

U.D 5

Eventos y formularios

CIFP Rodolfo Ucha Piñeiro 2020/2021

ÍNDICE

RELACIÓN CO CURRÍCULO	3
1. EVENTOS.....	4
A. EVENTOS DE RATÓN	5
B. EVENTOS DEL TECLADO	5
C. EVENTO HTML	6
D. EVENTO DOM	6
2. FORMULARIOS.....	6
A. ESTRUCTURA.....	6
B. ELEMENTOS	7
Etiqueta input	8
Estructura	8
Tipos	9
Select y optiongroup	13
Texarea.....	14
Botones	14
Label	14
Fieldset y legend	14
Datalist (HTML 5)	15
keygen (HTML 5)	15
output (HTML 5).....	15
C. MODIFICAR APARIENCIA Y COMPORTAMIENTO	16
Apariencia	17
Comportamiento.....	17
D. VALIDACIÓN Y ENVÍO.....	18
E. EXPRESIONES REGULARES	21
Caracteres especiales.....	22
Validar un formulario con expresiones regulares	22
F. COOKIES	24
Uso	24
Implementación	25
BIBLIOGRAFÍA	27

RELACIÓN CO CURRÍCULO

RA = Resultado de aprendizaxe (composto por CA - criterios de avaliación)

BC = Bloque de contido

RESULTADOS DE APRENDIZAXE E CRITERIOS DE AVALIACIÓN

RA5. Desenvolve aplicacións web interactivas integrando mecanismos de manexo de eventos.

CA5.1. Recoñecéronse as posibilidades da linguaxe de marcas relativas á captura dos eventos.

CA5.2. Identificáronse as características da linguaxe relativas á xestión dos eventos.

CA5.3. Diferenciáronse os tipos de eventos que se poden manexar.

CA5.4. Creouse un código que capture e utilice eventos.

CA5.5. Recoñecéronse as capacidades da linguaxe relativas á xestión de formularios web.

CA5.6. Validáronse formularios web utilizando eventos.

CA5.7. Utilizáronse expresións regulares para facilitar os procedementos de validación.

CA5.8. Probouse e documentouse o código.

CONTIDOS BÁSICOS

BC5. Interacción co usuario: eventos e formularios.

1. Modelo de xestión de eventos.

2. Uso de formularios desde código.

3. Modificación de aparencia e comportamento.

4. Validación e envío.

5. Expresións regulares nos procedementos de validación de formularios.

APARTADOS DA UNIDADE DIDÁCTICA

APARTADO	CRITERIO AVALIACIÓN	CONTIDO
1. Eventos	5.1, 5.2, 5.3, 5.4, 5.8	1
2. Formularios	5.5, 5.6, 5.7, 5.8	2, 3, 4, 5

La **interacción**, es la relación que establece el usuario con la aplicación web. Para ello en el lado del **usuario** se utilizan mecanismos de entrada como el teclado o el ratón que generarán los **eventos**. En el lado de la **maquina** se muestra la información a tratar, por ejemplo sobre un **formulario web**.

1. EVENTOS

Un evento es mecanismo que acciona el usuario cuando realiza un **cambio sobre una página web**. Por ejemplo, cerrar la ventana, hacer clic en un botón de radio o situarse con el ratón sobre la página.

En una página web existen multitud de elementos que la modelan. El encargado de crear la jerarquía de objetos que compone una página web es el **DOM (Document Object Model)** y por tanto, es el encargado de gestionar los eventos.

Para controlar un evento es necesario un **manejador**, es decir, una palabra reservada que indica la **acción** que va a manejar. *Ejemplo: para el evento click, el manejador sería **onClick**.*

JavaScript es un lenguaje interpretado que tiene la capacidad de actuar sobre los objetos DOM sin enviar una nueva petición al servidor, es decir, puede modificar las propiedades de los objetos de una página en el mismo navegador.

Ejemplo:

```
<html>
<head>
  <title> Prueba</title>
  <script>
    function mensaje() { alert ("Click en imagen"); }
  </script>
</head>
<body>
  
</body>
</html>
```

El manejador onclick llama a la función mensaje para que se ejecute. Esta función debe definirse dentro de un archivo js o entre las etiquetas <script> dentro de la cabecera.

La especificación de DOM define grupos de eventos dividiéndolos según su origen. Estos son:

- **Eventos de ratón:** cuando el usuario hace uso del ratón para realizar una acción. Cualquier movimiento de la flecha del ratón o acción sobre sus botones, puede desencadenar un evento.
- **Eventos del teclado:** cuando el usuario pulsa alguna tecla del teclado.
- **Eventos HTML:** cuando hay algún cambio en la página del navegador.
- **Eventos DOM:** cuando existe algún cambio en la estructura DOM de la página.

A. EVENTOS DE RATÓN

Son los más usuales, ya que la mayor parte de las acciones en la web se realizan a través de este.

- **onclick:** se pulsa el botón izquierdo del ratón.
- **oncontextmenu:** se pulsa el botón derecho del ratón.
- **ondblclick:** se hace doble clic sobre el botón izquierdo del ratón.
- **onmousedown:** se pulsa un botón del ratón.
- **onmouseout:** el puntero del ratón esta dentro de un elemento y es desplazado fuera de él.
- **onmouseover:** el puntero esta fuera de un elemento y éste se desplaza hacia el interior.
- **onmouseup:** se suelta un botón del ratón previamente pulsado.
- **onmousemove:** el puntero del ratón se encuentra dentro de un elemento. Se producirá continuamente mientras el puntero del ratón permanezca dentro del elemento.

Cuando se produce un evento, crea automáticamente el **objeto event** que contiene características adicionales como la posición del ratón (**screenx**, **screenY**), el nombre del evento (**type**), el elemento que origina el evento (**button**), etc.

La especificación DOM indica que el único parámetro que se debe pasar a las funciones es el objeto event.

B. EVENTOS DEL TECLADO

Los eventos de teclado son los que suceden cuando pulsamos una tecla.

- **onkeydown:** al pulsar una tecla. Si se mantiene pulsada, el evento se produce hasta que se suelte.
- **onkeypress:** al pulsar una tecla de carácter alfanumérico (no: Enter, barra espaciadora, etc.).
- **onkeyup:** cuando se suelta una tecla.

