Desarrollo Web en entorno Cliente: JAVASCRIPT

UNIDAD 6. EL MODELO DE EVENTOS EN JAVASCRIPT

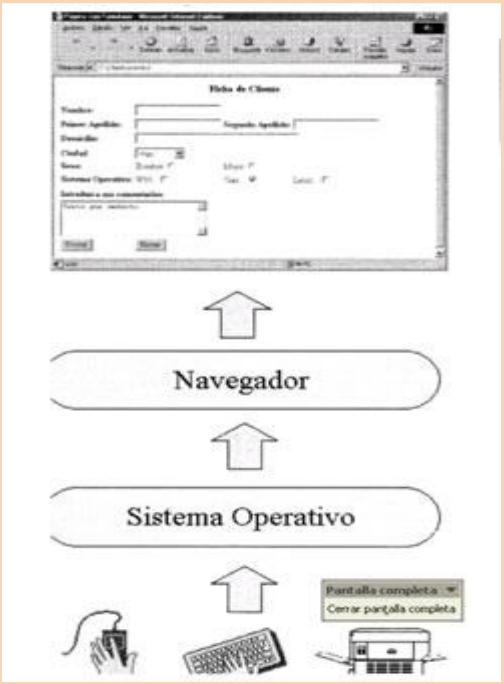
UNIDAD 6. EL MODELO DE EVENTOS EN JAVASCRIPT

6.1 Introducción

Javascript es un lenguaje de guiones (scripts). Los lenguajes de guiones tienen un diseño, filosofía y funcionamiento orientado a objetos y a eventos. En un lenguaje orientado a eventos, la ejecución de la mayor parte de los cómputos que va a realizar el ordenador está supeditada a la activación de alguno de los diferentes eventos que se han definido en el lenguaje y en su sistema de soporte (en nuestro caso el hardware, el sistema operativo y el navegador).

La filosofía de un lenguaje orientado a eventos se basa en dar respuesta a cada uno de los eventos disponibles cuando éstos se produzcan. Por ejemplo, cambiar el color del fondo de la página cuando se pulse con el ratón sobre un botón específico definido en un formulario.

La siguiente figura muestra el funcionamiento de los eventos desde los elementos hardware que los provocan.



Arquitecturalmente, la mayor parte de los eventos provienen de:

1. Señales hardware producidas directamente por el usuario a través del ratón y del teclado (onmouseover, ondblclick, onkeyup, etc.).
2. Señales externas, a veces iniciadas por el usuario (onafterprint, onload, onerror, etc.).
3. Cambios de estado en el entorno, por ejemplo referente a operaciones con el escritorio (oncopy, onpaste, etc.).
4. Sucesos de temporización (onbounce, onfinish, etc.).
5. Eventos generados por programa (disponemos de métodos para simular sucesos tales como onclick, onblur, onfocus, etc.).

Los eventos que los programadores podemos utilizar han variado a lo largo del tiempo, debido a que los navegadores cada vez permiten mayores posibilidades en este sentido.

Cómo se define un evento

Los eventos se capturan mediante los manejadores de eventos. El proceso a realizar se programa mediante funciones JavaScript llamadas por los manejadores de eventos.

Las funciones o código **JavaScript** que se definen para cada **evento** se denominan "**manejador de eventos**" o "**manipulador de evento**".

Para definir las acciones que queremos realizar al producirse un evento utilizamos los manejadores de eventos. Existen muchos tipos de manejadores/manipuladores de eventos, para muchos tipos de acciones del usuario. El manejador/manipulador de eventos se coloca en la etiqueta HTML del elemento de la página que queremos que responda a las acciones del usuario.

Por ejemplo tenemos el manejador de eventos onclick, que sirve para describir acciones que queremos que se ejecuten cuando se hace un click, o sea, se activa el evento click. Si queremos que al hacer click sobre un botón pase alguna cosa, escribimos el manejador onclick en la etiqueta <input type=button> de ese botón. Algo parecido a esto:

```
<input type=button value="pulsame" onclick="sentencias_javascript...">
```

Se coloca un atributo nuevo en la etiqueta que tiene el mismo nombre que el evento, en este caso onclick. El atributo se iguala a las sentencias Javascript que queremos que se ejecuten al producirse el evento.

