Desarrollo de Aplicaciones Web

Práctica optativa JavaScript

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.
- Aprender a seleccionar un estilo alternativo mediante el DOM y JavaScript.
- Conocer el concepto de *cookie* y sus posibles usos.
- Aprender a utilizar las *cookies* con JavaScript.

2. Recursos

¿Qué es el DOM?

- **W3C Documento Object Model (DOM)**[1]: especificación oficial del DOM según el W3C.
- **Mozilla Developer Center Gecko DOM Reference**[2]: 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**[3] y **DOM 2 HTML Objects supported in Opera**[4]: objetos, propiedades y métodos del DOM que admite el navegador Opera.

¿Cómo puedo explorar el DOM de una página web?

- **DOM Inspector**[5]: permite explorar y editar el DOM de una página web.

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

- **W3Schools**[6]: 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**[7]: definición de las expresiones regulares según la Wikipedia.
- **JavaScript Regular Expressions**[8]: resumen de la sintaxis de las expresiones regulares en JavaScript.
- **Programmer's Guide to Regular Expressions**[9]: tutorial sobre las expresiones regulares en JavaScript.
- **JavaScript RegExp Example: Regular Expression Tester**[10]: permite comprobar el funcionamiento de una expresión regular en JavaScript sin tener que programar.

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

- **Regular Expressions Cheat Sheet**[11]: resumen en una sola cara de lo más importante de las expresiones regulares.

¿Qué son las cookies?

- **RFC 2965 HTTP State Management Mechanism**[12]: documento oficial que especifica el uso de las cookies para lograr una comunicación con HTTP que conserve el estado.
- **Cookie**[13]: definición en la Wikipedia de cookie, explica qué son, cómo se usan, algunos inconvenientes y alternativas a su uso.
- **Cookie Central**[14]: explica qué son las cookies y contiene ejemplos de uso.
- **Todo sobre la cookies**[15]: explica qué son las cookies y contiene ejemplos de uso en JavaScript.

¿Cómo se emplean las cookies con JavaScript?

- **JavaScript Cookies**[16]: cómo utilizar las cookies en JavaScript.
- **Las cookies**[17]: explicación en español del uso de las cookies en JavaScript.

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 (el resto de restricciones ya programadas las debes conservar):

Página principal Contiene un formulario (nombre de usuario y contraseña) para acceder como usuario registrado. Antes de enviar el formulario, debes comprobar que el usuario ha escrito algo en ambos campos, pero evita que el usuario escriba únicamente espacios en blanco o tabuladores.

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; no puede comenzar con un número; longitud mínima 3 caracteres y máxima 15.
- **contraseña**: sólo puede contener letras del alfabeto inglés (en mayúsculas y minúsculas), números, el guion y el guion bajo (subrayado); al menos debe contener una letra en mayúscula, una letra en minúscula y un número; longitud mínima 6 caracteres y máxima 15.
- **dirección de email**: no puede estar vacío, hay que comprobar que cumple el siguiente patrón de una dirección de email (es una simplificación del estándar):
  - El formato de un correo electrónico es `parte-local@dominio`.
  - La longitud máxima de `parte-local` es 64 caracteres. La longitud máxima de `dominio` es 255 caracteres. Cada una de las partes tiene una longitud mínima de 1 carácter.
  - La `parte-local` es una combinación de las letras mayúsculas y minúsculas del alfabeto inglés, los dígitos del 0 al 9, los caracteres imprimibles `!#$%&'*+-/=?^_`{|}~` y el punto. El punto no puede aparecer ni al principio ni al final y tampoco pueden aparecer dos o más puntos seguidos.
  - El `dominio` es una secuencia de uno o más `subdominio` separados por un punto.

[11] http://www.addedbytes.com/cheat-sheets/regular-expressions-cheat-sheet/
[12] http://tools.ietf.org/html/rfc2965
[13] http://es.wikipedia.org/wiki/Cookie
[14] http://www.cookiecentral.com/c_concept.htm
[15] http://www.iec.csic.es/criptonomicon/cookies/default.html
[16] http://www.w3schools.com/js/js_cookies.asp
[17] http://www.mailxmail.com/curso/informatica/javascript/capitulo21.htm

- La longitud máxima de `subdominio` es 63 caracteres.
- Un `subdominio` es una combinación de las letras mayúsculas y minúsculas del alfabeto inglés, los dígitos del 0 al 9 y el guion. El guion no puede aparecer ni al principio ni al final.
- La longitud máxima de una dirección de correo electrónico es 254 caracteres[18].

Algunas de las validaciones se pueden implementar con HTML[19] y con el Constraint validation API[20]. Por ejemplo, se puede emplear el atributo `pattern` para definir una expresión regular, se puede emplear `type="email"` para comprobar que un correo electrónico es válido, se puede emplear `type="date"` para forzar al usuario a introducir una fecha válida, etc. Sin embargo, el objetivo de esta práctica es aprender a utilizar JavaScript y las expresiones regulares, por lo que **NO SE DEBE HACER USO DE ESAS FACILIDADES QUE PROPORCIONA HTML EN ESTA PRÁCTICA**.