Las **propiedades de event** para los eventos de teclado son: código numérico de la tecla pulsada (**keycode**), código unicode del carácter correspondiente a la tecla pulsada (**charcode**), elemento que origina el evento (**target**) y otras para identificar si se ha pulsado shift (**shiftKey**), control (**ctrlKey**), alt (**altKey**) o meta (**metaKey**).

Cuando se pulsa una tecla de un carácter alfanumérico, la secuencia de eventos es: keydown, keypress, keyup. En el caso de pulsar una tecla que no sea un carácter alfanumérico, la secuencia es: keydown,keyup.

Si se deja una tecla pulsada, en el primer caso, se repiten de forma continua los eventos keydown y keypress. En el segundo caso se repite de forma continuada el evento keydown.

C. EVENTO HTML

Los eventos HTML, son los que actúan cuando hay cambios en la ventana del navegador.

- **onload:** hace referencia a la carga de distintas partes de la página. Se produce en el objeto **window** cuando la página se ha cargado por completo. En el elemento **img** cuando la imagen se ha cargado. En el elemento **object** al cargar el objeto completo
- **onerror:** se produce cuando se ha producido un error. *Ejemplo: img y object no se cargan bien*
- **onselect:** cuando se selecciona texto de los cuadros de textos input y textarea
- **onchange:** cuando los cuadros de texto input y textarea pierden el foco y el contenido que tenían ha variado. También cuando un elemento select cambia de valor
- **onsubmit:** cuando se pulsa sobre un botón de tipo submit
- **onreset:** cuando se pulsa sobre un botón de tipo reset
- **onresize:** cuando se redimensiona el navegador. Actúa sobre el objeto window
- **onscroll:** cuando varía la posición de la barra de desplazamiento (scroll) en cualquier elemento
- **onfocus:** cuando un elemento obtiene el foco
- **onblur:** cuando un elemento pierde el foco

D. EVENTO DOM

Estos eventos hacen referencia a la especificación DOM. Se accionan **cuando varía el árbol DOM**. No están implementados en todos los navegadores.

- **DOMSubtreeModified:** al añadir o eliminar nodos en el subárbol de un elemento o documento
- **DOMNodeInserted:** al añadir un nodo hijo a un nodo padre
- **DOMNodeRemoved:** al eliminar un nodo que tiene nodo padre
- **DOMNodeRemovedFromDocument:** al eliminar un nodo del documento
- **DOMNodeInsertedIntoDocument:** al añadir un nodo al documento

2. FORMULARIOS

Permiten gestionar los datos enviados y recibidos entre un cliente (navegador) y un servidor web.

A. ESTRUCTURA

Los elementos del formulario se definen entre las etiquetas **<form>** y **</form>**.

Los atributos de esta etiqueta son:

- **action:** contiene la URL donde se redirigirán los datos del formulario.
- **method:** indica el método de envío de los datos. Los métodos de envío son:
 - POST: datos ocultos, no encriptados. Permite un envío de datos mayor que GET y es algo más seguro.
 - GET: como máximo se puede enviar 500 bytes. Los datos viajan en la URL a continuación de un ?. No permite el envío de ficheros adjuntos. Es útil cuando se envían un gran número de fotografías o vídeos para pasar el identificador de cada uno de ellos y para utilizar en tiempo de desarrollo ya que permite validar los datos y comprobar errores de una forma más visual.
- **enctype:** define el tipo de codificación para enviar el formulario al servidor. Se indica cuando el formulario permite enviar archivos adjuntos. Sus posibles valores son: application/x-www-form-urlencoded, multipart/form-data, text/plain.
- **accept-charset:** para formularios que aceptan ficheros adjuntos, indica el tipo de archivos que acepta el servidor.
- **name:** nombre utilizado para identificar el formulario
- **target:** especifica donde se abrirá la url especificada en action. Los posibles valores son: _blank, _self, _parent, _top. Por defecto, _self
- **autocomplete:** (html 5). Indica si el navegador autocompleta el formulario. Por defecto, on.
- **novalidate:** (html 5). Especifica que el navegador no debería validar el formulario.

Ejemplo:

```
<html>
<head>
  <title>Ejemplo de formulario</title>
</head>
<body>
  <form name="altausuario" action="alta_usuario.php" method="GET" target="_blank"
    accept-charset="UTF-8" autocomplete="off" novalidate enctype="application/x-www-form-
    urlencoded" >
    </form>
</body>
</html>
```

B. ELEMENTOS

El elemento principal del formulario se incluye con la etiqueta <input>. Los hay de varios tipos, unos guardan los datos a enviar (cajas de texto, botones de selección, casillas de verificación) y otros desencadenan acciones (botones).

Etiqueta input

Estructura

Es una etiqueta unaria <input ... />, se cierra a sí misma con la barra (/) al final. Contiene los atributos:

- **type:** tipo de elemento. El resto de parámetros varían en función del tipo. Los valores posibles son:
 - **text:** cuadro de texto
 - **password:** cuadro de contraseña, los caracteres aparecen ocultos con asteriscos
 - **checkbox:** casilla de verificación
 - **radio:** opción de entre dos o más
 - **submit:** botón de envío del formulario
 - **reset:** botón de vaciado de campos
 - **file:** botón para buscar ficheros
 - **hidden:** campo oculto
 - **image:** botón de imagen en el formulario
 - **button:** botón del formulario
 - **otros:** color, date, datetime, datetime-local, email, month, number, range, search, tel, time, url, week
- **name:** asigna un nombre al elemento. Si no se asigna, el servidor no tendrá acceso a él.
- **value:** inicializa el valor del elemento.
- **size:** tamaño inicial del elemento (px). En los campos text y password indica el número de caracteres.
- **maxlength:** número máximo de caracteres que pueden contener los elementos text y password.
- **checked:** exclusivo de checkbox y radio. Define la opción seleccionada por defecto.
- **disabled:** deshabilita el elemento. El dato no se envía al servidor.
- **readonly:** bloquea el contenido del control. El valor del elemento no se podrá modificar.
- **src:** exclusivo para asignar una URL a una imagen que ha sido establecida como botón del formulario.
- **alt:** descripción del elemento. Se visualiza en caso de que no se ha desactivado, cuando se posiciona el ratón encima del elemento.

