de texto.

- Sintaxis:
  - ***/pattern/modifiers;***

  - El patrón a buscar
  - Y los modificadores: por ej. No distinguir entre mayúsculas / minúsculas.

Métodos de cadena

En JavaScript, las expresiones regulares se usan a menudo con los dos **métodos de cadena** : `search()`y `replace()`.

- `search()`método utiliza una expresión para buscar una coincidencia y devuelve la posición de la coincidencia.

- `replace()`método devuelve una cadena modificada donde se reemplaza el patrón.

RegExp

- Se pueden utilizar de dos formas:
  - Con el objeto **RegExp** y los métodos:
    - exec():
    - test() : devuelve true / false
  - Con el objeto **String** y los métodos:
    - match()
    - matchAll()
    - replace()
    - replaceAll()
    - search()
    - split()

Para crear expresiones regulares

- Se puede hacer dos formas:
  - Una expresión literal encerrada entre slashes
    - const re = /ab+c/
  - O creando una instancia de RegExp:
    - const re = new RegExp(‘ab+c’);
    - En este caso se pasa una cadena.

Descripción

Método	Descripción
`exec()`	Ejecuta una búsqueda de una coincidencia en una cadena. Devuelve una matriz de información o `null` en una falta de coincidencia.
`test()`	Pruebas para una coincidencia en una cadena. Vuelve `true` o `false`.
`match()`	Devuelve una matriz que contiene todas las coincidencias, incluidos los grupos de captura, o `null` si no se encuentra ninguna coincidencia.
`matchAll()`	Devuelve un iterador que contiene todas las coincidencias, incluidos los grupos de captura.
`search()`	Pruebas para una coincidencia en una cadena. Devuelve el índice de la coincidencia, o `-1` si la búsqueda falla.
`replace()`	Ejecuta una búsqueda de una coincidencia en una cadena y reemplaza la subcadena coincidente con una subcadena de reemplazo.
`replaceAll()`	Ejecuta una búsqueda de todas las coincidencias en una cadena y reemplaza las subcadenas coincidentes con una subcadena de reemplazo.
`split()`	Utiliza una expresión regular o una cadena fija para dividir una cadena en una matriz de subcadenas.

Patrones

\w	Coincide con cualquier carácter de palabra (alfanumérico).
\W	Coincide con cualquier carácter que no sea una palabra.
\s	Coincide con cualquier carácter de espacio en blanco (tabulación, nueva línea, retorno de carro, avance de página, tabulación vertical).
\S	Coincide con cualquier carácter que no sea un espacio en blanco.
\d	Coincide con cualquier dígito numérico.
\D	Coincide con cualquier carácter que no sea un número.
[\b]	Coincide con un retroceso.
.	Coincide con cualquier carácter excepto una nueva línea.
[...]	Coincide con cualquier carácter dentro de los corchetes.
[^...]	Coincide con cualquier carácter que no esté entre corchetes.
[xy]	Coincide con cualquier carácter en el rango de x a y.
[^xy]	Coincide con cualquier carácter que no esté en el rango de x a y.
{x,y}	Coincide con el elemento anterior al menos x veces pero sin exceder y veces.
{X,}	Coincide con el elemento anterior al menos x veces.
{X}	Coincide con el elemento anterior exactamente x veces.
?	Coincide con el elemento anterior una vez o no coincide en absoluto.
+	Coincide con el elemento anterior al menos una vez.
*	Coincide con el elemento anterior cualquier número de veces o no coincide en absoluto.
|	Coincide con la expresión a la izquierda oa la derecha de | personaje.
(...)	Agrupa todo lo que está entre paréntesis en un subpatrón.
\X	Coincide con los mismos caracteres que resultaron del subpatrón en el grupo número x. Los grupos, que se designan entre paréntesis, se numeran de izquierda a derecha.
^	Coincide con el principio de la cadena o el principio de una línea, en coincidencias multilínea.
ps	Coincide con el final de la cadena o el final de una línea, en coincidencias multilínea.
\b	Coincide con la posición entre un carácter de palabra y un carácter que no es de palabra.
\B	Coincide con la posición que no está entre un carácter de palabra y un carácter que no es de palabra.

Ejemplos

- Tarjeta de crédito:

```
function IsMatchingCard(str){
   var myRegExp =  /[0-9]{4} [0-9]{4} [0-9]{4} [0-9]{4}/;
   return myRegExp.test(str)
}
```

Ejemplos

- Fecha

```
function isValidDate(sText) {
      var reDate = /(?:0[1-9]|[12][0-9]|3[01])\/(?:0[1-9]|1[0-2])\/(?:19|20\d{2})/;
      return reDate.test(sText);
   }
```

La expresión se delimita entre / /

(?:x)	Grupo sin captura (no se almacena en una propiedad group. El ejemplo anterior
(?<Name>x)	Grupo de captura nombrado. Almacena el valor en groups identificado por <Name>.

/(?:0[1-9]|[12][0-9]|3[01])\/(?:0[1-9]|1[0-2])\/(?:19|20\d{2})/;

cadena.match(ExpReg); -> El objeto o /patron/

Ejemplos

- const aliceExcerpt = 'The Caterpillar and Alice looked at each other';
- const regexpWithoutE = /\b[a-df-z]+\b/ig;
- console.log(aliceExcerpt.match(regexpWithoutE));
- // expected output: Array ["and", "at"]

- const imageDescription = 'This image has a resolution of 1440×900 pixels.';
- const regexpSize = /([0-9]+)×([0-9]+)/;
- const match = imageDescription.match(regexpSize);
- console.log(`Width: ${match[1]} / Height: ${match[2]}.`);
- // expected output: "Width: 1440 / Height: 900."

Ejemplos

- http://w3.unpocodetodo.info/utiles/regex-en-javascript.php