Cada elemento de la página tiene su propia lista de eventos soportados, vamos a ver otro ejemplo de manejo de eventos, esta vez sobre un menú desplegable, en el que definimos un comportamiento cuando cambiamos el valor seleccionado.

```
<select onchange="window.alert('Cambiaste la selección')">
        < option value="opcion1">Opcion 1
        < option value="opcion2">Opcion 2
< /select>
```

En este ejemplo cada vez que se cambia la opción muestra una caja de alerta: Onchange_ej1.html

Dentro de los manejadores de eventos podemos colocar tantas instrucciones como deseemos, pero siempre separadas por punto y coma. Lo habitual es colocar una sola instrucción, y si se desean colocar más de una se suele crear una función con todas las instrucciones y dentro del manejador se coloca una sola instrucción que es la llamada a la función.

Vamos a ver cómo se colocarían en un manejador varias instrucciones:

```
<input type=button value=Pulsame
  onclick="x=30; window.alert(x); window.document.bgColor = 'red'">
```

Son instrucciones muy simples como asignar a x el valor 30, hacer una ventana de alerta con el valor de x y cambiar el color del fondo a rojo: Onchange_ej2.html

Sin embargo, tantas instrucciones puestas en un manejador quedan un poco confusas, habría sido mejor crear una función así.

```
<script>
function ejecutaEventoOnclick(){
```

```
        x = 30;
         window.alert(x);
        window.document.bgColor = 'red';
}
< /script>

<form>
        < input  type=button  value=Pulsame  onclick="ejecutaEventoOnclick()">
< /form>
```

Ahora utilizamos más texto para hacer lo mismo, pero seguro que a la mayoría les parece más claro este segundo ejemplo.

Tener en cuenta: Jerarquía desde el objeto window

En los manejadores de eventos se tiene que especificar la jerarquía entera de objetos del navegador, empezando siempre por el objeto window. Esto es necesario porque hay algún browser antiguo que no sobreentiende el objeto window cuando se escriben sentencias Javascript vinculadas a manejadores de eventos.

6.2 LISTADO DE TODOS LOS EVENTOS/MANEJADORES DE EVENTOS

La lista de los Eventos/Manejadores de eventos soportados por javascript puede verse aquí:

onabort +++/+++++++
onafterprint ++++++
onafterupdate
onbeforecopy
onbeforecut
onbeforepaste
onbeforeprint ++++++
onbeforeunload +++
onbeforeupdate
onblur ++++
onbounce
oncellchange
onchange ++++
onclick +
oncontextmenu +
oncopy
oncut
ondataavailable
ondatasetchanged
ondatasetcomplete
ondblclick +
ondrag +++++
ondragend +++++
ondragenter +++++
ondragleave +++++
ondragover +++++
ondragstart +++++
ondrop +++++
onerror +++/+++++++
onerrorupdate
onfilterchange
onfinish
onfocus ++++
onhelp
onkeydown ++
onkeypress ++
onkeyup ++
onload +++
onlosecapture
onmousedown +
onmousemove +
onmouseout +
onmouseover +
onmouseup +
onmove
onpaste
onpropertychange
onreadystatechange
onreset ++++
onresize +++
onrowenter
onrowexit
onrowsdelete
onrowsinserted
onscroll +++
onselect ++++
onselectstart
onstart
onsubmit ++++
onunload +++

+ Eventos de ratón

++ Eventos de teclado

+++ Eventos de marco/objetos

++++ Eventos de formulario

+++++ Eventos de arrastre

++++++ Eventos de imprimir

+++++++ Eventos de medios

El modelo de eventos fue estandarizado por el W3C en DOM Nivel 2: OBJETO Event

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

HTML DOM addEventListener() Method (Nivel 2)

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

onabort

Se dispara cuando el usuario cancela la carga de una imagen.
Objetos a los que se aplica

img

onafterprint

Se dispara en el objeto inmediatamente después de que el documento asociado se imprima.
Objetos a los que se aplica

window	body	frameset

onafterupdate

Se dispara después de realizarse la transferencia de datos entre el objeto y el proveedor de datos.
Objetos a los que se aplica

bdo	htmlarea	input type="text"	rb
rt	ruby	textarea	

onbeforecopy

Se dispara en el objeto fuente antes de que la selección se copie en el portapapeles.
Objetos a los que se aplica

a	address	area	b
bdo	big	blockquote	caption
center	cite	code	dd
dfn	dir	div	dl
dt	em	fieldset	form
hn	htmlarea	i	img
label	legend	li	listing
menu	nobr	ol	p
plaintext	pre	s	samp
small	span	strike	strong
sub	sup	td	textarea
th	tr	tt	u
ul			