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, autor y país. El proceso de ordenación se tiene que realizar en la misma página mediante JavaScript, sin recargarla. Pon suficientes datos de ejemplo en la página resultado de una búsqueda de fotos para que se pueda verificar el correcto funcionamiento de la solución implementada al ordenar mediante los diferentes criterios.

Importante: los cambios sobre la tabla para realizar la ordenación se tienen que realizar con las funciones del DOM (*Document Object Model*) que permiten añadir contenido a una página web nodo a nodo. Es decir, no vale añadir el contenido como una cadena mediante `innerHTML` u otro mecanismo similar.

Como ejemplo de lo que se pide realizar, en las figuras 1, 2 y 3 se muestran tres ejemplos de ordenación de los resultados en Amazon, FNAC y MediaMarkt. En los tres casos, la ordenación se realiza a través de una lista desplegable en la que aparecen las siguientes opciones para indicar la ordenación de forma ascendente o descendente:

- "Precio: de más bajo a más alto" y "Precio: de más alto a más bajo".
- "Más baratos primero" y "Más caros primero".
- "Título: A ->Z" y "Título: Z ->A".
- "Precio: de más caro a más barato" y "Precio: de más barato a más caro".

En la segunda práctica de CSS realizaste unos estilos alternativos para tu sitio web. Algunos navegadores web, por ejemplo Mozilla Firefox, permiten a través de su menú seleccionar el estilo alternativo, mientras que otros navegadores no lo permiten. Además, la mayoría de los usuarios desconocen que existe la posibilidad de cambiar el estilo de una página web (no saben que existe esa opción en el navegador web). Por eso, en esta práctica debes permitir que el usuario seleccione el estilo alternativo que desea desde la propia página web para que esta opción esté disponible en cualquier navegador y sea claramente visible para el usuario. Puedes realizarlo mediante una lista desplegable, un botón que vaya cambiando entre los diferentes estilos u otra solución similar.

Por ejemplo, en la Figura 4 se muestra la selección de estilos alternativos en el buscador DuckDuckGo y en la Figura 5 en Twitter. En el caso de esta práctica, **la selección se puede resolver con una simple lista desplegable que muestre todos los estilos disponibles**.

Por último, tienes que emplear las cookies para conservar el estilo seleccionado entre diferentes páginas e incluso entre diferentes visitas al sitio web en diferentes días. El tiempo de duración de la cookie debe ser 45 días.

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.

[18] Evidentemente, no tiene sentido que se diga que la longitud máxima de dominio es 255 caracteres, pero la longitud total es 254 caracteres. Esta incongruencia se debe a un error que existe en el estándar que impide que se cumplan otros estándares relacionados. La explicación está en RFC Errata to 3696: `https://www.rfc-editor.org/errata/eid1690`