En HTML 5:

- **autocomplete:** si el elemento permite que se autocomplete.
- **autofocus:** elemento que obtiene el foco al cargar la página
- **form:** id de los formularios a los que pertenece.
- **formaction:** función igual a action.
- **formenctype:** función igual a enctype.
- **formmethod:** función igual a method.
- **formnovalidate:** función igual a novalidate.
- **formtarget:** función igual a target.
- **width:** para el elementos con type="image" especifica el ancho del elemento.
- **height:** para el elementos con type="image" especifica el alto del elemento
- **required:** especifica que tiene que estar cubierto antes de enviar el formulario.
- **placeholder:** especifica una descripción de lo que debería contener el elemento.
- **pattern:** para especificar una expresión regular.
- **multiple:** especifica que el usuario puede introducir más de un valor para ese elemento.
- **min:** especifica el mínimo valor para un elemento (fecha o número)
- **max:** especifica el máximo valor para un elemento (fecha o número)
- **list:** hace referencia a un datalist.
- **step:** especifica los intervalos numéricos para un elemento <input>.

Tipos

Su objetivo es **adecuar los datos** para facilitar el manejo y envío de datos a través del formulario.

- **Cuadro de texto:** lugar en el que el usuario puede introducir un texto. Pueden ser útiles los atributos maxlength (*un teléfono no debería permitir más de 9 caracteres*) o readonly (*no permitir modificar del nombre del usuario al modificar el resto de datos*).

Nombre <input type="text" name="nome" maxlength="25" readonly />

Nombre

- **Cuadro de contraseña:** los caracteres que escribe el usuario no se ven en pantalla. Sirven para escribir datos sensibles del usuario, como contraseñas.

Contraseña: <input type="password" name="contrasinal">

Contraseña:

- **Casilla de verificación:** permiten activar o desactivar las casillas de forma individual. Si alguna opción quiere seleccionarse por defecto, es necesario utilizar checked.

```
<input type="checkbox" name="sport" value="natacion" checked> Nadar<br>
<input type="checkbox" name="sport" value="atletismo"> Correr
```

☒ Nadar
☐ Correr

- **Opción de radio:** opciones excluyentes entre sí. Al seleccionar una opción, automáticamente se deselecciona la seleccionada. El nombre de los elementos que pertenezcan a un mismo grupo de opciones de radio deberá ser el mismo. También se puede utilizar el atributo checked.

```
<input type="radio" name="sexo" value="home" checked> Hombre <br>
<input type="radio" name="sexo" value="mujer"> Mujer
```

☒ Hombre
☐ Mujer

- **Ficheros adjuntos:** permite adjuntar ficheros. Las limitaciones sobre el número de ficheros y su tamaño deberá controlarse desde el código para evitar desbordamientos en el servidor. El elemento proporciona un cuadro de texto para almacenar la dirección del fichero adjunto. Pulsando un botón situado a la derecha del cuadro de texto se navega por el árbol de directorios en busca del fichero.

Para adjuntar archivos hay que indicar el tipo de envío a realizar con el atributo **enctype** de la etiqueta `<form>`. Habitualmente el valor que toma es **multipart/form-data**.

```
<form action="alta.php" method="post" enctype="multipart/form-data">
  <input type="file" name="adjunto" />
</form>
```

 No se ha seleccionado ningún archivo.

- **Campos ocultos:** no son visibles en el formulario por el usuario. Son útiles para que el servidor pueda tratar la información o hacer alguna acción determinada.

```
<input type="hidden" name="opcion" value="si">
```

- **Botón de imagen:** cambia el aspecto que tienen los botones de un formulario por una imagen.

```
<input type="image" src="red.png" width="25" height="15">
```



- **Botón:** se le pueden asociar diferentes funciones a través de javascript. El texto del botón es el que se asigne al atributo value.

```
<input type="button" onclick="alert('Hola!')" value="te voy a saludar">
```

- **Botón de envío:** envía los datos del formulario al servidor. El atributo type toma el valor submit. El valor del atributo value se muestra en el botón. El botón enviará los datos del formulario a la dirección del action. En caso de que no se incluya el atributo value aparecerá "Enviar consulta".

```
<input type="submit" value="Envio formulario">
```


- **Botón de reset:** borra el contenido de los elementos del formulario. En caso de que no se incluya el atributo value aparecerá "Restablecer".

```
<input type="reset" value="Borrar datos">
```

Entre los elementos de HTML 5 destacan los siguientes:

- **number:** para introducir valores numéricos. Se le pueden añadir restricciones, entre ellas: **disabled**, **max**, **maxlength**, **min**, **pattern**, **readonly**, **required**, **size**, **step**, **value**.

Número: `<input type="number" name="cantidad" min="1" max="5">`



Número: `<input type="number" name="cantidad" min="1" max="5" step="2">`



En este caso solo valdrían los números 1,3 y 5 (1+2, y 3+2)

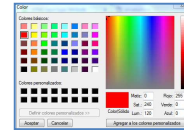
- **range:** para elementos que deben contener un valor dentro de un rango. En algunos navegadores se presenta como una barra de deslizamiento.

`<input type="range" name="puntos" min="0" max="10">`



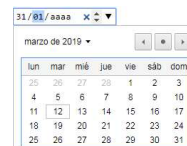
- **Color:** muestra una paleta de colores para poder seleccionar uno de ellos, o escribir su codificación. Aparece seleccionado por defecto el color que se ponga en el campo "value".

`<input type="color" name="cor" value="#ff0000">`



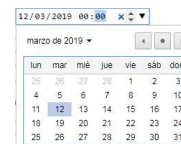
- **Fecha:** muestra el año, mes y día. Utiliza dos caracteres para cada uno de los datos. Presenta unas flechas que ayudan a seleccionar los números. Además incorpora un triángulo que al pulsarlo muestra un calendario que facilita la introducción del dato. Si se pulsa sobre el círculo, selecciona la fecha actual. El aspa, borra los datos introducidos.

Fecha aa/mm/dd `<input type="date" name="fecha">`



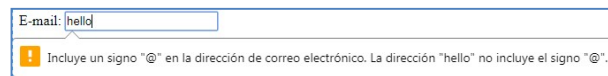
- **Fecha completa:** Igual que el anterior pero mostrando la hora. No lo soporta safari, ni firefox.

Fecha aa/mm/dd/hh `<input type="datetime-local" name="fh">`



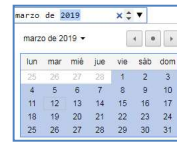
- **Email:** es un campo de texto que comprueba que los datos introducidos se correspondan con una dirección de correo electrónico, una vez se pulse enter sobre la caja de texto. En realidad, simplemente comprueba que se introduzca una @.

