



Ministerio de Educación, Cultura y Deporte.

# HTML5 en la educación

Módulo 12: Formularios y *JavaScript*.

## Formularios y JavaScript

### Interactuar con el usuario



Cuando se complementa HTML con otros lenguajes, conseguimos crear verdaderas aplicaciones interactivas que se ejecutan a través del navegador. Las páginas web HTML o las acciones que se generan mediante *JavaScript* conforman recursos que se ejecutan del lado del navegador, es decir, en nuestro ordenador. Este proceso se puede complementar con un trabajo que se ejecuta en el lado del servidor, mediante lenguajes como *PHP*, *ASP*, etc. Así el servidor es capaz de crear páginas sobre la marcha, que se transmiten al usuario.

En este punto nos centraremos, exclusivamente, en lo que sucede en el lado del navegador en nuestro propio equipo.



### Programación

#### Objetivos específicos

- Crear formularios.
- Conocer los diferentes campos de un formulario.
- Gestionar eventos dentro de una página web.

#### Contenidos

- Formularios.
- Campos de un formulario.
- Eventos.
- Creación de *scripts* con *JavaScript*.

#### Criterios de evaluación

- Diseñar formularios.
- Controlar eventos mediante *scripts*.



### Requisitos mínimos

- Conocimientos sobre HTML.
- Conocimientos sobre navegadores web.
- Conocimientos de procedimientos en el ordenador: seleccionar, cortar y pegar.

## Recurso TIC: Formularios y JavaScript

Seguiremos avanzando en nuestro conocimiento de las propiedades más empleadas de las hojas de estilo, centrándonos ahora en aquellas que conllevan posicionamiento de objetos, incluyendo algunas que definen su apariencia, pero más en un ámbito espacial.

### Formularios

Las páginas más completas y más complejas suelen contar con algún grado de interactividad. Esa interactividad comienza por recoger alguna operación que realiza el usuario y responderle de algún modo. El ejemplo más claro de esto es el uso de formularios, para realizar una recogida de datos que el navegador mandará por correo electrónico o que se almacenarán en una base de datos en el servidor, para que a su vez puedan ser procesados.

Para poder gestionar la información, remitida mediante el servidor y una base de datos, necesitaremos conocimientos avanzados de otras tecnologías que se ejecutan en el lado del servidor, lo que excede a los propósitos de este material. No obstante, sí podremos avanzar en el diseño de formularios, para enviar información por correo o para realizar algunas interacciones mediante *JavaScript*.

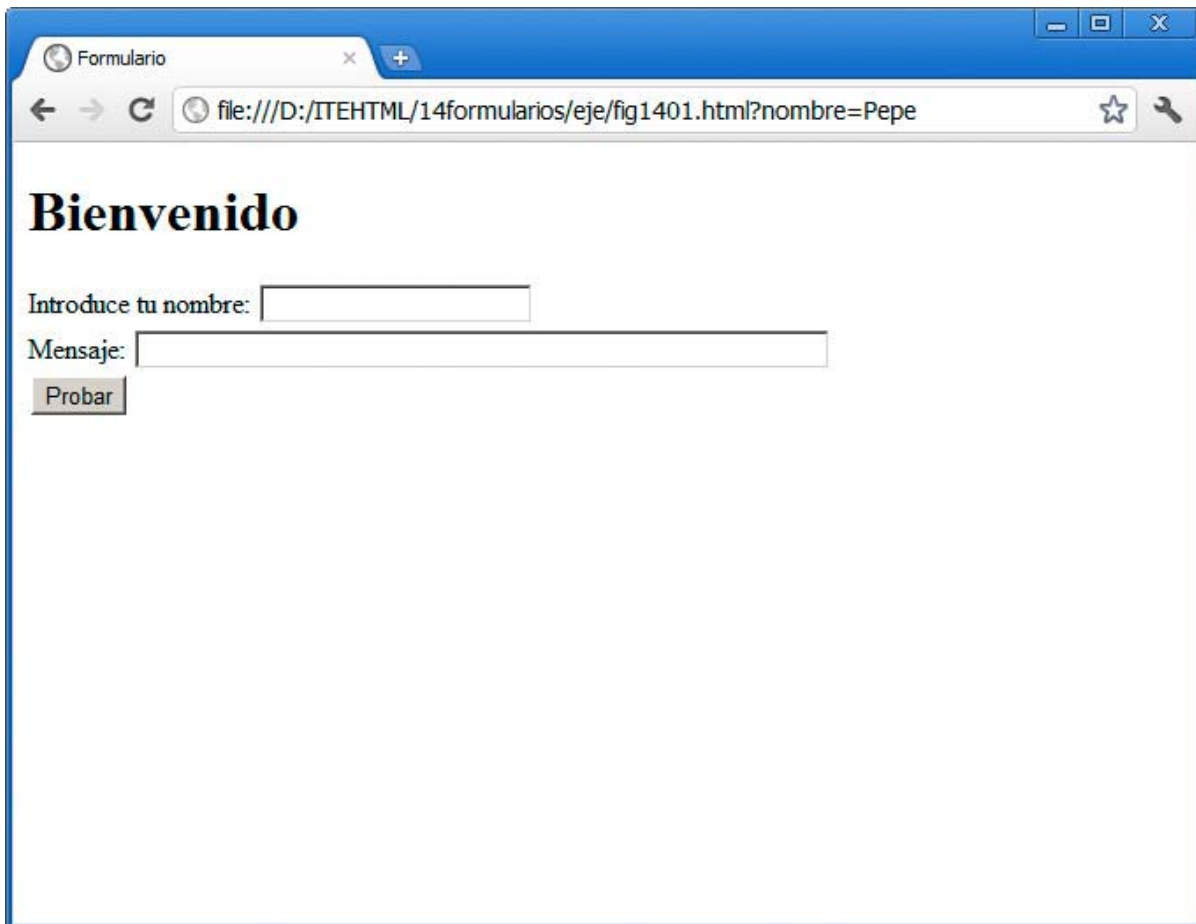
### Insertar un formulario

La creación de un formulario se realiza mediante la etiqueta **<form>** y su correspondiente **</form>**. Así generamos un bloque dentro del cual podemos introducir todos los contenidos que queramos: tablas, imágenes, etc. También incluiremos en su interior diferentes elementos HTML orientados a esa recogida de datos.

Probemos con un ejemplo elemental. Incorporaremos este texto en el **<body>** de una página vacía:

```
<form name="miformulario">
  <h1>Bienvenido</h1>
  Introduce tu nombre: <input type="text" name="nombre" />
  <br />Mensaje: <input name="mensaje" type="textarea" size="60" />
  <br /><input type="submit" value="Probar" />
</form>
```

Al probar esta página, se obtiene el resultado de la figura:



El formulario incluye un sencillo título y dos elementos de formulario: un cuadro de texto y un botón de envío.

