EXPRESIONES REGULARES

Las expresiones regulares describen un conjunto de elementos que siguen un patrón. Dicho de otra forma, es una regla que identifica a una serie de elementos que tiene algo en común. Un ejemplo de expresión regular podrían ser todas las palabras que comienzan por la letra "a" minúscula. La expresión regular para este patrón debe asegurarse de que la palabra comienza por "a" minúscula, el resto de palabras que precedan a la cadena podrán ser cualquier carácter.

La mayoría de los lenguajes de programación incluyen la implementación de expresiones regulares. En el caso que nos atañe, JavaScript implementa una configuración para implementar expresiones regulares y así facilitar las comprobaciones de ciertos datos que deben seguir una estructura concreta. Este tipo de expresiones es muy útil a la hora de evaluar algunos tipos de datos que normalmente utilizamos en los formularios.

CARACTERES ESPECIALES DE LAS EXPRESIONES REGULARES

A continuación vamos describir algunos de los elementos que utiliza JavaScript para crear expresiones regulares.

- **^ Principio de entrada o línea**. Este carácter indica que las cadenas deberán comenzar por el siguiente carácter. Si éste fuera una "a" minúscula, la expresión regular seria sería, ^a.
- **$ Fin de entrada o línea**. Indica que la cadena debe terminar por el elemento precedido al dólar.
- * **El carácter anterior 0 o más veces**. Indica que el carácter anterior se puede repetir en la cadena 0 o más veces.
- **+ El carácter anterior 1 o más veces.** El símbolo más indica que el carácter anterior se puede repetir en la cadena una o más veces.
- **? El carácter anterior una vez como máximo**. El símbolo interrogación indica que el carácter anterior se puede repetir en la cadena cero o una vez.
- **. Cualquier carácter individual**. El símbolo punto indica que puede haber cualquier carácter individual salvo el de salto de línea.
- **xIy x o y**. La barra vertical indica que puede ser el carácter x o el y.
- **{n} n veces el carácter anterior**. El carácter anterior a las llaves tiene que aparecer exactamente n veces.
- **{n,m} Entre n y m veces el carácter anterior.** El carácter anterior a las llaves tiene que aparecer como mínimo n y como máximo m veces.
- **[abc] Cualquier carácter de los corchetes.** En la cadena puede aparecer cualquier carácter que esté incluido en los corchetes. Además, podemos especificar rangos de caracteres que sigan un orden. Si se especifica el rango [a-z] equivaldría a incluir todas las letras minúsculas del abecedario.
- **[^abc] Un carácter que no esté en los corchetes**. En la cadena pueden aparecer todos los caracteres que no estén incluidos en los corchetes. También podemos especificar rangos de caracteres como en el punto anterior.
- **\b Fin de palabra**. El símbolo \b indica que tiene que haber un fin de palabra o retorno de carro.

- **\B No fin de palabra**. El símbolo \B indica cualquiera que no sea un límite de palabra.
- **\d Cualquier carácter dígito.** El símbolo \d indica que puede haber cualquier carácter numérico, de 0 a 9.
- **\D Carácter que no es dígito.** El símbolo \D indica que puede haber cualquier carácter siempre que no sea numérico.
- **\f Salto de página.** El símbolo \f indica que tiene que haber un salto de página.
- **\n Salto de línea.** El símbolo \n indica que tiene que haber un salto de línea.
- **\r Retorno de carro.** El símbolo \r indica que tiene que haber un retorno de carro.
- **\s Cualquier espacio en blanco.** El símbolo \s indica que tiene que haber un carácter individual de espacio en blanco: espacios, tabulaciones, saltos de página o saltos de línea.
- **\S Carácter que no sea blanco.** El símbolo \S indica que tiene que haber cualquier carácter individual que no sea un espacio en blanco.
- **\t Tabulación**. El símbolo \ t indica que tiene que haber cualquier tabulación.
- **\w Carácter alfanumérico**. El símbolo \w indica que puede haber cualquier carácter alfanumérico, incluido el de subrayado.
- **\W Carácter que no sea alfanumérico.** El símbolo \W indica que puede haber cualquier carácter que no sea alfanumérico.

