Expresiones regulares

- Las expresiones regulares (o regex) permiten trabajar con cadenas de texto de forma eficaz de tres formas diferentes:
  - Buscar texto en un string
  - Reemplazar un string con otros
  - Extraer un string de otro
- En javascript se pueden definir las expresiones regulares de dos formas diferentes:
  - Mediante un objeto.
  - Mediante un literal.

En el ejemplo re1 y re2 son dos expresiones regulares equivalentes. El patrón de búsqueda es “hola”.
El método test devuelve un bool indicando si el patrón aparece en el string.

```
const re1 = RegExp('hola');
const re2 = /hola/;
console.log(re1.test('El mensaje es "hola mundo"')); // true
console.log(re2.test('El mensaje es "hello mundo"')); // false
```

Expresiones regulares

- Para buscar en un string que comienza con un patrón se utiliza “^”.
- Para buscar en un string que termina en un patrón se utiliza “$”.
- “.” representa un carácter cualquiera que no sea nueva línea “\n”.
- “.*” es un patrón para cero o más caracteres.
- “[a-z]” representa un carácter cualquiera dentro del rango de la “a” a la “z”.
- Otros rangos posibles: “[xyz]”, “[0-9]”, “[A-Z]”, “[o-z]” o “[A-Za-z]

```
console.log(/^hello/.test('hola mundo')); // false
console.log(/world$/.test('hola mundo')); // false
console.log(/^h.*o$/.test('hola mundo')); // true
console.log(/^[0-9]/.test('hola mundo')); // false
console.log(/[u-v]/.test('hola mundo')); // true
```

El símbolo “^” dentro de un rango niega el rango. Ejemplo: [^0-9] representa un carácter no numérico.

Expresiones regulares

- “\d” coincide con un dígito, equivalente a [0-9]
- “\D” coincide con un carácter que no sea un dígito, equivalente a [^0-9]
- “\w” coincide con un carácter alfanumérico (más guion bajo), equivalente a [A-Za-z_0-9]
- “\W” coincide con un carácter no alfanumérico, cualquier cosa excepto [^A-Za-z_0-9]
- “\s” coincide con un carácter de espacio en blanco: espacios, tabulaciones, nuevas líneas y espacios Unicode
- “\S” coincide con un carácter que no sea un espacio en blanco
- “\0” coincide con nulo
- “\n” coincide con un carácter de nueva línea
- “\t” coincide con un carácter de tabulación
- “\uXXXX” coincide con un carácter Unicode con el código XXXX
- “.” coincide con un carácter que no sea un carácter de nueva.

Expresiones regulares

- “|” representa “or” para buscar por varios patrones posibles. Ejemplo:

```
console.log(/^hello|^hola/.test('hola mundo')); // true
console.log(/^[8-9]|o$/.test('hola mundo')); // true
```

- Cuantificadores:
  - “?” representa 0 o 1 elementos.
  - “+” representa 1 o más elementos.
  - “*” representa 0 o más elementos.
  - “{n}” representa n elementos.
  - {n,m} representa entre n y m elementos
  - {n,} representa “al menos” n elementos

```
console.log(/^a\d?/.test('abc')); // true
console.log(/^a\d?/.test('ab3')); // true
console.log(/^a\d/.test('abc')); // false
console.log(/^a\d/.test('ab3')); // false
console.log(/\d{3}/.test('h34a')); // false
console.log(/\d{3}/.test('h346a')); // true
console.log(/[a-c]{3,}/.test('--cab--')); // true
```

Expresiones regulares

- Los paréntesis (…) sirven para agrupar parte un patrón y opcionalmente indicar la repetición de dicho grupo. Ejemplo:

```
console.log(/(abc){2}(.\d)/.test('--abcabcx4--')); // true
```

- El primer grupo “(abc)” se tiene que repetir dos veces.
- El segundo grupo “(.\d)” debe aparecer una sola vez (seguido del primer grupo)
- Los grupos también sirven para capturar las coincidencias. Esto se hace con las dos funciones equivalentes: **String.match(RegExp)** y **RegExp.exec(string)**

```
const result = '--abcabcx4--'.match(/(abc){2}(.\d)/);
console.log(result.length); // 3
console.log(result[0]); // abcabcx4 --> texto total coincidente
console.log(result[1]); // abc --> captura del primer grupo
console.log(result[2]); // x4 --> captura del segundo grupo
```

Expresiones regulares

- Para reemplazar parte de un string por otro se puede utilizar el método replace() de un string. Solo reemplaza la primera ocurrencia. Ejemplo:

```
const r = 'My dog is a good dog';
console.log(r.replace('dog', 'cat')); // My cat is a good dog
```

- Con una expresión regular podría ser (la “g” indica búsqueda global):

```
const r = 'My dog is a good dog';
console.log(r.replace(/dog/g, 'cat')); // My cat is a good cat
```

- Mas ejemplos:

```
const r1 = 'My dog is a good. My cat is good';
console.log(r1.replace(/(dog|cat)/g, 'pet')); // My pet is good. My pet is good
const r2 = 'AAAsssAAssxAkkkkAAAAAA';
console.log(r2.replace(/A{2,}/g, 'B')); // BsssBssxAkkkkB
```