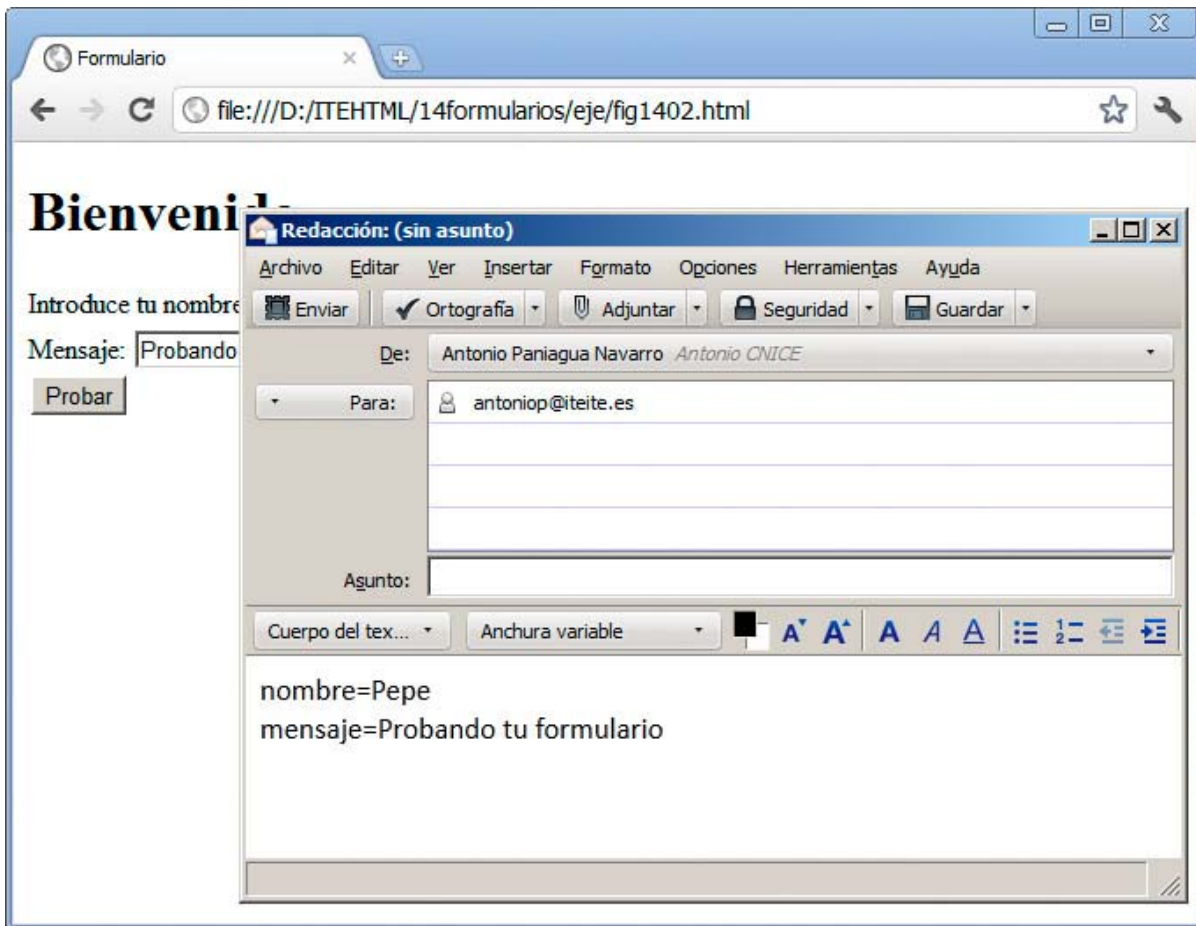
## Opciones de formulario

Nuestro formulario no tiene programada ninguna acción. Cuando se pulsa el botón **Probar**, no sucede nada. Se puede modificar el comportamiento del formulario y algunas opciones más mediante parámetros.

El parámetro **action** se acompaña de una URL que indica a dónde se saltará cuando el usuario pulse en el botón de envío (el botón de tipo **submit**, como veremos más tarde). Un formulario como el nuestro se podría remitir por correo mediante una URL del tipo **mailto** así:

```
<form name="miformulario" action="mailto:antoniop@iteite.es" method="post" enctype="text/plain">
```

y obtendríamos el resultado de la figura, donde se despliega la aplicación de correo que el usuario tiene configurada para remitirnos nuestro formulario por correo electrónico:



Esta opción de remisión de formularios por correo se emplea poco en la actualidad; obliga al usuario a tener configurada una aplicación de correo. Lo normal aquí sería ejecutar algún *script* en el lado del servidor que procesase el formulario, lo que, como decíamos anteriormente, no será tratado en este módulo.

Junto a **action** cada formulario suele tener otros tres parámetros:

- **name**: especifica un nombre único para el formulario. Es muy importante al ser procesado en el servidor, ya que permite discriminar entre varios formularios, si la página los tuviese.
- **method**: indica el formato de envío que se va a emplear para el formulario. Tenemos la opción **post** que remite la información de forma oculta y sin limitaciones de tamaño y, por otro lado, la opción **get**, que transmite los datos como parte de la URL de la página. El método **post** suele ser el más recomendable y es el que emplearemos en casi todos los casos.
- **enctype**: se usa para indicar el formato de codificación de los datos que estamos remitiendo. El valor predeterminado **application/x-www-form-urlencoded**. **text/plain** hace que se reemplacen los espacios por el signo **+**; se emplea para textos sencillos. Si vamos a enviar archivos, fotos, etc. usaremos la codificación **multipart/form-data** donde ningún carácter se codifica.

Aún hay algunos valores más, como **autocomplete** (con valores **on** u **off**) que decide si al rellenar el formulario se nos harán sugerencias (que es el comportamiento predeterminado) o no. Ponerlo en **off** es útil para formularios en los que se introducen datos personales. También podemos activar el valor **novalidate**, para que el formulario no se compruebe antes de enviarlo.

## ? Pregunta de Elección Múltiple

El parámetro *action* en los formularios suele ir acompañado otros tres: *name*, *method* y *enctype*.

- Estos parámetros sirven para especificar un nombre único para cada formulario.
- Sirven para indicar el formato de envío que se va a emplear para el formulario.
- Sirven para indicar el formato de codificación de los datos que estamos remitiendo.
- Todas las anteriores son falsas.



## Actividad 1

Generaremos un pequeño formulario para ir probando los diferentes tipos de campos que se pueden añadir. En el parámetro **action** optaremos por una acción de envío por correo electrónico, para así poder comprobar los valores que se mandan en cada correo.

## Campos de un formulario

Un formulario puede contener cualquier tipo de recurso, como ya hemos dicho antes, pero hay algunos campos que son específicos del formulario y cuyo valor se enviará cuando éste se remita.

### Cuadro de texto

Un cuadro de texto es un espacio definido para que el usuario introduzca un texto libre. Se crea con:

Introduce tus apellidos: `<input type="text" name="apellidos" />`

Como todos los elementos de un formulario, además del tipo se establece un nombre único mediante el parámetro **name**. Ese valor nos permite manipular el valor mediante *scripts* del lado del servidor o utilizando *JavaScript*.