VALIDAR UN FORMULARIO CON EXPRESIONES REGULARES

Con la combinación de estas expresiones podemos abordar infinidad de patrones para validar datos en la mayoría de los formularios. Combinando cada una de las condiciones obtenemos patrones como pueden ser, una dirección de correo electrónico, un teléfono, un código postal, un DNI, etc. A continuación, vamos a ver cómo realizar algunas de las configuraciones de expresiones regulares más utilizadas para validar datos de un formulario.

VALIDAR UNA DIRECCIÓN DE CORREO ELECTRÓNICO.

En el caso de que el usuario de la aplicación web introduzca una dirección de correo electrónico, es casi imprescindible que esta sea correcta. Utilizando expresiones regulares vamos a comprobar que la estructura de la dirección sea correcta. En primer lugar, comprobaremos que comienza por una cadena de texto, que sigue por una @ y que termina por otra cadena de texto un punto y otra cadena de texto. De esta forma aseguraremos que, al menos, la estructura de la dirección es correcta. Combinando las funciones con las expresiones regulares, obtenemos un código que valida una dirección de correo electrónico.

```
<script type="text/javascript">
        function validaEmail()
                valor = document.getElementById("campo").value;
                if (!(/\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)/.test(valor))){
                        return false;
                }else{
                        return true;
                }
```

```
</script>
```

VALIDAR UN DNI

```
<script type="text/javascript">
	function validaDNI(){
		valor=document.getElementById("dni").value;
		var letras=['T','R','W','A','G','M','Y','F','P','D','X','B','N','J','Z','S','Q','V','H','L','C','K','E','T'];
		if (!(/^\d{8}[A-Z]$/.test (valor))){
			return false;
		}
		if (valor.charAt(8)!=letras[(valor.substring(0,8))%23]){
			return false;
		}
		return true;
	}
</script>
```

En el código anterior validamos un DNI en varios pasos. Primero recogemos en la variable valor el campo del DNI que ha introducido el usuario. Generamos un array con las letras válidas para el DNI. En la primera condición comprobamos a través de expresiones regulares que el patrón del campo introducido tiene una longitud de 8 números y una letra. Si no es así devuelve falso la función.

En la última condición realizamos la división entre 23 y tomamos el resto. Comprobamos que la posición del array letras, perteneciente al resto de la operación, se corresponde con el valor de la letra del DNI introducido. En caso de que sea igual devuelve verdadero (al final de la función). Si no es igual la letra del valor introducido a la calculada, entra en la condición y devuelve falso.

VALIDAR UN NÚMERO DE TELÉFONO

Las expresiones regulares son un recurso muy útil para validar un teléfono. A continuación vamos a ver cómo validaremos un número de teléfono:

-

```
function validaTelefono(){
	valor = document.getElementById("telefono").value;
	if (!(/^\d{9}$/test(valor))){
		return false;
	}
	return true;
}
```

En el código anterior estamos validando que el número que introdujo el usuario este formado por dígitos y tenga una longitud de 9. A la hora de validar un teléfono podemos ser más estrictos. Si queremos que el campo solo corresponda a un número de teléfono móvil, podemos especificar que el número solo pueda comenzar por 6, puesto que aquí los teléfonos móviles comienzan todos por 6. La expresión regular integrada en la condición anterior para comprobar esto sería la siguiente:

```
if (!(/^[6]\d{8}$/.test(valor))){
```

En el caso de que queramos validar un número de teléfono fijo, el código sería el que mostramos a continuación:

If (!(/^[89]\d{8}$/.test(valor))){

En el código los números solo pueden comenzar por 8 o por 9, que son los números actuales válidos aquí.

Existen otros muchos datos que podemos validar con expresiones regulares, como pueden ser números de cuentas bancarias, CIF de empresas y cualquier dato que siga un patrón determinado. Validar los datos para que el usuario no envíe el formulario si estos no son correctos es algo que aumenta el rendimiento del servidor, al disminuir las peticiones. También es importante mostrar una información amplia de como el usuario tiene que introducir los datos que se validen, de lo contrario, si el usuario no conoce cómo debe ingresar el dato puede quedarse bloqueado ante la aplicación.