Docs_

Expresiones regulares

Nos permiten buscar patrones dentro de una cadena de texto.

En Javascript, las expresiones regulares empiezan y terminan por una barra:

```
/expresion/
```

La función test nos devuelve si hay coincidencia o no:

```
var expresion = new RegExp(/amor/);
const r1 = expresion.test("Busco amor");
const r2 = expresion.test("No busco nada");

console.log(r1); // true
console.log(r2); // false
```

La función match nos devuelve un array con información acerca de las ocurrencias que ha habido (si no usamos el modificador /g ,global, que veremos a continuación, sólo nos mostrará información de la primera ocurrencia).

```
const expresion = new RegExp(/amor/);
const r1 = "Busco amor, amor".match(expresion);
const r2 = "No busco nada".match(expresion);

console.log(r1); // array de un elemento
console.log(r2); // null
```

```
const expresion = new RegExp(/amor/g);
const r1 = "Busco amor".match(expresion);
const r2 = "Busco amor, amor".match(expresion);
const r3 = "No busco nada".match(expresion);

console.log(r1); // ["amor"]
console.log(r2); // ["amor", "amor"]
console.log(r3); // null
```

Utilizando el modificador /i (insensitive) podremos no tener en cuenta si el resultadode la comparación son mayúsculas o minúsculas:

```
const r1 = "Busco amor, Amor".match(/amor/g);
console.log(r1); // ["amor"]
```

```
const r1 = "Busco amor, Amor".match(/amor/gi);
console.log(r1); // ["amor", "Amor"]
```

[]

Para buscar letras o símbolos concretos, usaremos corchetes:

```
const r1 = "Busco amor, Amor".match(/[ao]/g);
console.log(r1); // ["o", "a", "o", "o"]
```

Lo anterior también es aplicable a un rango:

```
const r1 = "123456789".match(/[1-3]/g);
console.log(r1); // ["1", "2", "3"]
```

^

También podemos detectar cuando no se cumple la búsqueda con el símbolo ^:

```
const r1 = "amor".match(/[^a]/g);
console.log(r1); // ["m", "o", "r"]
```

```
const r1 = "r25".match(/[^1-3]/g);
console.log(r1); // ["rm", "5"]
```

.

Usar un punto (.) significa «cualquier caracter»:

```
const r1 = "ana".match(/./g);

console.log(r1); // ["a", "n", "a"]
```

\d

Para encontrar cualquier tipo de dígito usaremos \d:

```
const r1 = "ana".match(/.a/g);

console.log(r1); // ["na"]
```

Con \d podemos referenciar cualquier tipo de dígitos:

```
const r1 = "r2d2".match(/\d/g);

console.log(r1); // ["2", "2"]
```

\D

Y con \D invertiremos la búsqueda anterior:

```
const r1 = "r2d2".match(/\D/g);
console.log(r1); // ["r", "d"]
```

Para buscar letras que no sean símbolos ni caracters compuestos (á, ä, à…):

```
const r1 = "&%aá".match(/\w/g);

console.log(r1); // ["a"]
```

\b

Para buscar una cadena al principio de una cadena:

```
const r1 = "aabbcc".match(/\baa/g);

console.log(r1); // ["aa"]
```

Y para buscarla al final:

```
const r1 = "aabbcc".match(/cc\b/g);
console.log(r1); // ["cc"]
```

+

Para buscar ab y luego c al menos una vez:

```
const r1 = "abcoooabccccc".match(/abc+/g);

console.log(r1); // ["abc", "abccccc"]
```

*****

Mientras que el anterior símbolo de suma significaba «al menos una coincidencia» el símbolo de multiplicación significa «entre 0 o más coincidencias»:

```
const r1 = "aboooabccccc".match(/abc*/g);

console.log(r1); // ["ab", "abccccc"]
```

?

Lo mismo de antes, pero ahora entre 0 y 1 ocurrencia:

```
const r1 = "aboabcooabcc".match(/abc?/g);

console.log(r1); // ["ab", "abc", "abc"]
```

{}

Para hacer una búsqueda de un patrón una determinada cantidad de veces:

```
const r1 = "abcabc-abc-abcabc".match(/(abc){2}/g);

console.log(r1); // ["abc", "abc"]
```

Notas de la lección

a B C…	Letras
123…	dígitos
\d	Cualquier dígito
\D	Cualquier carácter que no sea un dígito
.	Cualquier personaje
\.	Período
[a B C]	Solo a, b o c
[^ abc]	Ni a, b, ni c
[Arizona]	Caracteres de la a a la z
[0-9]	Números del 0 al 9
\w	Cualquier carácter alfanumérico
\W	Cualquier carácter no alfanumérico
{metro}	m Repeticiones
{Minnesota}	m a n Repeticiones
*	Cero o más repeticiones
+	Una o más repeticiones
?	Carácter opcional
\s	Cualquier espacio en blanco
\S	Cualquier carácter que no sea un espacio en blanco
^…$	Comienza y termina
(…)	Grupo de captura
(a B C))	Subgrupo de captura
(.*)	capturar todo
(abc|def)	Coincide con abc o def

https://regexone.com/