### Nota

La etiqueta **<input>** cuenta con un copioso número de parámetros. No todos son aplicables a todas las etiquetas **input**; depende del tipo que indiquemos en el valor **type**. Más tarde veremos algunos valores.

### Cuadro de contraseña

Los campos de contraseña son iguales que los de texto, pero al escribir dentro de ellos sólo se ven asteriscos o algún carácter similar, para evitar que otras personas alrededor puedan ver la clave.

Introduce tu clave: `<input type="password" name="clave" />`

### Botones radiales

El usuario puede seleccionar uno de los valores del bloque, pero sólo uno de ellos; son excluyentes.

Selecciona el color de tu icono:  
`<br />`  
`<input type="radio" name="coloricono" value="rojo" /> Rojo<br />`  
`<input type="radio" name="coloricono" value="verde" /> Verde <br />`  
`<input type="radio" name="coloricono" value="azul" /> Azul <br />`

Dos detalles fundamentales para que este conjunto funcione:

- El valor indicado en **name** tiene que ser el mismo en todas las opciones; así es como el navegador sabrá qué elemento del conjunto está seleccionado.

- Se incorpora un parámetro nuevo, **value**. Cuando el formulario se envíe para ser procesado, el campo *coloricono* de nuestro ejemplo tomará el valor indicado en **value**. Este valor no tiene por qué coincidir con el texto que mostramos a continuación.

El conjunto se ve así, tras seleccionar la tercera opción:



Formulario

file:///D:/ITEHTML/14formularios/eje/fig1403.html

# Bienvenido

Introduce tu nombre:

Mensaje:

Selecciona el color de tu icono:

Rojo

Verde

Azul

## Casillas de verificación

En este tipo (denominado **checkbox**) el usuario sí puede seleccionar varias opciones. Todas ellas se remitirán para ser procesadas, asociadas bajo el mismo nombre, indicado en el parámetro **name**:

```
<hr />Selecciona los días con disponibilidad:<br />
<input type="checkbox" name="dias" value="l" /> Lunes<br />
<input type="checkbox" name="dias" value="m" /> Martes<br />
<input type="checkbox" name="dias" value="mm" /> Miércoles<br />
<input type="checkbox" name="dias" value="j" /> Jueves<br />
<input type="checkbox" name="dias" value="v" /> Viernes<br />
```

Bajo el valor *dias* se transmitirá a la página web encargada de procesar el resultado del formulario una secuencia con todos aquellos cuadros que hayan sido activados. Se transmitirá el valor que aparece dentro de cada **value**, únicamente. Por ejemplo, si enviamos el formulario de la figura, se mandará la secuencia *l, m y j*.

Formulario

file:///D:/ITEHTML/14formularios/eje/fig1404.html

# Bienvenido

Introduce tu nombre:

Mensaje:

Selecciona el color de tu icono:

Rojo

Verde

Azul

---

Selecciona los días con disponibilidad:

Lunes

Martes

Miércoles

Jueves

Viernes



## Nota

En el ejemplo hemos reemplazado los caracteres con tilde por entidades html. Por ejemplo, la secuencia **&iacute;** se reemplazará por una "í". Es apropiado hacer estos cambios para asegurarnos de que nuestra página se muestra siempre correctamente. Los editores web suelen hacerlo por nosotros. No obstante, si tenemos que hacerlo, podemos encontrar una lista de entidad html en varios sitios, por ejemplo en [http://www.w3schools.com/tags/ref\\_entities.asp](http://www.w3schools.com/tags/ref_entities.asp)

## Botón de envío

Un campo de tipo **submit** generará un botón que, al ser pulsado, hará que se aplique la acción indicada en el formulario:

```
<input type="submit" value="Haz clic para enviar la informac&iacute;on" />
```

## Botón de limpieza

Un campo de tipo **reset** eliminará todo el contenido introducido en un formulario, dejándolo limpio de nuevo. La figura muestra el botón anterior y uno de tipo **reset**.

```
<input type="reset" value="Comenzar a rellenar de nuevo" />
```



## Botón estándar

También podemos crear un botón que no sea del tipo **submit** o **reset**, sino genérico, mediante el tipo **button**.

```
<input type="button" value="Haz clic sobre mi" />
```

Ese botón tal y como está no hace absolutamente nada. Podemos pasarnos el día entero pulsándolo sin conseguir ningún resultado. Para que esos botones interactúen tenemos que definir en ellos el evento **onclick**. Así podremos asociarle una función *JavaScript*, cargar una página en el lado del servidor o simplemente remitir una información por correo electrónico.

```
<input type="button" value="Haz clic sobre mi" onclick="JavaScript:window.location='http://www.google.es'" />
```

Este botón sí realiza ya una operación. Al hacer clic sobre él, se ejecuta una pequeña acción de *JavaScript* (**window.location='http://www.google.es'**), que hace que el navegador salte a la página indicada.



### Nota

Más tarde analizaremos los eventos y su funcionamiento.

## Archivo

Un elemento **input** de tipo **file** nos sirve para enviar un archivo al servidor. Se emplea con frecuencia para enviar alguna imagen, documentos de texto, etc. Podemos remitir cualquier tipo de archivo.

```
Adjunta tu curriculum vitae <input type="file" name="curriculum" />
```

El resultado es éste:

## Valores ocultos

El tipo **hidden** se emplea para remitir información al servidor dentro de los formularios, pero sin que esta información se muestre en la pantalla. Su sentido es el de proporcionar datos complementarios, como podría ser la fecha, desde dónde se ha rellenado el formulario o cualquier otro aspecto que no necesitamos que rellene el usuario, sino que podemos definir por otros medios. Todo lo que podamos evitar que sea rellenado aligerará el formulario y favorecerá la interacción del usuario con nuestra página.

```
<input type="hidden" name="campaña" value="primavera" />
```

## Más tipos

La aparición de HTML5 nos trae varios tipos de campo **<input>** añadidos. En concreto las nuevas opciones son las siguientes: **color**, **date**, **datetime**, **datetime-local**, **month**, **week**, **time**, **email**, **number**, **range**, **search**, **tel** y **url**.

En la actualidad casi todos ellos se muestran como cuadros de texto normales pero, a medida que se implante el estándar, iremos viendo cómo se producen cambios. Por ejemplo, si definimos un campo indicando que será de tipo **tel** (teléfono) y accedemos mediante un dispositivo móvil, el teclado se cambiará automáticamente al numérico; o si definimos un campo de tipo **email** (correo), el navegador será capaz de realizar una validación previa para comprobar si es una dirección correcta.

En conclusión, puede ser interesante ir conociéndolos e incorporándolos a nuestra rutina, al diseñar formularios.