[19] `https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation`

[20] `https://developer.mozilla.org/en-US/docs/Web/API/Constraint_validation`

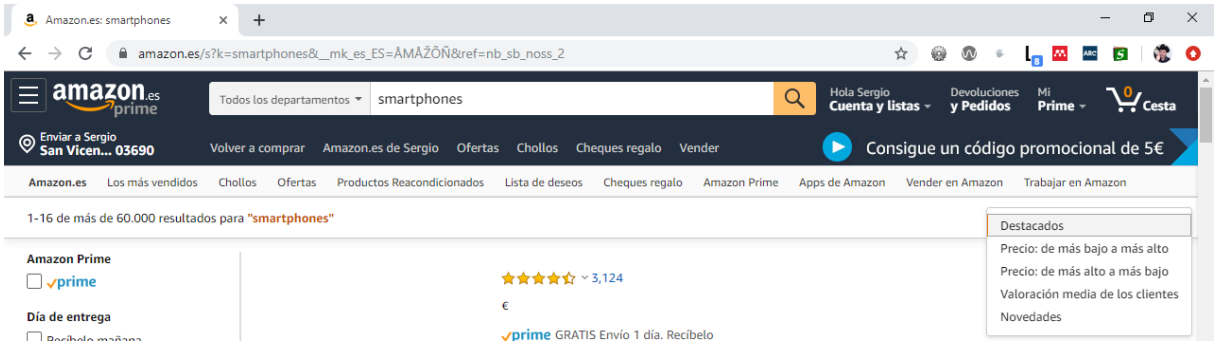

Figura 1: Amazon

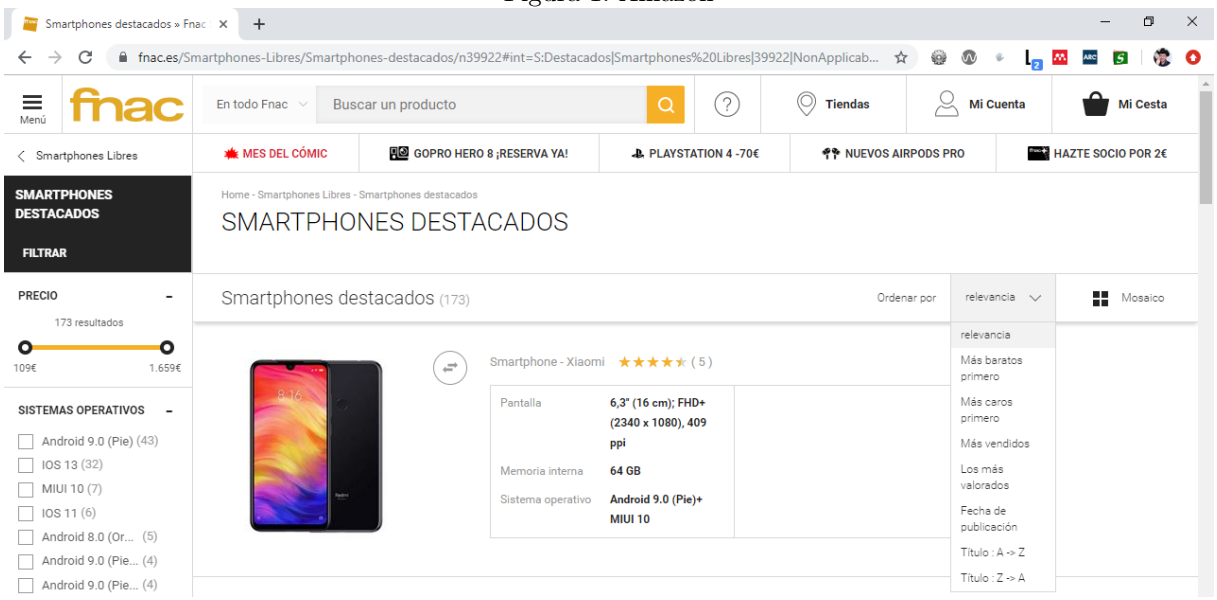

Figura 2: FNAC

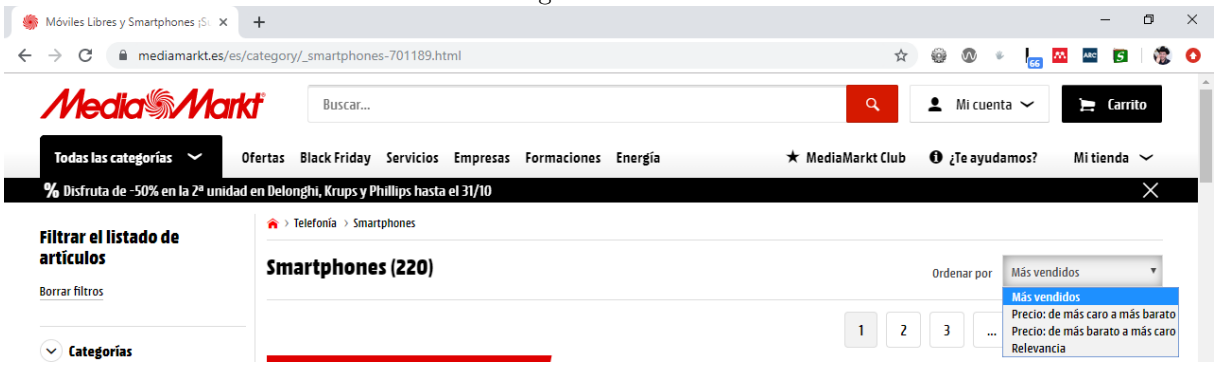

Figura 3: MediaMarkt

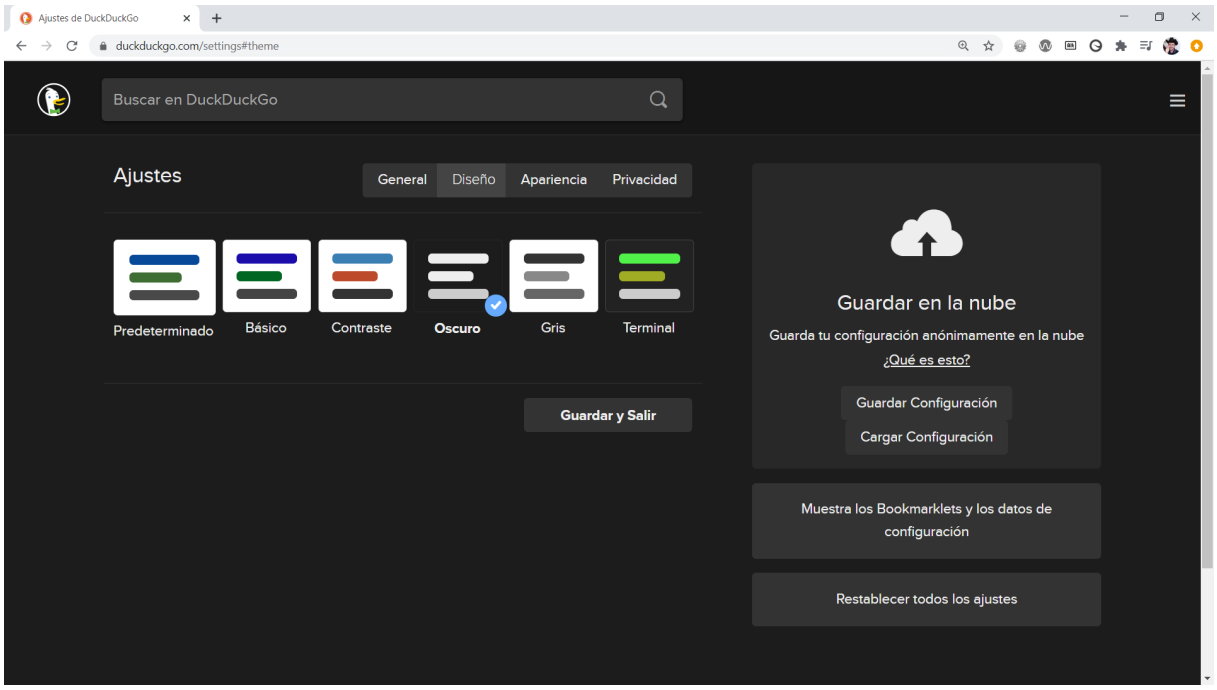

Figura 4: Selección de estilos alternativos en DuckDuckGo

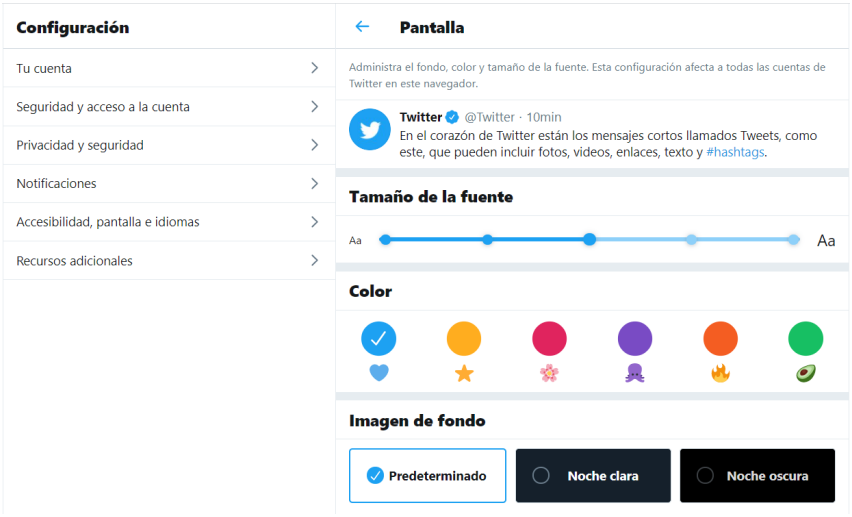

Figura 5: Selección de estilos alternativos en Twitter

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):

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Validación con expresión regular</title>
<script>
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. Selección de un estilo alternativo

A través del DOM podemos acceder al atributo `disabled` de la etiqueta `<link/>` que permite desactivar[21] (valor `true`) o activar (valor `false`) una hoja de estilo enlazada mediante la etiqueta `<link/>`.

Una primera propuesta de código JavaScript para realizar el cambio podría ser lo siguiente, donde la función `estilo()` recibe un parámetro que indica el estilo que se quiere activar (y, por tanto, todos los demás se tienen que desactivar):

```
<script>
function estilo(titulo) {
  var arrayLink = document.getElementsByTagName('link');

  for(var i = 0; i < arrayLink.length; i++) {
    if(arrayLink[i].getAttribute('title') == titulo)
      arrayLink[i].disabled = false;
    else
    arrayLink[i].disabled = true;
  }
}
</script>
```