Onbeforecut

Se dispara en el objeto fuente antes de que la selección se borre del documento
Objetos a los que se aplica

a	address	area	b
bdo	big	blockquote	caption
center	cite	code	dd
dfn	dir	div	dl
dt	em	fieldset	form
hn	htmlarea	i	img
label	legend	li	listing
menu	nobr	ol	p
plaintext	pre	s	samp
small	span	strike	strong
sub	sup	td	textarea
th	tr	tt	u
ul			

onbeforepaste

Se dispara en el objeto destino antes de que la selección se traspase desde el portapapeles del sistema al documento.
Objetos a los que se aplica

a	address	area	b
bdo	big	blockquote	caption
center	cite	code	dd
dfn	dir	div	dl
dt	em	fieldset	form
hn	htmlarea	i	img
label	legend	li	listing
menu	nobr	ol	p
plaintext	pre	s	samp
small	span	strike	strong
sub	sup	td	textarea
th	tr	tt	u
ul			

onbeforeprint

Se dispara en el objeto inmediatamente antes de que el documento asociado se imprima.
Objetos a los que se aplica

window	body	frameset

onbeforeunload

Se dispara antes de que se descargue la página.
Objetos a los que se aplica

frameset	window

onbeforeupdate

Se dispara antes de la transferencia de datos desde el objeto al proveedor de datos.
Objetos a los que se aplica

bdo	htmlarea	input type="text"	rb
rt	ruby	textarea	

onblur

Se dispara cuando el objeto pierde el foco de entrada.
Objetos a los que se aplica

a	window	area	bdo
button	div	embed	fieldset
hr	htmlarea	input type="button"	input type="checkbox"
input type="file"	input type="hidden"	input type="image"	input type="password"
input type="radio"	input type="reset"	input type="submit"	input type="text"
marquee	object	select	span
table	td	textarea	tr

onbounce

Se dispara cuando a la propiedad behavior del objeto marquee se le asigna el valor alternate y el contenido de la marquesina alcanza un extremo.
Objetos a los que se aplica

marquee

oncellchange

Se dispara siempre que cambia el contenido de un proveedor de datos.
Objetos a los que se aplica

bdo	htmlarea	object

onchange

Se dispara cuando el contenido de un objeto o selección ha cambiado
Objetos a los que se aplica

htmlarea	input type="text"	select	textarea

onclick

Se dispara cuando el usuario pulsa el botón izquierdo del ratón sobre el objeto.
Objetos a los que se aplica

a	address	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	dfn
dir	div	dl	document
dt	em	embed	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	legend
li	listing	map	marquee
menu	nextid	nobr	object
ol	option	p	plaintext
pre	rb	rt	ruby
s	samp	select	small
span	strike	strong	sub
sup	table	tbody	td
textarea	tfoot	th	thead
tr	tt	u	ul

xmp			

oncontextmenu

Se dispara cuando el usuario pulsa el botón derecho del ratón en el área del cliente.
Objetos a los que se aplica

document

oncopy

Se dispara en el elemento fuente cuando el objeto o selección se duplica y añade al portapapeles del sistema.
Objetos a los que se aplica

a	address	area	b
bdo	big	blockquote	caption
center	cite	code	dd
dfn	dir	div	dl
dt	em	fieldset	form
hn	hr	htmlarea	i
img	legend	li	listing
menu	nobr	ol	p
plaintext	pre	s	samp
small	span	strike	strong
sub	sup	td	th
tr	tt	u	ul</TD< tr>

oncut

Se dispara en el elemento fuente cuando el objeto o selección es eliminado del documento y añadido al portapapeles.
Objetos a los que se aplica

a	address	area	b
bdo	big	blockquote	caption
center	cite	code	dd
dfn	dir	div	dl
dt	em	fieldset	form
hn	htmlarea	i	img

legend	li	listing	menu
nobr	ol	p	plaintext
pre	s	samp	small
span	strike	strong	sub
sup	td	th	tr
tt	u	ul	

ondataavailable

Se dispara peiódicamente cuando llegan datos provenientes de objetos fuente que transmiten asíncronamente.