## Otros campos diferentes a input

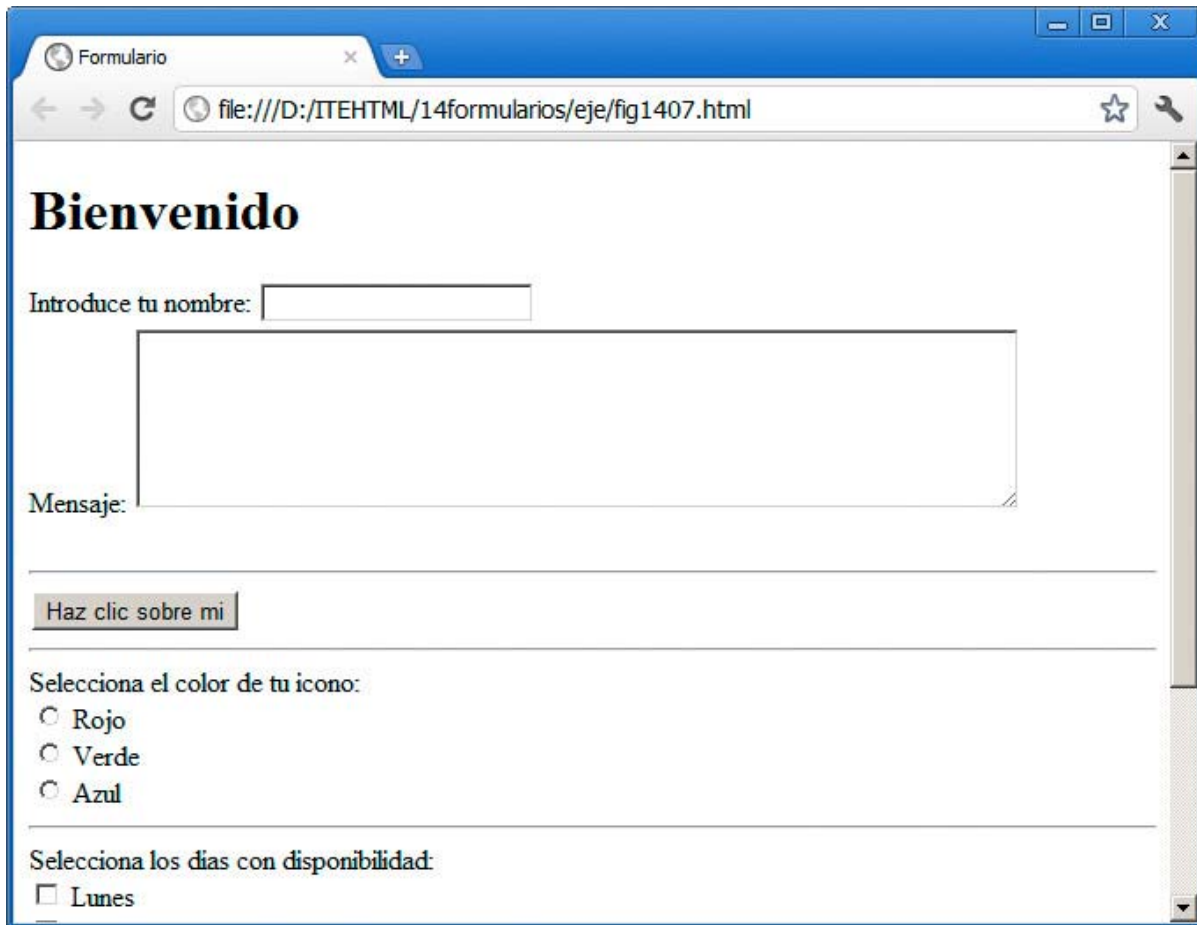
En los formularios emplearemos otros recursos diferentes a los que nos facilita la etiqueta **input**.

### Área de texto

Un área de texto es similar a los cuadros creados con **<input type="text">** pero más amplios, pensados para albergar secuencias de texto tan largas como sea preciso. Se crean mediante la etiqueta **<textarea>** y en ellas podemos indicar el tamaño visual del campo mediante los parámetros **cols** (columnas) y **rows** (filas).

```
<textarea name="mensaje" cols="60" rows="6" /> </textarea>
```

La figura muestra la etiqueta anterior:



Como se puede observar, esta etiqueta lleva su cierre correspondiente. Si colocamos cualquier texto entre ambas etiquetas, se mostrará dentro del cuadro al cargarse la página.

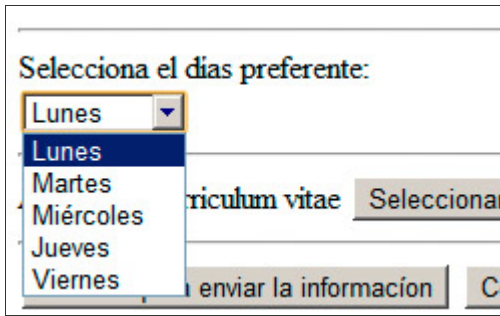
## Listas desplegables

La etiqueta `<select>` y `</select>` se utiliza para crear una lista desplegable de opciones, donde el usuario puede seleccionar una de ellas. Su sentido es similar al de los botones radiales, pero en un espacio más reducido, sobre todo si tenemos muchas opciones. Tiene también la ventaja de que con el teclado podemos desplazarnos rápidamente hasta la opción que comience con la letra que pulsamos, algo muy útil en las listas muy largas.

Cada opción de una lista desplegable se engloba en la etiqueta `<option>`, por lo que el conjunto quedaría así:

```
Selecciona el día preferente:<br />
<select>
  <option value="l"> Lunes</option>
  <option value="m">Martes</option>
  <option value="mm">Miércoles</option>
  <option value="j">Jueves</option>
  <option value="v">Viernes</option>
</select>
```

Al probarlo, obtendremos una lista como la de la figura:



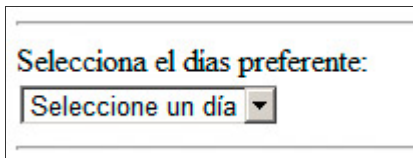
Si queremos que de forma predeterminada el valor seleccionado sea diferente del primero, podemos añadir el parámetro **selected** al elemento. Así:

```
<option value="j" selected>Jueves</option>
```

También es habitual añadir un elemento en blanco al principio o con un texto animando a realizar la selección:

```
<option value=""> Seleccione un día</option>
```

El cambio se vería de esta manera:



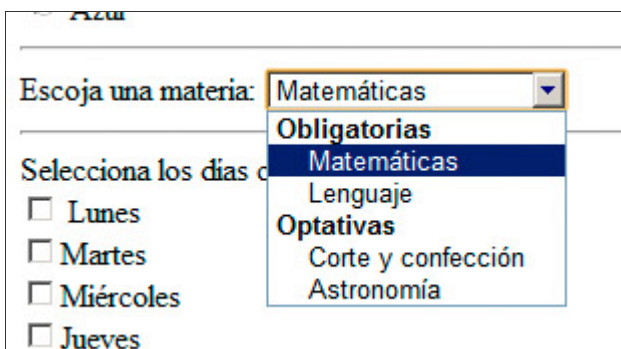
Al no tener valor asociado, podríamos saber, al tratar el formulario, que el usuario no ha escogido ningún día.