Sin embargo, este código puede fallar ya que la etiqueta `<link/>` tiene otros usos además de emplearse para enlazar una hoja de estilo. Además, ¿qué pasa con el estilo predeterminado que siempre se tiene que visualizar aunque se elija otro estilo alternativo? Tal como está escrito el código, el estilo predeterminado se desactivará la primera vez que se cambie de estilo alternativo.

La siguiente página es un ejemplo completo con una función de selección de estilo más compleja que contempla los problemas comentados anteriormente. En este ejemplo, un usuario puede seleccionar un estilo entre dos posibles estilos, además existe un estilo predeterminado y un estilo para sólo impresión:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Página con varios estilos</title>
<link rel="stylesheet" type="text/css" href="predeterminado.css" media="screen" />
<link rel="stylesheet" type="text/css" href="principal.css" media="screen"
title="Principal" />
<link rel="alternate stylesheet" type="text/css" href="secundario.css" media="screen"
title="Secundario" />
<link rel="stylesheet" type="text/css" href="impresion.css" media="print" />
<script>
function estilo(titulo) {
  var arrayLink = document.getElementsByTagName('link');

  for(var i = 0; i < arrayLink.length; i++) {
    // Sólo aquellas etiquetas link que hacen referencia a un estilo
    // y que no sea para impresión
```

[21] Hay que tener cuidado con la propiedad `disabled` (desactivado) porque va al revés de lo que se podría pensar: para activar la etiqueta hay que poner su valor a `false`, y para desactivarla a `true`.

