Programación Hipermedia I

Práctica 6: Expresiones regulares y el Modelo de Objetos de Documento

1. Objetivos

- Aprender a manejar el DOM de una página web para manipular su contenido.
- Aprender a validar un formulario con el lenguaje de programación JavaScript y el empleo de expresiones regulares.

2. Recursos

¿Qué son las expresiones regulares? ¿Cómo se emplean con JavaScript?

- **W3Schools**[1]: cursos de aprendizaje y guías de referencia de diversas tecnologías empleadas en la programación web. Incluye un tutorial y temas avanzados sobre JavaScript, expresiones regulares con JavaScript y DOM.
- **Expresión regular**[2]: definición de las expresiones regulares según la Wikipedia.
- **JavaScript Regular Expressions**[3]: resumen de la sintaxis de las expresiones regulares en JavaScript.
- **Programmer's Guide to Regular Expressions**[4]: tutorial sobre las expresiones regulares en JavaScript.
- **JavaScript RegExp Example: Regular Expression Tester**[5]: permite comprobar el funcionamiento de una expresión regular en JavaScript sin tener que programar.
- **Regular Expressions Cheat Sheet**[6]: resumen en una sola cara de lo más importante de las expresiones regulares.

¿Qué es el DOM?

- **W3C Documento Object Model (DOM)**[7]: especificación oficial del DOM según el W3C.
- **Mozilla Developer Center Gecko DOM Reference**[8]: guía de referencia de los objetos del DOM disponible en el motor Gecko empleado por varios navegadores web como Mozilla Firefox.
- **Support for DOM2 Core and XML modules in Opera**[9] y **DOM 2 HTML Objects supported in Opera**[10]: objetos, propiedades y métodos del DOM que admite el navegador Opera.

¿Cómo puedo explorar el DOM de una página web?

- **DOM Inspector**[11]: permite explorar y editar el DOM de una página web.

[1] http://www.w3schools.com
[2] http://es.wikipedia.org/wiki/Expresión_regular
[3] http://www.visibone.com/regular-expressions/
[4] http://www.javascriptkit.com/javatutors/redev.shtml
[5] http://www.regular-expressions.info/javascriptexample.html
[6] http://www.addedbytes.com/cheat-sheets/regular-expressions-cheat-sheet/
[7] http://www.w3.org/DOM/
[8] http://developer.mozilla.org/en/Gecko_DOM_Reference
[9] http://www.opera.com/docs/specs/js/ecma/dom/index.dml
[10] http://www.opera.com/docs/specs/js/ecma/dom/html/index.dml
[11] http://developer.mozilla.org/en/DOM_Inspector

3. ¿Qué tengo que hacer?

En esta práctica tienes que emplear las expresiones regulares de JavaScript para validar el formulario de registro como nuevo usuario que has realizado en las prácticas anteriores (sustituye el código de validación implementado en la práctica anterior). En concreto, tienes que programar las siguientes restricciones:

Página con el formulario de registro como nuevo usuario Contiene un formulario con los datos necesarios para registrarse (nombre de usuario, contraseña, repetir contraseña, dirección de email, sexo, fecha de nacimiento, ciudad y país de residencia, foto). Antes de enviar el formulario, debes realizar las siguientes comprobaciones:

- **nombre de usuario**: sólo puede contener letras del alfabeto inglés (en mayúsculas y minúsculas) y números; longitud mínima 3 caracteres y máxima 15.
- **contraseña**: sólo puede contener letras del alfabeto inglés (en mayúsculas y minúsculas), números y el subrayado; al menos debe contener una letra en mayúsculas, una letra en minúsculas y un número; longitud mínima 6 caracteres y máxima 15.
- **repetir contraseña**: su valor debe coincidir con el escrito en el campo **contraseña**.
- **dirección de email**: no puede estar vacío, hay que comprobar que cumple el patrón de una dirección de email (compuesto de cualquier carácter alfanumérico del alfabeto inglés, el subrayado, el guión y el punto; no permitir dominios principales de menos de 2 caracteres y más de 4 caracteres).
- **sexo**: se debe elegir un valor.
- **fecha de nacimiento**: comprobar que es una fecha válida.
- El resto de campos no indicados se pueden quedar vacíos.