## Grupos de opciones

La etiqueta **optgroup** nos proporciona una variante de los campos de tipo **option**. Su diferencia es que podemos agrupar las opciones con subtítulos. Observemos este ejemplo:

```
<hr />
Escoja una materia:
<select>
  <optgroup label="Obligatorias">
    <option value="mat">Matemáticas</option>
    <option value="len">Lenguaje</option>
  </optgroup>
  <optgroup label="Optativas">
    <option value="cor">Corte y confección</option>
    <option value="ast">Astronomía</option>
  </optgroup>
</select>
```

Se mostraría así:



## Elementos complementarios

En formularios largos conviene agrupar los grandes bloques de elementos mediante la etiqueta **<fieldset>**. Es una división visual marcada por una simple línea. Si empleamos la etiqueta **<legend>** podremos además añadir un nombre a ese subconjunto. Observemos este ejemplo y cómo se mostraría en el navegador:

```
<fieldset>

  <legend>Datos personales</legend>
  Introduce tu nombre: <input name="nombre" type="text" /> <br />
  Mensaje: <textarea name="mensaje" cols="60" rows="6"></textarea> <br />

</fieldset>
```



Esta división es meramente visual y no afecta de ninguna manera al comportamiento del formulario, aunque sí le aporta un sentido semántico.

Además, podemos utilizar la etiqueta **<label>** para establecer un nombre para un campo. Esta etiqueta no se muestra, por lo que aplicarla vuelve a ser una medida apropiada para favorecer la accesibilidad del formulario e incluso para aplicar estilos, pero no tendrá efecto en el formulario.

Se utiliza junto al parámetro **for**, así:

```
<label for="nombre">Nombre</label>
Introduce tu nombre: <input id="nombre" name="nombre" type="text"> <br>
```

Su valor debe ser el mismo que el valor **id** del campo, por lo que nos obligamos a añadir un valor **id** a cada elemento.

## Parámetros comunes

La mayoría de los elementos de un formulario cuentan con algunos parámetros comunes muy útiles:

- **size**: en los cuadros de texto se emplea para definir el tamaño del campo. Por ejemplo **size="20"**.

- **maxlength**: también en los cuadros de texto nos servirá para limitar el tamaño. Por ejemplo: `<input type="text" name="telefono" maxlength="9" size="9" />`
- **readonly**: en los cuadros de texto hace que el valor sea de sólo lectura, que no se pueda cambiar. Por ejemplo: `<input type="text" name="pais" readonly="readonly" />España`
- **disabled**: hace que el elemento se encuentre desactivado y no se pueda modificar. Sirve con varios elementos diferentes. Se añade así: **disabled="disabled"**
- **placeholder**: permite añadir un texto dentro del cuadro, que desaparece automáticamente al hacer clic sobre él. Son muy útiles para incorporar pequeñas indicaciones sobre el valor que se espera. Se añade así: **placeholder="Introduce tu nombre"**
- **tabindex**: es habitual desplazarse por un formulario presionando la tecla **Tab** para avanzar al campo siguiente o **Mayús-Tab** para ir al anterior. Mediante **tabindex** podemos modificar el orden de esos saltos entre los campos de nuestro formulario. Iremos numerando cada campo mediante un número, así : `<input type="password" name="clave" tabindex="1" />`

Con todo esto tenemos todos los recursos necesarios para diseñar los formularios más completos. Sólo nos faltará aplicar los estilos adecuados para que el formulario se integre correctamente con nuestro sitio.



## Pregunta Verdadero-Falso

Las siguientes afirmaciones, ¿son verdaderas o falsas?

Para que los botones radiales funcionen hay que tener en cuenta que el valor introducido en *name* tiene que ser el mismo en todas las opciones, además hay que escribir el parámetro *input*.

Verdadero  Falso

El parámetro *checkbox* sirve para el usuario pueda seleccionar varias opciones de una lista dada.

Verdadero  Falso

El parámetro *reset* eliminará sólo el contenido de los botones radiales, dejándolo limpio de nuevo.

Verdadero  Falso

La etiqueta `<select>` y `</select>` se utiliza para crear una lista desplegable de opciones, donde el usuario puede seleccionar una de ellas.

Verdadero  Falso



## Nota

Con los estilos podemos modificar cualquier valor de los campos. Podemos cambiar los colores de los botones, el fondo de los campos, etc.



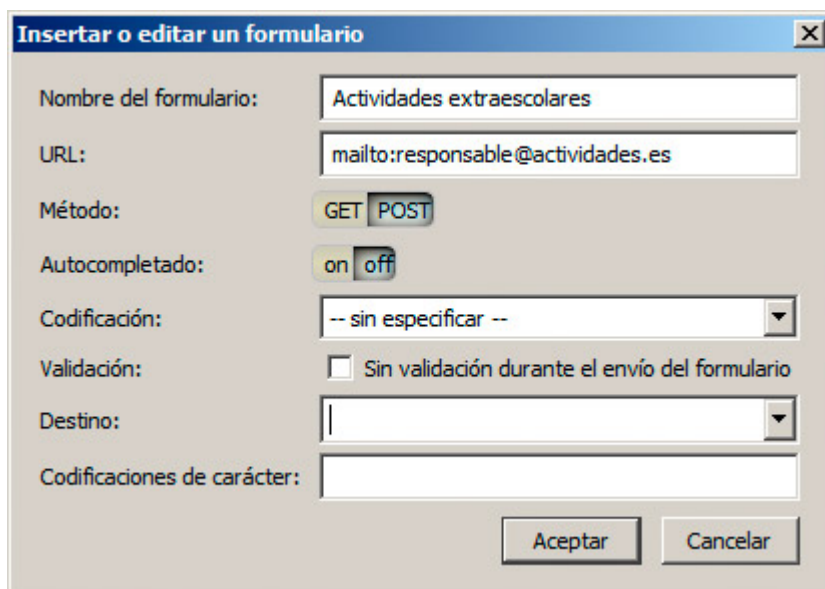
## Actividad 2

Incorporaremos a nuestro formulario diferentes tipos de campos. Intentaremos que sean lo más variados posibles.