E-mail: `<input type="email" name="correo">`



- **Fecha corta:** muestra el mes en texto y el año con dígitos. Una vez se pulsa sobre una letra, automáticamente escribe el mes correspondiente a esa letra. No soportado por Safari ni Firefox.

Fecha mm/aa:



- **Búsqueda:** cadena de caracteres que se enviarán al servidor como búsqueda para su motor.

Búsqueda:

Búsqueda:

- **Teléfono:** sólo soportado por Safari. Valida que el dato introducido sea un teléfono. En caso de que el navegador no lo soporte, actúa como si fuese un campo de texto.

Teléfono

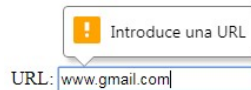
- **Hora:** muestra la hora con horas y minutos.

Hora:

Hora:

- **URL:** es necesario introducir una URL correcta, y para ello hay que introducir el protocolo previamente, http://dirección.es.

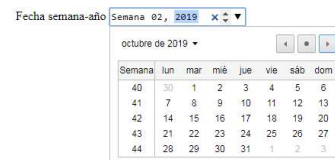
URL:



URL:

- **Semana:** No soportado por firefox ni Safari.

Fecha semana-año



```
<html>
<head></head>
<body>
  <form action="pagina.php" method="post" enctype="multipart/form-data" /> <br/>
```

```
<H1> ELEMENTOS DE HTML 4.01 </H1>
```

```
Texto: <input type="text" name="texto" value="" size="42" maxlength="30" /> <br/>
```

```
Contraseña : <input type="password" name="contraseña" value="" size="40" maxlength="80" /> <br/>
```

```
Verifica <input type="checkbox" name="verifica" value="verifica" checked="checked" /> Verifica <br/>
```

```
Radio: <br/><input type="radio" name="opciones" value="opcion1" checked="checked" />
```

```
Opcion1 <br/><input type="radio" name="opciones" value="opcion2" /> Opcion2 <br/><br/>
```

```
Adjuntos: <input type="file" name="adjunto" /> <br/><br/>
```

```
Oculto: <input type="hidden" name="oculto" value="oculto"> <br/><br/>
```

```
Imagen: <input type="image" src="red.png" width="25" height="15"> <br/><br/>
```

```
Botón: <input type="button" onclick="alert('Hola!')" value="Funcion boton"> <br/><br/>
```

```
Enviar Datos <input type="submit" name="enviar" value="Enviar datos" /> <br/><br/>
```

```
Borrar datos <input type="reset" name="limpiar" value="Borrar datos introducidos" /> <br/><br/>
```

<H1> ELEMENTOS DE HTML 5 </H1>

```
Color: <input type="color" name="eligecolor" value="#ff0000"> <br/><br/>
Fecha aa/mm/dd <input type="date" name="fecha"><br/><!-- No soportado por safari -->
Fecha aa/mm/dd/hh <input type="datetime-local" name="fechahora"><br/> <!-- No firefox,safari -->
Fecha semana-año <input type="week" name="semanaaño"><br/> <!-- No firefox ni safari -->
Fecha mm/aa <input type="month" name="fechacorta"><br/><!-- No soportado por firefox y safari -->
Hora: <input type="time" name="hora"><br/><br/> <!-- No soportado por safari -->
E-mail: <input type="email" name="correo"><br/><br/>
Búsqueda: <input type="search" q="busqueda"><br/><br/>
Teléfono <input type="tel" name="telefono"><br/><br/> <!-- Sólo soportado por safari --->
URL: <input type="url" name="dweb"><br/><br/>
</form>
</body>
</html>
```

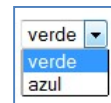
Select y optiongroup

Define una lista de elementos. Los elementos se incluyen entre las etiquetas **<select>** y **</select>**. Si se permite seleccionar varias opciones es necesario incluir el atributo **multiple**. Se pueden incluir los atributos: **disabled**, **name** y **size**. En html5 **autofocus**, **form** y **required**.

Cada elemento se define entre **<option>** y **</option>**. Normalmente aparece seleccionado el primer elemento de la lista, pero puede añadirse un atributo **selected** para definir cuál va a ser el elemento seleccionado por defecto. Otros atributos posibles son: **disabled**, **label** y **value**

Ejemplo:

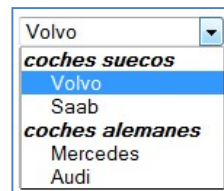
```
<select name="cars">
  <option value="green"> verde </option>
  <option value="blue"> azul </option>
</select>
```



La etiqueta **<optgroup>** se utiliza para **agrupar conjuntos de opciones**. Puede estar acompañada de los atributos **disabled** y **label**.

Ejemplo:

```
<select>
  <optgroup label="coches suecos">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
  </optgroup>
  <optgroup label="coches alemanes">
    <option value="mercedes">Mercedes</option>
    <option value="audi">Audi</option>
  </optgroup>
</select>
```



Texarea

En este elemento se pueden incluir varias líneas de texto. Se define con las etiquetas `<texarea>` y `</texarea>`.

Sus posibles **atributos** son:

- **rows y cols:** número de filas y columnas, respectivamente de la caja de texto.
- **otros:** disabled, name y readonly
- **html5:** autofocus, form, maxlength, placeholder, required.

```
<textarea name="opinion" rows="10" cols="30">  
    Creo que esta aplicación es muy buena.  
</textarea>
```

Botones

Permite hacer click sobre él para ejecutar alguna acción.

```
<button type="button" onclick="alert('Hola!')"> Te voy a saludar </button>
```

Sus posibles atributos son: **disabled, name, type, y value**. En html5, **autofocus, form, formaction, formenctype, formmethod, formnovalidate, formtarget, ...**

```
<button type="submit" formmethod="post" formaction="alta.asp" formtarget="_blank">  
Enviar form </button>
```

Label

Define el elemento al que acompaña con la etiqueta `<label>`. Puede incluir el atributo **for** que especifica el id del elemento al que acompaña. En html5 se puede utilizar el atributo **form** que especifica el id de los formularios a los que pertenece.

```
<label for="nome"> Nombre </label>  
<input type="text" name="nombre" id="nome" value="">
```

Fieldset y legend

La etiqueta `<fieldset>` se utiliza para agrupar elementos. Los atributos son todos de html5 y son: **disabled, form y name**.