```
    if(arrayLink[i].getAttribute('rel') != null &&
      arrayLink[i].getAttribute('rel').indexOf('stylesheet') != -1 &&
      arrayLink[i].getAttribute('media') != 'print') {
      // Si tiene título es un estilo preferido o alternativo,
      // si no tiene título es un estilo
      // predeterminado y siempre tiene que utilizarse
      if(arrayLink[i].getAttribute('title') != null &&
        arrayLink[i].getAttribute('title').length > 0) {
        if(arrayLink[i].getAttribute('title') == titulo)
          arrayLink[i].disabled = false;
        else
          arrayLink[i].disabled = true;
      }
    }
  }
}
</script>
</head>
<body>
<ul>
<!-- Esto está mal, así no se tienen que hacer los botones -->
<li><a href="javascript:estilo('Principal')">Principal</a></li>
<li><a href="javascript:estilo('Secundario')">Secundario</a></li>
</ul>
<p>
Una página web sencilla.
</p>
</body>
</html>
```

A continuación se muestra el código de la hoja de estilo predeterminada y de las dos hojas de estilo alternativas.

El código de la hoja de estilo predeterminada, almacenada en el fichero `predeterminado.css` es:

```
/* Estilo predeterminado */
html {
  font-size: 30px;
}

p {
  border: 5px solid #FF0000;
}

ul {
  list-style-type: none;
}

li {
  display: inline;
}
```

El código de la hoja de estilo principal, almacenada en el fichero `principal.css` es:

```
/* Estilo principal */
html {
  background-color: blue;
  color: white;
  font-family: Arial;
}

ul {
  background-color: white;
  color: red;
}
```

```
a {
  color: red;
}
```

El código de la hoja de estilo secundaria, almacenada en el fichero `secundario.css`:

```
/* Estilo secundario */
html {
  background: black;
  color: yellow;
}

a {
  color: green;
  font-weight: bolder;
}

p {
  font-size: 15px;
}
```

En la Figura 6 se muestra cómo se ve la página web con el estilo principal y en la Figura 7 se muestra con el estilo secundario.

En la página "How to Use JavaScript to Change a Cascading Style Sheet (CSS) Dynamically"[22] se muestra una implementación alternativa, pero similar, al cambio de estilo de forma dinámica mediante JavaScript.

Un problema que es difícil de controlar es el llamado "Flash of unstyled content"[23] que consiste en que una página web aparece brevemente con el estilo predeterminado por el navegador o por la página web antes de cargar una hoja de estilo alternativa, debido a que el motor del navegador muestra la página web antes de aplicar el estilo alternativo. Este problema tiene difícil solución y no te debes preocupar por él al realizar esta práctica.

4.3. Cookies

Una cookie es un fragmento de información que un navegador web almacena en el disco duro del visitante a una página web. La información se almacena a petición del servidor web, ya sea directamente desde la propia página web con JavaScript o desde el servidor web mediante las cabeceras HTTP, que pueden ser generadas desde un lenguaje de web scripting como PHP. La información almacenada en una cookie puede ser recuperada por el servidor web en posteriores visitas a la misma página web.

Las cookies resuelven un grave problema del protocolo HTTP: al ser un protocolo de comunicación "sin estado" (*stateless*), no es capaz de mantener información persistente entre diferentes peticiones. Gracias a las cookies se puede compartir información entre distintas páginas de un sitio web o incluso en la misma página web pero en diferentes instantes de tiempo.

En JavaScript, se puede acceder a las cookies a través de la propiedad cookie del objeto `document`. Esta propiedad permite acceder a todas las cookies de una página web, pero el acceso no es directo, ya que la propiedad `cookie` devuelve una única cadena que contiene todas las cookies de la página. Para acceder a una cookie concreta, es necesario analizar la cadena para localizar su valor.

La siguiente página de ejemplo contiene las funciones `setCookie()` y `getCookie()` que facilitan el acceso a las cookies en JavaScript. La función `setCookie()` recibe tres parámetros: el nombre de la cookie (`c_name`), el valor de la cookie (`value`) y la fecha de caducidad como el número de días a partir de la fecha actual (`expiredays`). La función `getCookie()` recibe sólo un parámetro, el nombre de la cookie que se quiere recuperar (`c_name`). En este ejemplo se emplea un cookie para almacenar la fecha de la última visita a la página web; la cookie tiene una validez de un año (365 días).

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Ejemplo de uso de cookies</title>
<script>
```

[22] https://www.thesitewizard.com/javascripts/change-style-sheets.shtml
[23] https://en.wikipedia.org/wiki/Flash_of_unstyled_content