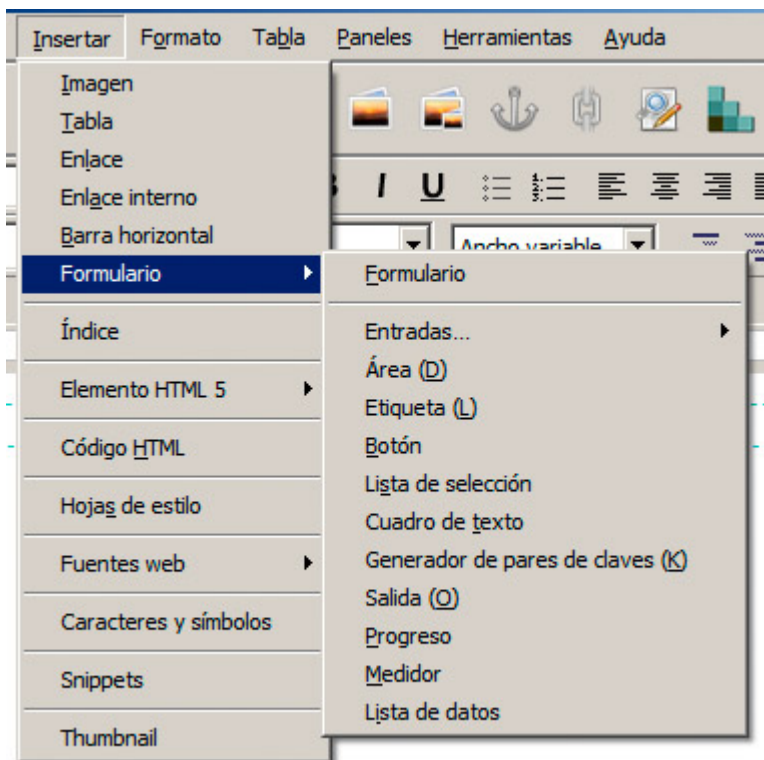
## Insertar un formulario con un editor web

Una vez más los editores web vienen a nuestro rescate, permitiéndonos mantener un conocimiento más relajado de toda la sintaxis de un formulario. Podremos insertar todo tipo de campos con facilidad. En *BlueGriffon* seguiríamos un procedimiento similar a éste:

1. Abrimos la aplicación y creamos un documento nuevo.
2. En el menú **Insertar** seleccionamos la opción **Formulario>Formulario**.
3. En la ventana de la figura estableceremos los datos básicos del formulario y hacemos clic en **Aceptar**.



4. El formulario se muestra delimitado por una línea azul. A partir de aquí podemos crear contenidos dentro del formulario o fuera. Incluso podemos crear más formularios dentro de la página.
5. Para insertar un campo, haremos clic en el interior de la zona azul y seleccionaremos una de las opciones del menú **Insertar>Formulario**, que se recogen en la figura:



6. En la categoría **Entradas** encontramos todos los elementos denominados **<input>**, así como otros que se están incorporando en la actualidad en el desarrollo de HTML5. En todos los casos, al seleccionar un tipo, aparecerá una ventana desde la que estableceremos los datos básicos del campo.

Es así de sencillo; luego sólo debemos ir incorporando los campos necesarios, intercalándolos con las etiquetas que aclaren su funcionamiento.

El siguiente vídeo reproduce la creación de un formulario.



### Vídeo

Creación de un formulario



### Nota

Algunos otros elementos como **<progress>** y **<meter>** no están directamente vinculados con los formularios, aunque el autor de la aplicación ha estimado que es un buen lugar para ubicarlos. Estos dos elementos, en concreto, se emplean para realizar barras de progreso y de medición.



### Actividad 3

Diseñaremos un formulario con un editor web que incluya al menos 6 tipos de campos diferentes. Para hacer las pruebas, podemos crearlo con una acción de tipo **mailto**. Así podremos comprobar qué resultados se envían al componer el correo electrónico.



## Eventos

Con independencia del tratamiento que realicemos con un formulario, que como venimos diciendo requiere en ocasiones que se complemente con otras tecnologías diferentes a las que estamos aprendiendo, podemos profundizar en la interactividad de las páginas web gracias a los eventos.

Un **evento** es algo que sucede en el navegador, de tal manera que, cuando ocurren, podemos definir algún comportamiento concreto, una respuesta por parte de la página web, mediante nuestro medio o utilizando esas otras tecnologías del lado del servidor.

Veamos un ejemplo sencillo que nos ayudará a entender este concepto:

### Carga y descarga de la página

Cada vez que el navegador termina de cargar una página, le remite al documento una indicación de que se ha producido un evento que se llama **onload**. De este modo, desde la página web podemos saber en qué preciso momento se ha terminado de cargar la página completamente y podemos realizar alguna acción. Para realizar estas acciones podemos, por ejemplo, utilizar el lenguaje *JavaScript*, con el que ya hemos realizado un par de prácticas.

Observa la línea siguiente:

```
<body onload="JavaScript:alert('Perfecto. La página ya se ha cargado.');">
```

**alert** es una función del lenguaje *JavaScript* que se utiliza para mostrar un mensaje en la ventana del navegador, como una ventana emergente. Se suele acompañar de un pequeño texto, que será el que se muestre. Ésta es una función muy utilizada, porque permite averiguar los mensajes que se están generando dentro de la página web.

Como se puede observar, indicamos que es una instrucción de *JavaScript* poniendo primero ese término seguido de un signo de dos puntos.

Al probar la página web y terminar de cargarse, obtendremos el siguiente resultado:



Todas las funciones de *JavaScript* (y de otros muchos lenguajes de programación) siguen una estructura parecida: primero se indica el nombre de la función y a continuación, entre paréntesis, los diferentes parámetros que la función necesita para funcionar. Además las líneas en *JavaScript* finalizan con un signo de punto y coma, lo que le indica a la parte encargada de interpretar las órdenes que ahí ha finalizado la instrucción que le estamos dando.

---

Además de **onload** tenemos otros controles que se lanzan antes de que se cargue la página (**onbeforeload**), cuando se abandona la página (**onunload**).

## Más eventos de body

No se acaban ahí las opciones; tenemos eventos para controlar aspectos como el momento en que se cambia el tamaño de una ventana (**onresize**), el momento en que se imprime una página web (**onprint**) o el momento antes de comenzar la impresión (**onbeforeprint**) y otros muchos más.

La conclusión es que empleando *JavaScript*, *Ajax* o tecnologías del lado del servidor podemos responder perfectamente ante cualquier operación que se realice con la página web en general.

## Enfoque y desenfoque

En el ámbito informático, en sistemas como una página web donde varios elementos se presentan ante el usuario, se emplea el concepto de **foco** para determinar qué elemento, de los disponibles en la pantalla, es el que tiene la atención en cada momento. Así, si por ejemplo pulsamos una tecla en el teclado, afectará al elemento que tenga el foco. Cuando pasamos a un segundo elemento, haciendo clic con el ratón, el primero deja de tener el foco y lo obtiene a su vez el segundo.

Contamos con dos eventos para realizar operaciones cuando un elemento obtiene el foco (**onfocus**) y para cuando lo pierde (**onblur**). Suelen resultar bastante útiles al trabajar con formularios.

Observemos el ejemplo de la figura. El estado inicial de los campos es similar en todo caso, pero cuando el usuario hace clic en el campo *Nombre*, éste se vuelve de color rojo. Cuando el usuario pasa a otro campo (cuando nombre pierde el foco), vuelve a su color habitual:



Para conseguirlo de nuevo, debemos recurrir a técnicas basadas en *JavaScript*. En esta ocasión hemos optado por incorporar el código necesario en la cabecera de la página, entre `<head>` y `</head>`.

```
<head>

[...]
```

```
<script language="JavaScript">

    function enfocado(id_objeto) {
        document.getElementById(id_objeto).style.border="2px solid red";
    }

    function desenfocado(id_objeto) {
        document.getElementById(id_objeto).style.border="inherit";
    }

</script>

</head>
```

Es un código muy sencillo que ya hemos empleado además en otra ocasión. Definimos dos funciones que recibirán el `id` de un elemento HTML y modificarán el borde en cada caso. En el primero pondrá un borde rojo de 2 píxeles y en el segundo regresará a su valor original.