Importante: no siempre una expresión regular es la mejor solución. Debes de decidir en qué casos es mejor realizar una validación con una expresión regular y en qué casos es mejor realizarla "a mano".

Además, tienes que permitir que el usuario pueda ordenar, de forma ascendente y descendente, el listado resultado de una búsqueda de fotos por los campos título, fecha y país. El proceso de ordenación se tiene que realizar en la misma página mediante JavaScript, sin recargarla.

4. ¿Cómo lo hago?

4.1. Expresiones regulares

Una expresión regular es un patrón que define un conjunto de cadenas sin enumerar todos sus elementos. Una expresión regular está formada de caracteres y metacaracteres que tienen una función definida. La principal utilidad de las expresiones regulares es la de describir un conjunto de cadenas, lo que resulta de utilidad en editores de texto y aplicaciones para buscar y manipular texto. Además, otro uso es la validación de un formato específico en una cadena de caracteres dada, como por ejemplo fechas, correos electrónicos o identificadores.

En JavaScript existen dos formas de definir una expresión regular: mediante una cadena literal y mediante el constructor de objeto `RegExp`. Una expresión regular se puede aplicar a diferentes cadenas mediante `expReg.test(cadena)`, que devuelve `true` si `cadena` cumple `expReg` y `false` en caso contrario.

Por ejemplo, en el siguiente código se comprueba si el usuario ha escrito correctamente una matrícula de automóvil que debe seguir el patrón *código del país* (1 o 2 letras), un espacio en blanco, *numeración* (4 dígitos), un espacio en blanco y *letras* (3 letras, empezando en BBB y acabando en ZZZ, sin las vocales):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Validación con expresión regular</title>
<script type="text/javascript">
function literal() {
```

```
  var m = document.getElementById("matricula").value;
  var expreg = /^[A-Z]{1,2}\s\d{4}\s([B-D]|[F-H]|[J-N]|[P-T]|[V-Z]){3}$/;

  if(expreg.test(m))
    alert("La matrícula es correcta");
  else
    alert("La matrícula NO es correcta");
}

function objeto() {
  var m = document.getElementById("matricula").value;
  var expreg = new RegExp("^[A-Z]{1,2}\\s\\d{4}\\s([B-D]|[F-H]|[J-N]|[P-T]|[V-Z]){3}$");