Objetos a los que se aplica

	object	xml

ondatasetchanged

Se dispara cuando cambia el conjunto de datos asociados a la información proporcionada por un objeto fuente.

Objetos a los que se aplica

object	xml

ondatasetcomplete

Se dispara para indicar que todos los datos provenientes del objeto fuente están disponibles.

Objetos a los que se aplica

object	xml

ondblclick

Se dispara cuando el usuario realiza una pulsación doble en el objeto.

Objetos a los que se aplica

a	address	document	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	dfn
dir	div	dl	

dt	em	embed	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	legend
li	listing	map	marquee
menu	nextid	nobr	object
ol	p	plaintext	pre
rb	rt	ruby	s
samp	select	small	span
strike	strong	sub	sup
table	tbody	td	textarea
tfoot	th	thead	tr
tt	u	ul	var
xmp			

ondrag

Se dispara continuamente en el objeto fuente durante una operación de arrastre.
Objetos a los que se aplica

a	acronym	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	del
dfn	dir	div	dl
document	dt	em	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	li
listing	map	marquee	menu
nobr	object	ol	option
p	plaintext	pre	q
s	samp	select	small
span	strike	strong	sub

sup	table	tbody	td
textarea	tr	tt	u
ul	xmp		

ondragend

Se dispara en el obejto fuente al finalizar una operación de arrastre.
Objetos a los que se aplica

a	acronym	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	del
dfn	dir	div	dl
document	dt	em	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	li
listing	map	marquee	menu
nobr	object	ol	option
p	plaintext	pre	q
s	samp	select	small
span	strike	strong	sub
sup	table	tbody	td
textarea	tr	tt	u
ul	xmp		

ondragenter

Se dispara en el elemento destino cuando el objeto arrastrado se introduce en un elemento válido para ser soltado.
Objetos a los que se aplica

a	acronym	var	area
b	bdo	big	blockquote
body	button	caption	center

cite	code	dd	del
dfn	dir	div	dl
document	dt	em	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	li
listing	map	marquee	menu
nobr	object	ol	option
p	plaintext	pre	q
s	samp	select	small
span	strike	strong	sub
sup	table	tbody	td
textarea	tfoot	th	thead
tr	tt	u	ul
xmp			

ondragleave

Se dispara Durante una operación de arrastre, este suceso se dispara en el objeto destino cuando el ratón sale de un elemento válido para soltar el origen.

Objetos a los que se aplica

a	acronym	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	del
dfn	dir	div	dl
document	dt	em	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	li
listing	map	marquee	menu
nobr	object	ol	option

p	plaintext	pre	q
s	samp	select	small
span	strike	strong	sub
sup	table	tbody	td
textarea	tr	tt	u
ul	xmp		

ondragover

Se dispara continuamente en el elemento destino mientras que el objeto arrastrado se encuentre en un elemento válido para soltar el objeto fuente.

Objetos a los que se aplica

a	acronym	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	del
dfn	dir	div	dl
document	dt	em	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	li
listing	map	marquee	menu
nobr	object	ol	option
p	plaintext	pre	q
s	samp	select	small
span	strike	strong	sub
sup	table	tbody	td
textarea	tr	tt	u
ul	xmp		

ondragstart

Se dispara en el objeto fuente cuando el usuario comienza a arrastrar un texto u objeto seleccionado.

Objetos a los que se aplica

a	acronym	textarea	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	del
dfn	dir	div	dl
document	dt	em	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	li
listing	map	marquee	menu
nextid	nobr	object	ol
option	p	plaintext	pre
q	rb	rt	ruby
s	samp	select	small
span	strike	strong	sub
sup	table	tbody	td

ondrop

Objetos a los que se aplica

a	acronym	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	del
dfn	dir	div	dl
document	dt	em	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	li
listing	map	marquee	menu
nobr	object	ol	option

p	plaintext	pre	q
s	samp	select	small
span	strike	strong	sub
sup	table	tbody	td
textarea	tr	tt	u
ul	xmp		

onerror

Se dispara cuando se produce un error en la carga de un objeto.
Objetos a los que se aplica

img	link	object	script;
style	window		

onerrorupdate

Se dispara cuando el manejador del suceso onbeforeupdate especificado en el objeto ha cancelado la transferencia de datos y se dispara en lugar del suceso onafterupdate.
Objetos a los que se aplica (Explorer)