Las estructuras de las funciones en *JavaScript* son siempre muy similares a éstas. Se indica el nombre de la función, los parámetros que va a utilizar y posteriormente, entre llaves, se indican las acciones a realizar. En nuestro ejemplo cada función tiene una única opción, pero normalmente tendrán muchas más.

Para que funcione, modificaremos el elemento HTML dejándolo así:

```
<fieldset>

    <legend>Datos personales</legend>
```

```
Introduce tu nombre: <input name="nombre" id="nombre" type="text"
onfocus="JavaScript:enfocado(this.id);" onblur="JavaScript:desenfocado (this.id);"> <br />
Mensaje: <textarea name="mensaje" cols="60" rows="6"></textarea> <br />

</fieldset>
```

Como se puede observar, sólo hemos cambiado la línea del nombre respecto a los ejemplos anteriores. Hemos añadido tres cosas:

- Un **id** denominado **nombre**.
- Un evento **onfocus** que ejecutará la función **enfocado** que hemos definido en la cabecera. **this.id** es el valor que le enviamos a la función y equivaldría a "nombre" en este ejemplo.
- Un evento **onblur** que ejecutará la otra función.

Cuando probemos la página todo funcionará correctamente. De hecho, añadiendo estas tres últimas modificaciones a cualquier campo del formulario, conseguiremos que tengan ese cambio de color. Las funciones se pueden reutilizar tantas veces como sea necesario.



## Nota

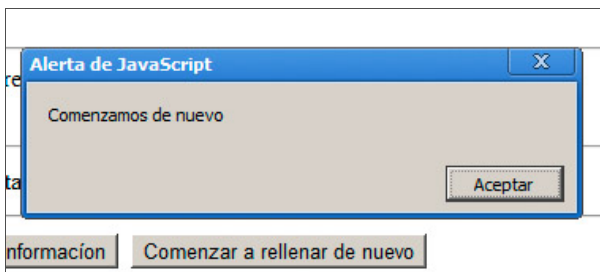
Hay formas más sencillas y directas para conseguir este mismo resultado, pero resultan menos intuitivas para nuestro aprendizaje. Cuanto más profundicemos en el uso de *JavaScript*, más sencillo se irá volviendo su aplicación.

Los formularios cuentan con otros muchos eventos, como cuando alguien pulsa el botón de envío del formulario (**onsubmit**), cuando alguien hace clic en el botón de reiniciarlo (**onreset**), cuando se introduce información en un cuadro (**oninput**) o cuando un valor cambia (**onchange**).



## Actividad 4

Probaremos a realizar una pequeña gestión de eventos, de tal modo que, cuando un usuario haga clic en el botón de limpieza del formulario, se le muestre un aviso con cualquier texto, como se recoge en la figura. Ya hemos visto todo lo necesario para conseguirlo. Como pista, la única parte que debemos deducir es dónde ubicaremos la gestión del evento, si en el botón de **reset** o en la etiqueta del formulario. Al ser un evento que afecta a todo el formulario, debe ser en la etiqueta de formulario, pero podemos probar en ambos sitios para asegurarnos.



## El teclado

Contamos con tres posibles eventos cuando trabajamos con el teclado.

- **onkeydown**: sucede cuando una tecla está presionada.
- **onkeyup**: cuando se libera una tecla.

- **onkeypress**: cuando se pulsa una tecla.

Con estos eventos podemos desplazarnos a zonas de la página, desplegar un menú determinado e incluso realizar pequeños juegos que respondan a la pulsación de las teclas.

El siguiente código realiza un sencillo uso de **onkeypress**, mostrando en una parte de la página la tecla que se pulsa:

```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type">
<title>Formulario</title>
<script language="JavaScript">

    function mostrar_tecla(e) {
    if (e.keyCode) keycode=e.keyCode;
    else keycode=e.which;
    document.getElementById('zonadeteclas').innerHTML =
    document.getElementById('zonadeteclas').innerHTML + String.fromCharCode(keycode);
    }

</script>
</head>

<body onkeypress="mostrar_tecla(event);">

    Prueba de teclas: <span id="zonadeteclas"></span>

</body>

</html>
```

Cada vez que se pulse una tecla, se mostrará en la zona en la que hemos ubicado la etiqueta **<span>**. El código *JavaScript* utiliza una sencilla fórmula para averiguar qué tecla se ha pulsado. Estas tres líneas nos mostrarían la tecla en una ventana emergente.

```
if (e.keyCode) keycode=e.keyCode;
else keycode=e.which;
alert(String.fromCharCode(keycode));
```

En su lugar la mostramos en la pantalla mediante la propiedad **innerHTML**, que nos permite incorporar en tiempo real un texto, etiqueta o cualquier otro elemento a una página web.



## Nota

Esto se está volviendo complicado, lo sabemos. Podemos tomarlo como pequeñas fórmulas que se pueden aplicar en diferentes condiciones o simplemente no darle mucha importancia. *JavaScript* es un buen complemento, pero no es el centro de la creación de páginas web.

## Controlando el ratón

El ratón es el elemento lanzador de eventos por antonomasia. Cada vez que lo desplazamos o que pasamos por encima de algún lugar, estamos generando eventos que pueden ser manipulados mediante nuestra página web.

Aquí van algunos de esos eventos:

- **onclick**, **ondblclick**, **onmousewheel**: generan un evento cuando se hace clic, doble clic o se usa la rueda del ratón.

- **ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop**: controlan todas las posibilidades que se generan al arrastrar y soltar un elemento en la página web.
- **onmousedown, onmousemove, onmouseout, onmouseover, onmouseup**: controlan el movimiento del ratón cuando no está presionado el botón, al pasar sobre un elemento, al salir de él, etc.
- **onscroll**: cuando hay un desplazamiento vertical.

Más adelante probaremos alguno de estos eventos y otros ya los hemos probado. Aquí es importante fijarse en dónde ponemos el evento. Por ejemplo, un evento **onmousemove** dentro de una imagen nos asegurará que se ejecute la acción prevista cada vez que el ratón se desplace, pero sólo dentro de esa imagen. Si por el contrario lo indicamos en la etiqueta **<body>**, el evento se generará cada vez que el ratón se desplace por cualquier lugar de la página.



## Actividad 5

Probaremos a realizar un evento que muestre un aviso cada vez que se hace doble clic sobre una imagen. Observaremos la diferencia al colocar ese mismo evento en la etiqueta **<body>** o en algún **<div>** que ocupe parte de la pantalla.