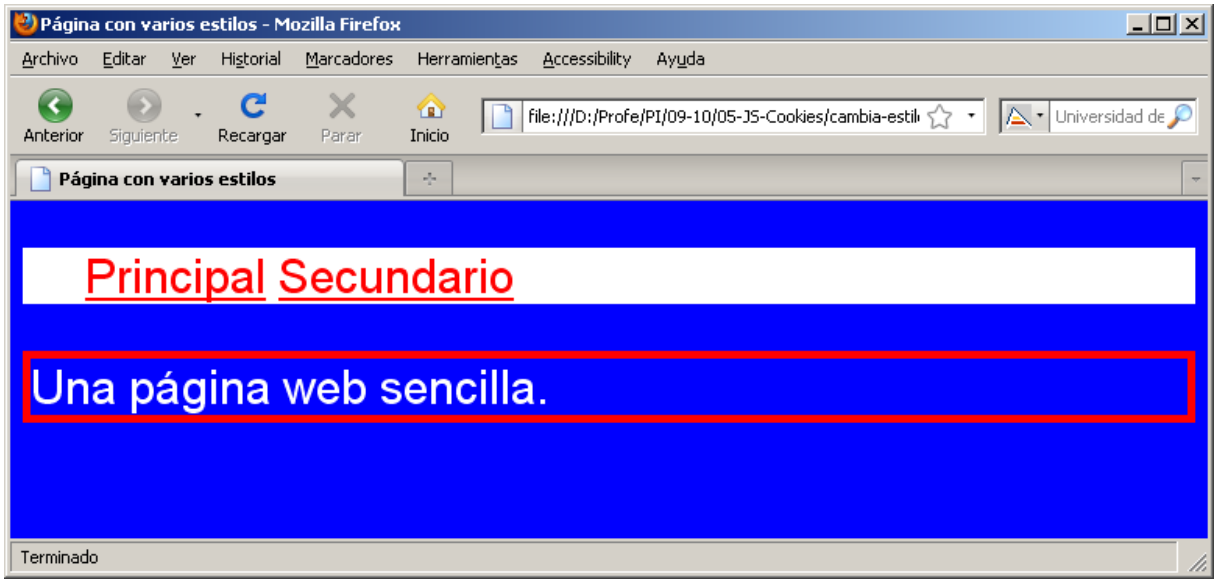

Figura 6: Estilo principal

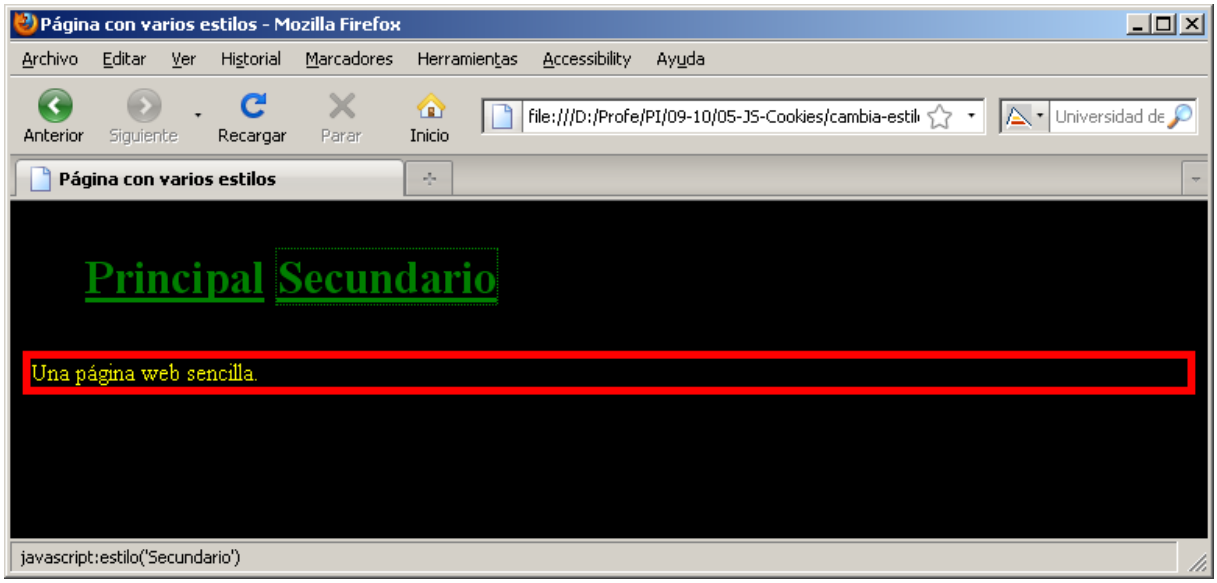

Figura 7: Estilo secundario

```
// Código de http://www.w3schools.com/js/js_cookies.asp
function setCookie(c_name, value, expiredays) {
  var exdate = new Date();
  exdate.setDate(exdate.getDate() + expiredays);
  document.cookie = c_name + "=" + escape(value) +
    ((expiredays == null) ? "" : ";expires="+ exdate.toGMTString());
}

function getCookie(c_name) {
  if(document.cookie.length > 0) {
    c_start = document.cookie.indexOf(c_name + "=");
    if(c_start != -1) {
      c_start = c_start + c_name.length + 1;
      c_end = document.cookie.indexOf(";", c_start);
      if(c_end == -1)
        c_end = document.cookie.length;

      return unescape(document.cookie.substring(c_start, c_end));
    }
  }
  return "";
}
</script>
</head>
<body>
<p>
<script>
var ultimavez = getCookie('ultimavez');

if(ultimavez != null && ultimavez != "") {
  document.writeln("Tu última visita fue: " + ultimavez);
  setCookie('ultimavez', Date(), 365);
}
else {
  document.writeln("Bienvenido por primera vez a este sitio web");
  setCookie('ultimavez', Date(), 365);
}
</script>
</p>
</body>
</html>
```

Para borrar una cookie, se tiene que asignar a la cookie una fecha de caducidad (`expires`) en el pasado, es decir, una fecha anterior a la actual.