bdo	htmlarea	input type="text"	rb
rt	ruby	textarea	

onfilterchange

Se dispara cuando un filtro visual cambia su estado o completa una transición.
Objetos a los que se aplica

bdo	body	button	div
fieldset	htmlarea	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	marquee	nextid	rb
rt	ruby	span	table
td	textarea	th	tr

onfinish

Se dispara cuando se finaliza un bucle.
Objetos a los que se aplica

marquee

onfocus

Se dispara cuando el objeto recibe el foco.
Objetos a los que se aplica

a	window	area	bdo
button	div	embed	fieldset
hr	htmlarea	input type="button"	input type="checkbox"
input type="file"	input type="hidden"	input type="image"	input type="password"
input type="radio"	input type="reset"	input type="submit"	input type="text"
marquee	object	select	span
table	td	textarea	tr

onhelp

Se dispara cuando el usuario pulsa la tecla F1 mientras la ventana actual es el navegador.
Objetos a los que se aplica

a	address	window	area
b	bdo	big	blockquote
button	caption	center	cite
code	dd	dfn	dir
div	dl	document	dt
em	embed	fieldset	font
form	hn	hr	htmlarea
i	img	input type="button"	input type="checkbox"
input type="file"	input type="hidden"	input type="image"	input type="password"
input type="radio"	input type="reset"	input type="submit"	input type="text"
kbd	label	legend	li
listing	map	marquee	menu
nextid	nobr	ol	p
plaintext	pre	rb	rt
ruby	s	samp	select
small	span	strike	strong
sub	sup	table	tbody
td	textarea	tfoot	th
thead	tr	tt	u
ul	var	xmp	

onkeydown

Se dispara cuando el usuario realiza la bajada correspondiente a una pulsación de tecla.
Objetos a los que se aplica

a	acronym	address	var
area	b	bdo	big
blockquote	body	button	caption
center	cite	code	dd
del	dfn	dir	div
document	dt	em	fieldset
font	form	hn	hr
htmlarea	i	input type="button"	input type="checkbox"
input type="file"	input type="hidden"	input type="image"	input type="password"
input type="radio"	input type="reset"	input type="submit"	input type="text"
kbd	label	legend	li
listing	map	marquee	menu
nextid	nobr	object	ol
p	plaintext	pre	q
rb	rt	ruby	s
samp	select	small	span
strike	strong	sub	sup
table	tbody	td	textarea
tfoot	th	thead	tr
tt	u	ul	xmp

onkeypress

Se dispara cuando el usuario pulsa una tecla.
Objetos a los que se aplica

a	acronym	address	var
area	b	bdo	big
blockquote	body	button	caption
center	cite	code	dd
del	dfn	dir	div
document	dt	em	fieldset
font	form	hn	hr
htmlarea	i	input type="button"	input type="checkbox"

input type="file"	input type="hidden"	input type="image"	input type="password"
input type="radio"	input type="reset"	input type="submit"	input type="text"
kbd	label	legend	li
listing	map	marquee	menu
nextid	nobr	object	ol
p	plaintext	pre	q
rb	rt	ruby	s
samp	select	small	span
strike	strong	sub	sup
table	tbody	td	textarea
tfoot	th	thead	tr
tt	u	ul	xmp

onkeyup

Se dispara cuando el usuario realiza la subida correspondiente a una pulsación de una tecla.
Objetos a los que se aplica

a	acronym	address	var
area	b	bdo	big
blockquote	body	button	caption
center	cite	code	dd
del	dfn	dir	div
document	dt	em	fieldset
font	form	hn	hr
htmlarea	i	input type="button"	input type="checkbox"
input type="file"	input type="hidden"	input type="image"	input type="password"
input type="radio"	input type="reset"	input type="submit"	input type="text"
kbd	label	legend	li
listing	map	marquee	menu
nextid	nobr	object	ol
p	plaintext	pre	q
rb	rt	ruby	s
samp	select	small	span
strike	strong	sub	sup
table	tbody	td	textarea
tfoot	th	thead	tr
tt	u	ul	xmp

onload

Se dispara inmediatamente después de que el navegador cargue el objeto.
Objetos a los que se aplica

applet	embed	frameset	img
link	object	script	style
window			

onlosecapture

Se dispara cuando el objeto pierde la captura del ratón.
Objetos a los que se aplica