## Eventos multimedia

Con la llegada de las etiquetas **<video>** y **<audio>** encontramos también una serie de eventos destinados a gestionar lo que sucede al reproducir elementos multimedia. Encontraremos eventos que se lanzan al reproducir un recurso, al pausarlo, al finalizar, si no se puede reproducir, al cargar datos, al modificarse su volumen y otros muchos más, cuyo análisis es largo y su implementación aún es irregular. De momento nos bastará saber que están ahí.



## Pregunta Verdadero-Falso

Las siguientes afirmaciones, ¿son verdaderas o falsas?

*alert* es una función del lenguaje *JavaScript* que se utiliza para mostrar un mensaje en la ventana del navegador, como una ventana emergente.

Verdadero  Falso

En *JavaScript* hay eventos para todo excepto para controlar aspectos como el momento en que se cambia el tamaño de una ventana, el momento en que se imprime una página web o el momento antes de comenzar la impresión.

Verdadero  Falso

En *JavaScript* las estructuras utilizadas son siempre muy similares, se suele indicar el nombre de la función, los parámetros que va a utilizar y posteriormente, entre llaves, se indican las acciones a realizar.

Verdadero  Falso

Existen cuatro posibles eventos cuando trabajamos con el teclado: *onkeydown*, *onkeyup*, *onkeypress* y *onkeypress*.

Verdadero  Falso

## Resumen



La etiqueta **<form>** nos facilita la creación de formularios, con sus diferentes parámetros.

En el interior de un formulario podemos añadir varios campos.

- Cuadros de texto: **<input type="text" />**
- Cuadros de contraseña: **<input type="password" />**
- Cuadros radiales: **<input type="radio" />**
- Casillas de verificación: **<input type="checkbox" />**
- Botón de envío: **<input type="submit" />**
- Botón de reinicio: **<input type="reset" />**
- Botones: **<input type="button" />**
- Cuadros de archivo: **<input type="file" />**
- Valores ocultos: **<input type="hidden" />**
- Área de texto: **<textfield>**
- Listas desplegables **<select>** con sus correspondientes **<option>** o **<optgroup>**

Aún quedan otros tipos más, menos implantados, pero que ya debemos ir utilizando progresivamente.

Los formularios se complementan con **<fieldset>** y **<legend>** para agrupar elementos y con **<label>** para etiquetarlos.

Por último, hemos realizado un repaso a los posibles eventos que se generan en una página web, así como una aproximación al trabajo con *JavaScript*, el lenguaje de *scripts* que complementa a las páginas web.

## Actividades y ejemplos



### Actividad 1. Formularios

Generaremos un pequeño formulario para ir probando los diferentes tipos de campos que se pueden añadir. En el parámetro **action** optaremos por una acción de envío por correo electrónico, para así poder comprobar los valores que se mandan en cada correo.



### Actividad 2. Campos de un formulario

Incorporaremos a nuestro formulario diferentes tipos de campos. Intentaremos que sean lo más variados posibles.



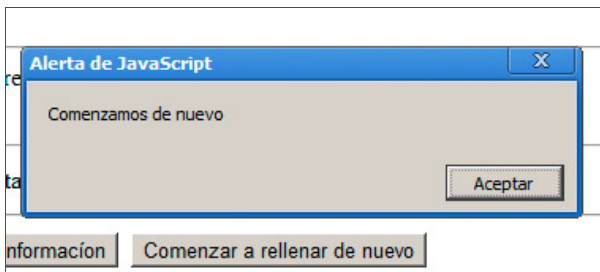
### Actividad 3. Insertar un formulario con un editor web

Diseñaremos un formulario con un editor web que incluya al menos 6 tipos de campos diferentes. Para hacer las pruebas, podemos crearlo con una acción de tipo **mailto**. Así podremos comprobar qué resultados se envían al componer el correo electrónico.



### Actividad 4. Eventos

Probaremos a realizar una pequeña gestión de eventos, de tal modo que, cuando un usuario haga clic en el botón de limpieza del formulario, se le muestre un aviso con cualquier texto, como se recoge en la figura. Ya hemos visto todo lo necesario para conseguirlo. Como pista, la única parte que debemos deducir es dónde ubicaremos la gestión del evento, si en el botón de **reset** o en la etiqueta del formulario. Al ser un evento que afecta a todo el formulario, debe ser en la etiqueta de formulario, pero podemos probar en ambos sitios para asegurarnos.



### Actividad 5. Eventos

Probaremos a realizar un evento que muestre un aviso cada vez que se hace doble clic sobre una imagen. Observaremos la diferencia al colocar ese mismo evento en la etiqueta **<body>** o en algún **<div>** que ocupe parte de la pantalla.



### Ejemplos

Las diferentes prácticas, recursos y ejemplos realizadas en este módulo están disponibles para realizar pruebas.

Ejemplos del módulo

### Aplicación al aula

## Formularios y JavaScript



Crearemos un formulario con nuestros alumnos que se remitirá por correo electrónico, además de aplicarle un estilo apropiado al mismo.



## Programación dirigida al alumnado

### Objetivos

- Aplicar los conceptos de formularios.
- Diseñar diferentes tipos de campos.
- Modificar la apariencia de un formulario mediante estilos.

### Contenidos

- Formularios.
- Campos de un formulario.
- Estilos.

### Materiales y recursos

- Ordenador con acceso a Internet.

### Temporalización

- Dos sesiones.

## Planificación



El alumnado creará un formulario, incorporará varios tipos de campos y finalmente le aplicará los estilos necesarios para que el resultado sea cómodo y atractivo para los usuarios.

### Organización del aula

Trabajaremos en un aula con ordenadores con un agrupamiento individual o por parejas.

### Desarrollo de la actividad

- Se repasan los conceptos de formularios y tipos de campos.
- Se diseña un formulario.
- Se genera una hoja de estilos externa con los elementos básicos del formulario.
- Se comparten los resultados.

### Presentación y evaluación de los resultados

La evaluación se realizaría mediante la revisión del resultado y la observación del proceso. Se pueden evaluar varios aspectos a lo largo de todo el proceso:

- Diseño del formulario.
- Capacidad para aplicar estilos.
- Apariencia general del documento.

---

## Sugerencias metodológicas



La metodología empleada es la de **proyecto**.

Para su aplicación proponemos:

### Primera sesión

Explicamos el objetivo de la actividad y describimos los conceptos necesarios.

Diseñamos un formulario básico que se remitirá por correo.

Decidimos qué campos necesitaremos recoger y posteriormente los incorporamos a nuestro proyecto.

### Segunda sesión

Retomando el ejemplo anterior, aplicamos los estilos necesarios.

Revisamos los resultados y buscamos alternativas.

Compartimos los resultados entre los alumnos, rellenamos los formularios y comprobamos su comodidad y sencillez de uso.

---

## Atención a la diversidad



### Actividad de refuerzo

Para aquellos alumnos/as que puedan tener más dificultad, se les puede facilitar un modelo inicial de plantilla para que ellos la modifiquen.



### Actividad de ampliación

---

La profundización en esta actividad se basaría en que el alumnado probase a añadir algo de interactividad, mediante ejemplos cerrados de eventos con su correspondiente código *JavaScript*.

---