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**[24] puedes ver la implementación de los más conocidos: *bubble sort*, *insertion sort*, *shell sort* y *quick sort*. O quizás haya otra forma más fácil de hacerlo, piénsalo, no te pongas a programar "a lo loco" sin pensar un poco antes.

La página web **HTML DOM Link Object**[25] explica las propiedades de la etiqueta `<link/>` que son accesibles a través del DOM.

En el ejemplo de este enunciado, la lista de estilos que puede seleccionar el usuario está escrita directamente en la página web. Si se añade un nuevo estilo alternativo, es necesario añadirlo a mano a la lista de selección. ¿Se puede automatizar el proceso, es decir, se puede crear de forma automática la lista de estilos que puede seleccionar el usuario?

[24] http://www.the-art-of-web.com/javascript/
[25] http://www.w3schools.com/jsref/dom_obj_link.asp

¡Cuidado! Por defecto, Google Chrome no proporciona soporte para el manejo de cookies a nivel local[26] (`file://`). Para realizar esta práctica, puedes activar el soporte de las cookies a nivel local con el modificador `--enable-file-cookies` al ejecutar Google Chrome o, mucho más sencillo, puedes usar otro navegador. También puedes emplear un servidor web para que las cookies funcionen correctamente sin tener que hacer nada.

Existen diferentes opciones para consultar, modificar y eliminar las cookies que almacena un navegador web. Por ejemplo, en Google Chrome puedes emplear la extensión Web Developer[27] (también disponible para Mozilla Firefox[28]) y en Mozilla Firefox puedes emplear Cookie Manager[29] o Cookie Quick Manager[30].

La Figura 8 muestra las opciones del menú Cookies de Web Developer.

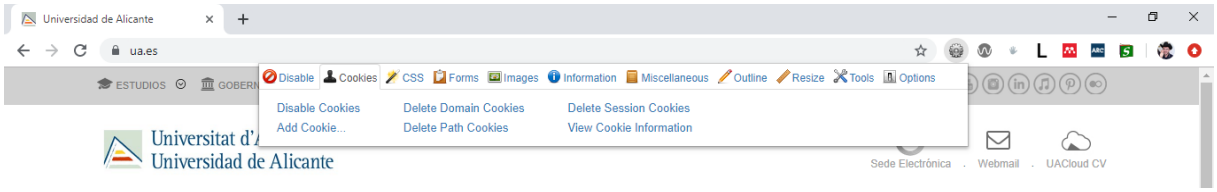

Figura 8: Menú "Cookies" de Web Developer

La Figura 9 muestra las cookies que contiene la página principal del sitio web de la Universidad de Alicante mediante la opción "View Cookie Information" de Web Developer. Para cada cookie se muestra su nombre, valor, dominio y ruta en la que existe, fecha de caducidad y los *flags* de segura y solo disponible en el servidor. Se puede ver que hay un botón para eliminar (*Delete*) la cookie y otro para modificarla (*Edit*).