a	address	var	area
b	bdo	big	blockquote
body	br	button	caption
center	cite	code	dd
dfn	dir	div	dl
dt	em	embed	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	legend
li	listing	map	marquee
menu	p	plaintext	pre
s	samp	select	small
span	strike	strong	sub
sup	table	tbody	td
textarea	tfoot	th	thead
tr	tt	u	ul
xmp			

onmousedown

Se dispara cuando el usuario pulsa sobre el objeto con cualquier botón del ratón.
Objetos a los que se aplica

a	address	var	area
b	bdo	big	blockquote

body	button	caption	center
cite	code	dd	dfn
dir	div	dl	document
dt	em	embed	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	legend
li	listing	map	marquee
menu	nextid	nobr	ol
p	plaintext	pre	rb
rt	ruby	s	samp
select	small	span	strike
strong	sub	sup	table
tbody	td	textarea	tfoot
th	thead	tr	tt
u	ul	xmp	

onmousemove

Se dispara cuando el usuario mueve el ratón sobre el objeto.
Objetos a los que se aplica

a	address	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	dfn
dir	div	dl	document
dt	em	embed	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	legend
li	listing	map	marquee
menu	nextid	nobr	ol
p	plaintext	pre	rb

rt	ruby	s	samp
select	small	span	strike
strong	sub	sup	table
tbody	td	textarea	tfoot
th	thead	tr	tt
u	ul	xmp	

onmouseout

Se dispara cuando el usuario mueve el puntero del ratón desde dentro hacia fuera del objeto.
Objetos a los que se aplica

a	address	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	dfn
dir	div	dl	document
dt	em	embed	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	legend
li	listing	map	marquee
menu	nextid	nobr	ol
p	plaintext	pre	rb
rt	ruby	s	samp
select	small	span	strike
strong	sub	sup	table
tbody	td	textarea	tfoot
th	thead	tr	tt
u	ul	xmp	

onmouseover

Se dispara cunado el usuario mueve el puntero del ratón desde fuera hacia dentro del objeto.
Objetos a los que se aplica

a	address	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	dfn
dir	div	dl	document
dt	em	embed	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	legend
li	listing	map	marquee
menu	nextid	nobr	ol
p	plaintext	pre	rb
rt	ruby	s	samp
select	small	span	strike
strong	sub	sup	table
tbody	td	textarea	tfoot
th	thead	tr	tt
u	ul	xmp	

onmouseup

Se dispara cunado el usuario suelta un botón del ratón situado sobre el objeto.
Objetos a los que se aplica

a	address	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	dfn
dir	div	dl	document
dt	em	embed	fieldset
font	form	hn	hr
htmlarea	i	img	input type="button"
input type="checkbox"	input type="file"	input type="hidden"	input type="image"
input type="password"	input type="radio"	input type="reset"	input type="submit"
input type="text"	kbd	label	legend
li	listing	map	marquee

menu	nextid	nobr	ol
p	plaintext	pre	rb
rt	ruby	s	samp
select	small	span	strike
strong	sub	sup	table
tbody	td	textarea	tfoot
th	thead	tr	tt
u	ul	xmp	

onmove

Se dispara cuando se mueve la ventana
Objetos a los que se aplica

window

onpaste

Se dispara en el objeto destino cuando se transfieren datos desde el portapapeles hasta el documento.
Objetos a los que se aplica

a	address	area	b
bdo	big	blockquote	caption
center	cite	code	dd
dfn	dir	div	dl
dt	em	fieldset	form
hn	htmlarea	i	img
legend	li	listing	menu
nobr	ol	p	plaintext
pre	s	samp	small
span	strike	strong	sub
sup	td	th	tr
tt	u	ul	

onpropertychage

Se dispara cuando una propiedad cambia en el objeto.
Objetos a los que se aplica

a	acronym	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	del
dfn	dir	div	dl
document	dt	em	embed
fieldset	font	form	hn
hr	htmlarea	i	img
input type="button"	input type="checkbox"	input type="file"	input type="hidden"
input type="image"	input type="password"	input type="radio"	input type="reset"
input type="submit"	input type="text"	kbd	label
legend	li	listing	map
marquee	menu	nobr	object
ol	option	p	plaintext
pre	s	samp	script
select	small	span	strike
strong	sub	sup	table
tbody	td	textarea	tfoot
th	thead	tr	tt
u	ul	xmp	