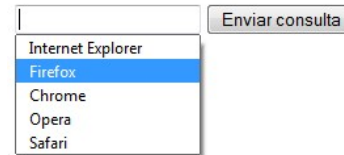
La etiqueta `<legend>` sirve para poner un título al elemento fieldset.

```
<fieldset>  
    <legend> Administrador </legend>  
    Nombre: <input type="text"><br><br>  
    Email: <input type="text"><br>  
</fieldset>
```

Datalist (HTML 5)

Especifica una **lista de opciones predefinidas para un elemento** <input>, aunque en la caja se puede escribir cualquier otro valor. Los usuarios verán una lista desplegable de opciones predefinidas. El atributo de lista del elemento <input>, debe hacer referencia al atributo id del elemento <lista de datos>.

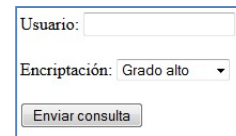
```
<form action="action_page.php">
<input list="navegadores" name="browser">
<datalist id="navegadores">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
<input type="submit">
</form>
```



keygen (HTML 5)

Su propósito es proporcionar una **forma segura para autenticar usuarios**. Cuando se envía el formulario, dos claves se generan, una privada y una pública. La clave privada se almacena localmente, y la clave pública se envía al servidor. No soportado en Explorer ni Chrome.

```
<form action="alta.php">
  Usuario: <input type="text" name="usuario">
  <br><br>
  Encriptación: <keygen name="security">
  <br><br>
  <input type="submit">
</form>
```



output (HTML 5)

Representa el **resultado de un cálculo**.

```
<form action="pagina.php" oninput="x.value=parseInt(a.value)+parseInt(b.value)">
  0 <input type="range" id="a" name="a" value="50">
  100 + <input type="number" id="b" name="b" value="50">
  = <output name="x" for="a b"></output><br><br>
  <input type="submit">
</form>
```



```
<html>
  <head></head>
  <body>
    <form action="pagina.php" oninput="x.value=parseInt(a.value)+parseInt(b.value)">
      0 <input type="range" id="a" name="a" value="50">
      100 + <input type="number" id="b" name="b" value="50">
      = <output name="x" for="a b"></output><br><br>
    </form>
  </body>
</html>
```

```

<html>
<head> </head>
<body>
  <form action="pagina.php" method="post" enctype="multipart/form-data" /> <br/>
  <H1> ELEMENTOS DE HTML 4.01 </H1>
  <select name="opciones">
    <option value="opcion1"> opción1 </option>
    <option value="opcion2"> opción2 </option>
  </select> <br><br>
  <select>
    <optgroup label="grupo1">
      <option value="opcionA"> opcion A </option>
      <option value="opcionB"> opcion B </option>
    </optgroup>
    <optgroup label="grupo2">
      <option value="opcionI"> opcion I </option>
      <option value="opcionII"> opcion II </option>
    </optgroup>
  </select><br><br>
  <textarea name="areatexto" rows="10" cols="30"> Texto por defecto </textarea><br><br>
  <button type="button" onclick="alert('Hola!')"> Ejecuta función </button><br><br>
  <label for="nome"> Etiqueta </label><input type="text" name="cajatexto" id="nome" value=""><br><br>
  <fieldset>
    <legend> Legend </legend>
    Caja1: <input type="text"><br><br>
    Caja2: <input type="text"><br>
  </fieldset><br><br>
  <H1> ELEMENTOS DE HTML 5 </H1>
  <datalist <input list="opciones" name="selector">
    <datalist id="opciones">
      <option value="Opcion1">
      <option value="Opcion2">
    </datalist><br><br>
  </form>
</body>
</html>

```

Actividad: FORMULARIO

Crea un formulario con todos los tipos de elementos, tanto de HTML 4.01 como de HTML 5.

C. MODIFICAR APARIENCIA Y COMPORTAMIENTO

A través de los estilos se puede mejorar notablemente el aspecto de los formularios, redondeando más sus formas, haciendo más uniformes los colores y estructurando las partes del formulario para que estos obtengan un aspecto mucho más agradable.

Apariencia

Los formularios, por defecto, tienen unos estilos asignados con unos colores y bordes determinados. Para modificar la apariencia de los elementos principales de un formulario:

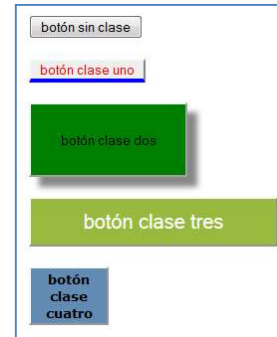
- **Aspecto de un botón**

```
.uno {color:red;
border-bottom:4px
solid blue;}

.dos {padding: 2em;
box-shadow: 10px 10px 5px #888888;
background-color:green;}

.tres {width:250px;
padding: 10px 25px 10px 25px;
font-size:20px;
color:#ffffff;
background-color: #98ba3f;}

.cuatro {font-family:Verdana;
font-weight:bold;
background:#638cb5;
width:80px;
height:59px;}
```

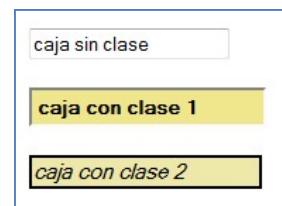


- **Suavizar el aspecto de un campo de texto**

Los campos de texto por defecto comienzan a escribir justo al principio del elemento, desde css se puede modificar el punto a partir del que se comienza a escribir los caracteres.

```
.uno {
background-color: khaki;
padding: .3em;
font-weight: bolder;}

.dos {
background-color: palegoldenrod;
font-size: 15px;
font-style: italic;
border: 2px solid black;}
```



Comportamiento

Los formularios tienen acciones predeterminadas que se pueden modificar en caso de necesidad. *Ejemplo: si se desea que los datos de un formulario se envíen a URL diferentes dependiendo de un dato introducido.*

```
function enviar(form) {
    if (formulario.alta.checked === true) {
        formulario.action = "paginas/alta.html"; }
    if (formulario.alta.checked === false) {
        formulario.action = "paginas/baja.html"; }
    form.submit(); }
```

Se comprueba si la casilla alta ha sido chequeada. En caso de haber sido pulsada el action del form se inicia a un valor, si no ha sido pulsada se inicia a otro. Al final se envía el formulario por medio de un submit. Para que la función enviar sea ejecutada, hay que incluir un input de tipo button que contenga el evento onclick: `OnClick="enviar(this.form)"`. El objeto this hace que se pueda enviar el formulario a través de la función. De esta forma nunca se llama al action de la estructura principal del formulario.