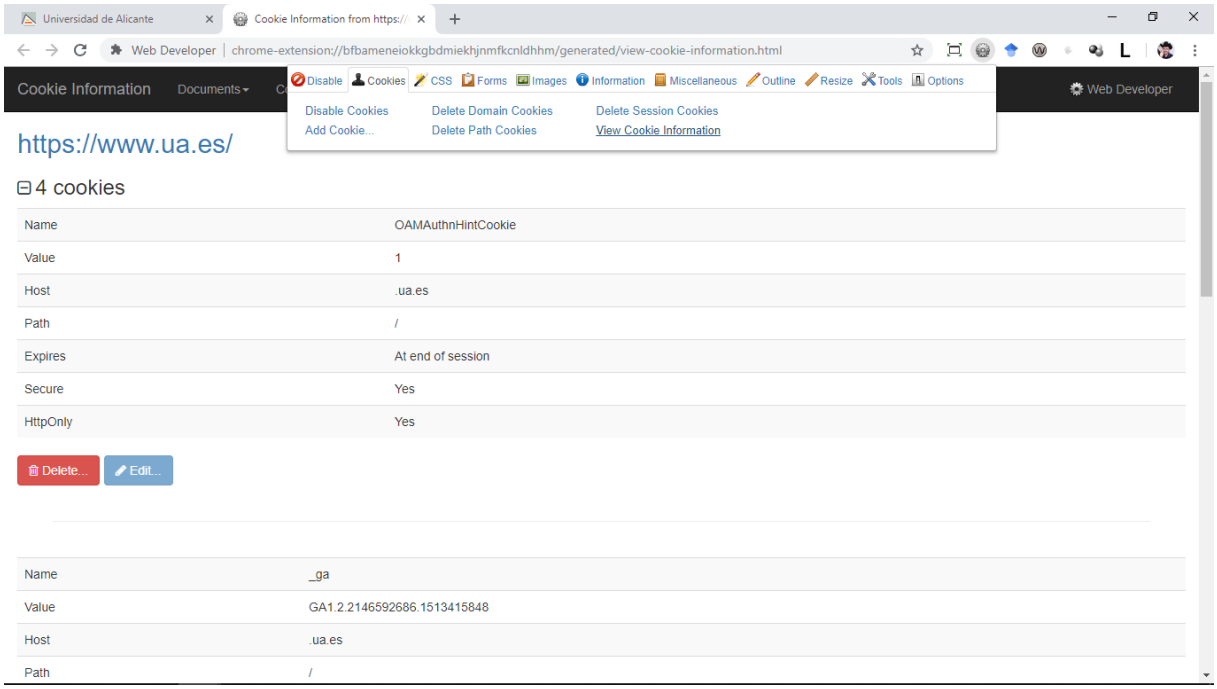

Figura 9: "View Cookie Information" de Web Developer

La Figura 10 muestra las cookies que contiene el navegador web Mozilla Firefox mediante Cookie Quick Manager. En el panel de la izquierda Domains aparecen todos los sitios web que han almacenado alguna cookie. Esta extensión también permite modificar y eliminar las cookies almacenadas.

[26] Support cookies on file://: https://code.google.com/p/chromium/issues/detail?id=535
[27] `https://chrome.google.com/webstore/detail/web-developer/bfbameneiokkgbdmiekhjnmfkcnldhhm?hl=es`
[28] `https://addons.mozilla.org/es/firefox/addon/web-developer/`
[29] `https://addons.mozilla.org/es/firefox/addon/a-cookie-manager/`
[30] `https://addons.mozilla.org/es/firefox/addon/cookie-quick-manager/?src=search`

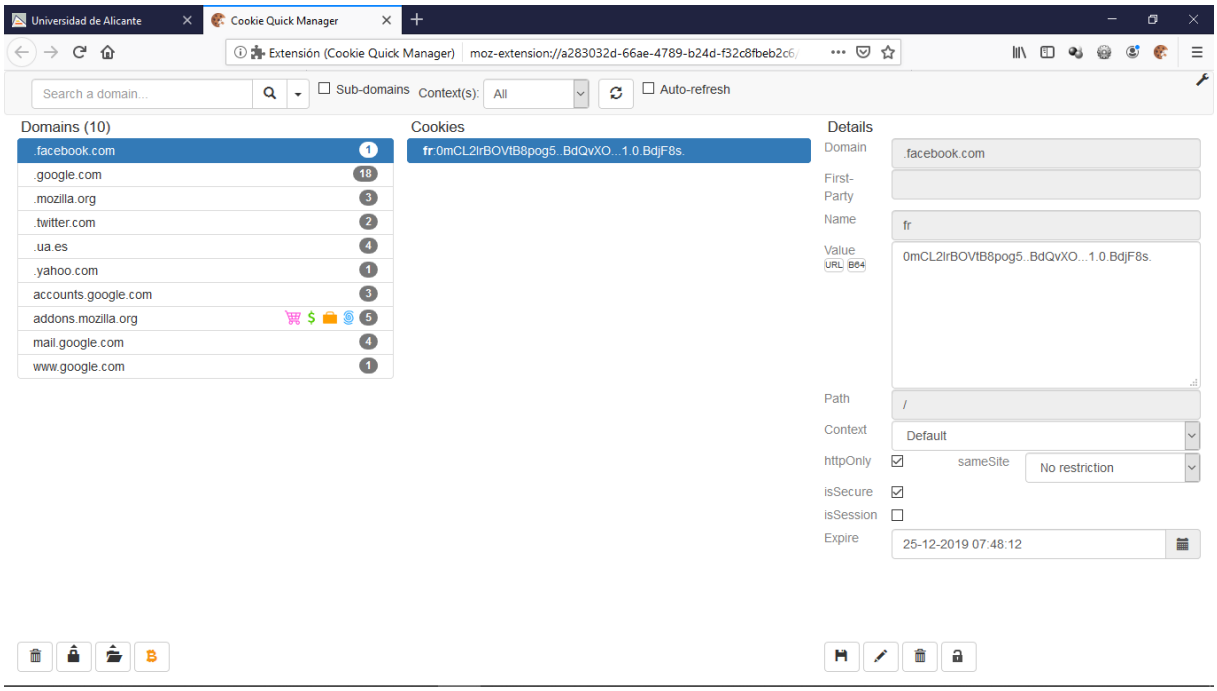

Figura 10: Cookie Quick Manager