  if(expreg.test(m))
    alert("La matrícula es correcta");
  else
    alert("La matrícula NO es correcta");
}
</script>
</head>
<body>
<form id="miForm" action="" method="get">
<p>
Matrícula: <input type="text" id="matricula" />
<br />
<input type="button" value="Literal" onclick="literal()" />
<input type="button" value="Objeto" onclick="objeto()" />
</p>
</form>
</body>
</html>
```

Al usar la expresión regular con el objeto `RegExp`, como se escribe dentro de una cadena es necesario escribir dos barras invertidas `\\` para representar una barra invertida, ya que el carácter barra invertida se emplea para *escapar* los caracteres especiales. Cuando se emplea mediante una cadena literal no se debe escapar la barra invertida.

¿Qué significa cada elemento de la expresión regular de este ejemplo?

- `^`: el emparejamiento se debe realizar desde el principio de la cadena.
- `[A-Z]`: cualquier carácter entre la A mayúscula y la Z mayúscula.
- `{1,2}`: uno o dos caracteres.
- `\s`: un espacio en blanco.
- `\d`: un dígito.
- `{4}`: cuatro dígitos.
- `\s`: un espacio en blanco.
- `([B-D]|[F-H]|[J-N]|[P-T]|[V-Z])`: cualquier carácter entre la B mayúscula y la Z mayúscula, excepto las vocales.
- `{3}`: tres caracteres.
- `$`: el emparejamiento se debe realizar hasta el final de la cadena.

Con esta expresión regular podemos tener matrículas desde “A 1111 BBB” hasta “ZZ 9999 ZZZ”.

4.2. DOM

El DOM (*Document Object Model*) es una API destinada a trabajar con documentos HTML y XML. El DOM está desarrollado por el W3C.

El DOM proporciona una representación en forma de árbol de un documento y permite su manipulación (añadir un elemento nuevo, borrar un elemento, mover un elemento de sitio). Algunas funciones útiles del DOM:

- `document.getElementById(id)`: devuelve el elemento con el identificador indicado. **¡Cuidado!**: recuerda que JavaScript es un lenguaje sensible a mayúsculas y minúsculas, `Id` en `getElementById(id)` se tiene que escribir con la letra "I" en mayúsculas y la letra "d" en minúsculas, si se escribe de otra forma, como por ejemplo "ID", fallará.
- `elementoPadre.getElementsByTagName(etiqueta)`: devuelve un array de elementos con la etiqueta indicada, los elementos se buscan a partir de `elementoPadre` (si `elementoPadre` es `document`, la búsqueda se realiza en toda la página web).
- `document.createElement(etiqueta)`: crea un nuevo elemento según la etiqueta indicada.
- `elementoPadre.appendChild(elementoNuevo)`: añade `elementoNuevo` a `elementoPadre` al final de su lista de elementos hijo.
- `elementoPadre.insertBefore(elementoNuevo, elementoReferencia)`: añade `elementoNuevo` en la lista de elementos hijo de `elementoPadre` antes de `elementoReferencia`.

El siguiente ejemplo muestra cómo se pueden añadir párrafos de texto nuevos a una página web con el texto que introduzca el usuario:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Validación con expresión regular</title>
<script type="text/javascript">
function nuevoTexto() {
  // Obtiene el div que tiene id="parrafos"
  var d = document.getElementById("parrafos");

  // Crea un nuevo elemento de tipo párrafo
  var np = document.createElement("p");

  // Inserta el texto en el contenido del nuevo párrafo
  // textContent es un atributo (propiedad) de un nodo
  // que representa el contenido textual
  np.textContent = document.getElementById("texto").value;

  // Añade el párrafo al final de los elementos que contenga el div
  d.appendChild(np);

  return false;
}
</script>
</head>
<body>
<form id="miForm" action="" method="get" onsubmit="return nuevoTexto()" >
<p>
Texto: <input type="text" id="texto" />
<br />
<input type="submit" value="Añade texto" />
```

```
</p>
</form>
<div id="parrafos">
<!-- Inicialmente vacío -->
</div>
</body>
</html>
```

Con HTML5 han aparecido nuevas funciones para manipular el DOM, pero se tienen que usar con mucho cuidado para que no se produzcan problemas de compatibilidad con navegadores web antiguos. De las nuevas funciones, las más interesantes son:

- `elementoPadre.getElementsByClassName(nombres)`: devuelve un array de elementos que poseen los nombres de clases indicados, los elementos se buscan a partir de `elementoPadre` (si `elementoPadre` es `document`, la búsqueda se realiza en toda la página web). Este método está definido en **W3C DOM4**[12].
- `elementoPadre.querySelector(selector)`: devuelve el primer elemento que cumple el selector indicado, el elemento se busca a partir de `elementoPadre`. Este método está definido en **W3C Selectors API Level 1**[13].
- `elementoPadre.querySelectorAll(selector)`: devuelve un array de elementos que cumplen el selector indicado, los elementos se buscan a partir de `elementoPadre`. Este método está definido en **W3C Selectors API Level 1**.

5. Recomendaciones

Las expresiones regulares son un mecanismo de programación muy potentes, pero esa potencia origina que sean complejas de utilizar. Comienza a trabajar con expresiones regulares sencillas y poco a poco intenta escribir expresiones más complejas.

Para ordenar los elementos de una lista puedes emplear diferentes algoritmos de ordenación. En la página **The Art of Web: JavaScript**[14] puedes ver la implementación de los más conocidos: *bubble sort*, *insertion sort*, *shell sort* y *quick sort*.

Puedes usar las nuevas funciones para manipular el DOM, pero para ello tienes que:

- Detectar que las funciones están disponibles y las puedes usar.
- Proporcionar una alternativa si las funciones no están disponibles en el navegador.

¿Cómo se puede hacer?

[12] http://www.w3.org/TR/dom/
[13] http://www.w3.org/TR/selectors-api/
[14] http://www.the-art-of-web.com/javascript/