D. VALIDACIÓN Y ENVÍO

Para controlar que los campos de un formulario han sido rellenados correctamente es necesario hacer uso de las validaciones.

Las **validaciones a nivel de servidor** suponen que el servidor recibe una petición, la procesa y emite una respuesta. En el caso de que el usuario incluya muchos datos que no son correctos el servidor se puede ver sobrecargado.

Para **solucionar la sobrecarga del servidor** se realizan las **validaciones a nivel del cliente**. En ellas, a través del navegador, se analizan con scripts los datos que ha incluido el usuario una vez haya pulsado el botón de enviar formulario. Si existe algún dato incorrecto se muestra un mensaje al usuario indicando que debe realizar alguna modificación. Si los datos se validan como correctos se envía el formulario al servidor.

En función de la respuesta del **evento onsubmit**, el formulario **envía los datos al servidor o muestra diferentes mensajes** de error.

Ejemplo: `<form action="alta.php" method="get" name="altausuario" onsubmit="return valida()">`

En este caso el formulario enviaría los datos a "alta.php" siempre y cuando la función "valida()" devuelva un valor true. Se supone que devolverá true cuando todos los datos del formulario sean correctos.

- **Validar un campo como obligatorio**

Antes de que el formulario sea enviado hay que comprobar si el campo está vacío.

```
function validacion() {  
    valor = document.getElementById("campo").value;  
    if( valor.length === 0 ) {  
        alert ("El campo ha de contener un valor") ;  
        return false; }  
    return true; }
```

- **Validar un campo de texto como numérico**

Es habitual utilizar campos de texto que solo admiten números como teléfonos, código postal, DNI (sin letra).

```
function validaNum() {  
    valor = document.getElementById("telefono").value;  
    if(isNaN(valor)) {  
        alert("El campo tiene que ser numérico");  
        return false; }  
    return true; }
```

Las fechas suelen ser datos difíciles de comprobar, ya que hay infinidad de formas de expresarlas.

```
function validaFecha() {  
    var dia = document.getElementById("dia").value;  
    var mes = document.getElementById("mes").value;  
    var ano = document.getElementById("ano").value;  
    fecha = new Date (ano, mes, dia);  
    if( !isNaN(fecha) ) {  
        return false; }  
    return true; }
```

- **Validar un checkbox**

Es habitual encontrar casillas de verificación para aceptar ciertas condiciones que permitan acceder a diversos servicios. Es útil crear una función que compruebe si esa casilla esta seleccionada.

```
function validaCheck() {  
    elemento = document.getElementById ("campoCondiciones" );  
    if( !elemento.checked ){  
        return false; }  
    return true;  
}
```

Actividad: VALIDA

Crea un formulario con dos campos de texto. Los 2 campos no pueden estar vacios y el segundo es numérico. Cuando los datos sean correctos se visualizará una página con el mensaje “Datos válidos”. Crea distintos tipos de validación: campo a campo, los dos campos juntos, mostrar mensajes de error en cada campo, mostrar errores todos juntos al final,...

Opción1: Muestra un alert por cada error cometido. Mientras el primer error no se solucione, no se mostrarán los demás. Los errores se muestran, uno a uno. La validación se hace en el momento de pulsar el botón "Enviar datos". Cuando todos los datos sean correctos se visualizará "resultado.html".

```
<html>  
<head>  
  <title>Ejemplo de formulario</title>  
<script>  
  function valida(){  
    campo1 = document.getElementById("nombre").value;  
    if (campo1.length === 0) {  
      alert ("El campo nombre ha de contener un valor") ;  
      return false;    }  
    campo2 = document.getElementById("edad").value;  
    if (campo2.length === 0) {  
      alert ("El campo edad ha de contener un valor") ;  
      return false;    }  
    if(isNaN(campo2)) {  
      alert("El campo tiene que ser numérico");  
      return false;    }  
    return true; }  
  </script>  
</head>  
<body>  
  <form name="altausuario" action="resultado.html" method="POST" onsubmit="return valida()">  
    Nombre <input type="text" id="nombre"> <br><br><br>  
    Edad <input type="text" id="edad"> <br><br><br>  
    Enviar Datos <input type="submit" name="enviar" value="Enviar datos"/><br><br>  
  </form></body></html>  
  
Resultado.html  
<html>  
<head> </head>  
<body>  
  <h1> Datos correctos </h1>  
</body></html>
```

Opción 2: En lugar de utilizar un alert, se crea una capa debajo de cada campo en donde ir mostrando los errores. A estas capas se les puede dar estilo. Cuando el error deja de existir, debe eliminarse. La validación sigue siendo en conjunto, para los dos campos.

```
<html>
<head>
<title>Ejemplo de formulario</title>
<script>
function valida(){
    campo1 = document.getElementById("nombre").value;
    if (campo1.length === 0) {
        document.getElementById("errornombre").innerHTML = "El campo nombre está vacío" }
    else document.getElementById("errornombre").innerHTML = "";
    campo2 = document.getElementById("edad").value;
    if (campo2.length === 0) {
        document.getElementById("erroredad").innerHTML = "El campo edad está vacío. ";      }
    else document.getElementById("erroredad").innerHTML = "";
    if(isNaN(campo2)) {
        document.getElementById("erroredad").innerHTML += "El campo tiene que ser numérico";      }
    if ((campo1.length === 0) || (campo2.length === 0) || (isNaN(campo2)) ){
        return false; }
    return true; }
</script>
</head>
<body>
<form name="altausuario" action="resultado.html" method="POST" onsubmit="return valida()">
    Nombre <input type="text" id="nombre"> <br>
    <div id="errornombre"></div><br><br>
    Edad <input type="text" id="edad"> <br>
    <div id="erroredad"></div><br><br>
    Enviar Datos <input type="submit" name="enviar" value="Enviar datos"/><br/><br/>
</form>
</body>
</html>
```

Opción 3: Se utilizan capas para mostrar los errores. En este caso las validaciones se hacen campo a campo, eligiendo como evento el "onblur". El problema que tiene es que si el usuario pulsa el botón "enviar" sin haber pulsado sobre las cajas de texto, los datos se enviarán al servidor.

```
<html>
<head>
<title>Ejemplo de formulario</title>
<script>
function validanombre(){
    campo1 = document.getElementById("nombre").value;
    if (campo1.length === 0) {
        document.getElementById("errornombre").innerHTML = "El campo nombre está vacío" ;
        return false;      }
    else { document.getElementById("errornombre").innerHTML = "";
        return true;      }      }
    }
```

```

function validaedad(){
    campo2 = document.getElementById("edad").value;
    if (campo2.length === 0) {
        document.getElementById("erroredad").innerHTML = "El campo edad está vacío. ";
        return false;    }
    else {
        document.getElementById("erroredad").innerHTML = "";
        if(isNaN(campo2)) {
            document.getElementById("erroredad").innerHTML += "El campo tiene que ser numérico";
            return false;    }
        return true    } }
</script>
</head>
<body>
    <form name="altausuario" action="resultado.html" method="POST">
        Nombre <input type="text" id="nombre" onblur="return validanombre()"> <br>
        <div id="errornombre"></div><br><br>
        Edad <input type="text" id="edad" onblur="return validaedad()"> <br>
        <div id="erroredad"></div><br><br>
        Enviar Datos <input type="submit" name="enviar" value="Enviar datos"/><br/><br/>
    </form>
</body>
</html>

```

Actividad: VALIDACIÓN FINAL

Crea un formulario con uno o varios elementos de cada tipo posible (HTML 5), teniendo en cuenta:

- Los elementos indispensables son: cajas de texto, cajas de área de texto, casillas de verificación (checkbox y radio button) y selectores desplegables.
- La función de validación debe comprobar, como mínimo, que hay un valor introducido para cada uno de los campos.
- La función de validación debe incluir alguna expresión regular.
- Se puede elegir entre validar campo a campo o realizar una validación conjunta.
- Es indispensable que para validar cada uno de los campos se utilicen funciones específicas.
- Es indispensable que los errores de cada campo se visualicen en el formulario (con una capa por cada campo o una capa en la que aparezcan todos los errores).
- Cuando un error deje de producirse, es indispensable que desaparezca el mensaje asociado.
- Sería deseable que existiesen validaciones de modo que el valor de uno o varios de los campos, condicionen los valores de otro u otros campos del formulario.

E. EXPRESIONES REGULARES

Las expresiones regulares describen un conjunto de **elementos que siguen un patrón**. Son útiles a la hora de evaluar algunos tipos de datos que normalmente se utilizan en los formularios. *Ejemplo: palabras que comienzan por la letra "a" minúscula.*

Caracteres especiales

La simbología se interpreta del siguiente modo:

- **^ principio de entrada o línea:** las cadenas deberán comenzar por el siguiente carácter.
- **\$ fin de entrada o línea:** la cadena debe terminar por el elemento precedido al dólar.
- *** el carácter anterior 0 o más veces:** el carácter anterior se puede repetir 0 o más veces.
- **+ el carácter anterior 1 o más veces:** el carácter anterior se puede repetir una o más veces.
- **? el carácter anterior una vez como máximo:** el carácter anterior se puede repetir 0 o 1 vez.
- **. cualquier carácter individual:** puede haber cualquier carácter salvo el de salto de línea.
- **x | y x ó y:** la barra vertical indica que puede ser el carácter x o 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:** puede aparecer cualquier carácter que esté incluido en los corchetes. Se pueden 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:** pueden aparecer todos los caracteres que no estén incluidos en los corchetes. También se pueden especificar rangos de caracteres.
- **\b fin de palabra:** tiene que haber un fin de palabra o retorno de carro.
- **\B no fin de palabra:** cualquiera que no sea un límite de palabra.
- **\d cualquier carácter dígito:** puede haber cualquier carácter numérico, de 0 a 9.
- **\D carácter que no es dígito:** puede haber cualquier carácter siempre que no sea numérico.
- **\f salto de página:** tiene que haber un salto de página.
- **\n salto de línea:** tiene que haber un salto de línea.
- **\r retorno de carro:** tiene que haber un retorno de carro.
- **\s cualquier espacio en blanco:** 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:** cualquier carácter individual que no sea un espacio en blanco.
- **\t tabulación** tiene que haber cualquier tabulación.
- **\w carácter alfanumérico:** puede haber cualquier carácter alfanumérico.
- **\W Carácter que no sea alfanumérico:** cualquier carácter que no sea alfanumérico.

Validar un formulario con expresiones regulares

Combinando las expresiones se pueden crear patrones para validar datos en la mayoría de los formularios.

• Validar una dirección de correo electrónico

En primer lugar, se comprueba 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.

```
function valida() {  
    valor = document.getElementById("email").value;  
    if (! (/^\w+([\.|-]?\w+)*@(\w+([\.|-]?\w+)*\.(\w{2,4})+)$/).test(valor)) {  
        alert ("error"); return false;  
    }  
    alert("bien"); return true;  
}
```

- **Validar un DNI**

No se puede asegurar que el DNI sea el de la persona que se está identificando, pero sí se puede asegurar que el DNI que introduce es correcto.

```
function valida() {
    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)) ) {
        alert("error"); return false;
    }
    if (valor.charAt(8) !== letras[(valor.substring(0, 8))%23]) {
        alert("error"); return false;
    }
    alert("bien"); return true;
}
```

- **Validar un número de teléfono**

Al igual que en el caso anterior, no se puede comprobar que el numero pertenezca al usuario que lo introduce, pero sí que el número de teléfono tenga un formato correcto.

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

Se valida que el número este formado por dígitos y tenga una longitud de 9.

Si se quiere que el campo solo corresponda a un número de teléfono móvil, se puede especificar que el número solo pueda comenzar por 6.

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

Existen otros muchos **datos que se pueden validar con expresiones regulares**, como pueden **cuentas bancarias, CIF de empresas** y cualquier dato que siga un patrón. Validar los datos para que el usuario no envíe el formulario si estos no son correctos **aumenta el rendimiento del servidor**, al disminuir las peticiones. Es importante mostrar **información de cómo el usuario tiene que introducir los datos**.

```
<html>
<head>
<title>Ejemplo de formulario</title>
<script>
    function validaEmail() {
        valor = document.getElementById("email").value;
        if (!(/^w+([.-]?w+)*@w+([.-]?w+)*(\.w{2,4})+$/).test(valor)) {
            document.getElementById("errorEmail").innerHTML = "Email incorrecto"; return false;
        }
        document.getElementById("errorEmail").innerHTML = ""; return true;
    }
    function validaDNI() {
        valor = document.getElementById("dni").value;
        var letras = ['T', 'R', 'W', 'A', 'G', 'M', 'Y', 'F', 'P', 'D', 'X', 'B', 'N', 'J', 'Z', 'S', 'Q', 'V', 'H', 'L', 'C', 'K', 'E', 'T'];
        if (!(/^d{8}[A-Z]$/.test(valor)) ) {
            document.getElementById("errorDNI").innerHTML = "DNI incorrecto"; return false;
        }
        if (valor.charAt(8) !== letras[(valor.substring(0, 8))%23]){
            document.getElementById("errorDNI").innerHTML = "DNI incorrecto"; return false;
        }
        document.getElementById("errorDNI").innerHTML = ""; return true;
    }
</script>
```

```

function validaTelefono() {
    valor = document.getElementById("telefono").value;
    if( !(/^d{9}$/.test(valor)) ) {
        document.getElementById("errorTelefono").innerHTML = "Telefono incorrecto"; return false; }
    document.getElementById("errorTelefono").innerHTML = ""; return true; }
</script>
</head>
<body>
    <form name="expresionregular" action="resultado.html" method="POST">
        Mail <input type="text" id="email" onblur="return validaEmail()"> <br>
        <div id="errorEmail"></div><br><br>
        DNI <input type="text" id="dni" onblur="return validaDNI()"> <br>
        <div id="errorDNI"></div><br><br>
        Telefono <input type="text" id="telefono" onblur="return validaTelefono()"> <br>
        <div id="errorTelefono"></div><br><br>
        Enviar Datos <input type="submit" name="enviar" value="Enviar datos"/><br><br>
    </form>
</body>
</html>

```

Actividad: EXPRESIONES REGULARES

Crea un formulario que contenga un campo “codproducto”. Utiliza la expresión regular que te parezca más adecuada para validarlo.

F. COOKIES

Surgieron con el fin de mantener la información de los **carritos de la compra virtuales**. Por medio de las cookies el usuario podía navegar entre las páginas sin perder los elementos que había seleccionado, pudiendo modificar esta cesta virtual en cualquier momento.

Con la evolución de las páginas web surge la necesidad de **identificar al usuario** que realiza la petición al servidor. En una página se puede identificar al usuario mediante un nombre y contraseña pero, en la siguiente, ¿cómo sabe el servidor que envía los datos al cliente que se identificó? Podría realizar una petición de esa URL otro navegador y visualizar sin ningún tipo de validación, la interfaz de entrada del correo electrónico. Para resolver este problema nacieron las cookies.

Una cookie es un **fichero de texto** que se almacena en el ordenador del cliente. Cada navegador especifica una ruta en la que guarda sus cookies. El servidor solicita al navegador que guarde las cookies. En las sucesivas peticiones del navegador al servidor, el navegador envía la petición junto con la cookie. Así, el servidor sabe quién es el cliente que realiza la petición.

Uso

Su principal objetivo es que el servidor pueda conocer algunas características del usuario, para así poder tomar una decisión con respecto a la respuesta.

Al navegar por páginas que no ofrecen confianza puede realizarse una **vulnerabilidad** en el sistema, ya que el servidor podría acceder al disco para fines a los que no se le ha autorizado. En estos casos puede limitarse o anular el uso de las cookies en el navegador.

Son utilizadas en ocasiones para mantener **preferencias de visualización**.

Se pueden emplear para **almacenar variables** que se necesiten utilizar en el navegador. Por ejemplo, en una página se solicitan unos datos, en la siguiente solicitan otros y así hasta la página final. Antes del envío al servidor se recuperan los campos del formulario guardados en las cookies.

En ocasiones los servidores hacen uso de ellas para almacenar **preferencias y hábitos del usuario al navegar**. Le permite al servidor personalizar sus servicios y publicidad orientándolo a cada cliente en particular.

Uno de los usos más habituales es **autenticar a los usuarios**.

Las cookies tienen **caducidad**, cuando pasa un período de tiempo establecido, desaparecen junto con el fichero de texto que guarda el navegador. En el caso de la autenticación caducan cuando finaliza la navegación por la página, cerrando la ventana. Habitualmente, como mecanismo de seguridad, las aplicaciones web aplican un tiempo máximo de inactividad tras el cual las cookies caducan si no se produjo movimiento en la navegación de la aplicación web.

Implementación

Los dos procesos de implementación principales de una cookie son la escritura y la lectura de la misma.

Las **aplicaciones web que hagan uso de las cookies requieren que éstas estén activadas** en el navegador. De lo contrario, por ejemplo, si se usan para identificarse en la aplicación, el usuario no podría acceder a la misma.

El código siguiente funciona en los navegadores Explorer y Firefox pero no en Chrome.

```
<html>
<head>
<script>
    function RecuperaCookie(nombre) {
        var i,x,y,ARRcookies=document.cookie.split("; ");
        //después del ; hay un espacio en blanco ya que la segunda cookie se guarda así.
        for (i=0;i<ARRcookies.length;i++){
            x=ARRcookies[i].substr(0,ARRcookies[i].indexOf("="));
            y=ARRcookies[i].substr(ARRcookies[i].indexOf("=")+1);
            if (x == nombre) {
                return unescape(y); } } }
    function GuardarCookie (nombre, valor, dias) {
        var fecha=new Date();
        fecha.setDate(fecha.getDate() + dias);
        var dato = escape(valor) + ( (dias==null) ? "" : ";expires = " + fecha.toUTCString());
        document.cookie = nombre + "=" + dato; }
```

```

function ComprobarCookie() {
    var usuario = RecuperaCookie("usuario");
    if (usuario != null && usuario != "") {
        alert("Bienvenido " + usuario);
    } else {
        usuario = prompt("Por favor, introduzca tu nombre:", "");
        if (usuario != null && usuario != "") {
            GuardarCookie ("usuario", usuario, 365); } } }
</script>
</head>
<body>
    <input type="button" name="guardaCookie" value="Comprobar cookie" onclick="ComprobarCookie();">
</body></html>

```

Actividad: GALLETAS

Crea un formulario con una caja de texto y botón que almacene su valor como una cookie. Utiliza otro botón para recuperar la cookie almacenada.

BIBLIOGRAFÍA

Eventos

http://www.w3schools.com/jsref/dom_obj_event.asp