onreadystatechange

Se dispara cuando ha cambiado el estado ready del objeto.
Objetos a los que se aplica

applet	bdo	document	embed
fieldset	img	link	object
script	xml		

onreset

Se dispara cuando un usuario inicializa un formulario.
Objetos a los que se aplica

form

onresize

Se dispara cuando se cambia el tamaño de un objeto.
Objetos a los que se aplica

a	address	var	b
big	blockquote	button	center
cite	code	dd	dfn
dir	div	dl	dt
em	embed	fieldset	form
hn	hr	htmlarea	i
iframe	img	input type="button"	input type="file"
input type="image"	input type="password"	input type="reset"	input type="submit"
input type="text"	isindex	kbd	label
legend	li	listing	marquee
menu	s	samp	select
small	span	strike	strong
sub	sup	table	textarea
tt	u	ul	xmp
window			

onrowenter

Se dispara para indicar que la fila actual ha cambiado y existen nuevos valores de datos en el objeto.
Objetos a los que se aplica

object	xml

onrowexit

Se dispara justo antes de que el control fuente de datos cambia la fila actual en el objeto.
Objetos a los que se aplica

object	xml

onrowsdelete

Se dispara cuando se borran filas en el recordset.
Objetos a los que se aplica

applet	object	xml

onrowsinserted

Se dispara después de añadir nuevas filas en el recordset actual.
Objetos a los que se aplica

object	xml

onscroll

Se dispara cuando se hace una operación de desplazamiento en el objeto.
Objetos a los que se aplica

textarea	bdo	body	div
embed	htmlarea	map	marquee
object	select	table	

onselect

Se dispara cuando cambia la selección actual.
Objetos a los que se aplica (Explorer)

htmlarea	input type="text"	textarea

onselectstart

Se dispara cuando el objeto está siendo seleccionado.
Objetos a los que se aplica

a	address	var	area
b	bdo	big	blockquote
body	button	caption	center
cite	code	dd	del
dfn	dir	div	dl
dt	em	fieldset	font
form	hn	hr	htmlarea
i	img	input type="button"	input type="checkbox"
input type="file"	input type="hidden"	input type="image"	input type="password"
input type="radio"	input type="reset"	input type="submit"	input type="text"
kbd	label	li	listing
map	marquee	menu	nextid

nobr	object	ol	option
p	plaintext	pre	q
rb	rt	ruby	s
samp	select	small	span
strike	strong	sub	sup
table	tbody	td	textarea
tfoot	th	thead	tr
tt	u	ul	xmp

onstart

Se dispara al comienzo de cada bucle en el objeto marquee.
Objetos a los que se aplica

marquee

onsubmit

Se dispara cuando un formulario está a punto de enviarse.
Objetos a los que se aplica

form

onunload

Se dispara inmediatamente después de que el objeto se ha descargado.
Objetos a los que se aplica

frameset	img	window

Consultar en el siguiente enlace la lista de eventos en javascript

que incluye los navegadores donde funcionan dichos eventos

http://help.dottoro.com/ljfvvdnm.php

6.3 DESCRIPCIÓN DE LOS EVENTOS MÁS COMUNES

EVENTOS DE RATÓN	
onmousedown	Se ha presionado un botón del ratón. Se produce el evento onmousedown cuando el usuario pulsa sobre un elemento de la página. onmousedown se produce en el momento de pulsar el botón, se suelte o no.
onmousemove	El puntero del ratón ha sido movido. Se produce cuando el ratón se mueve por la página.
onmouseup	Se ha levantado un botón del ratón. Este evento se produce en el momento que el usuario suelta el botón del ratón, que previamente había pulsado.
onclick	Se ha realizado una pulsación completa (subida y bajada) del botón izquierdo del ratón. Se produce cuando se da una pulsación o clic al botón del ratón sobre un elemento de la página, generalmente un botón o un enlace.
ondblclick	Se ha realizado una pulsación doble sobre el botón izquierdo del ratón.
onmouseover	El puntero del ratón se ha colocado sobre un elemento. Este evento se desata cuando el puntero del ratón entra en el área ocupada por un elemento de la página.
onmouseout	El puntero del ratón ha abandonado el ámbito de un elemento. Se desata un evento onmouseout cuando el puntero del ratón sale del área ocupada por un elemento de la página.
onselect	El evento se produce cuando el usuario selecciona un texto de un textarea o bien de un campo de texto normal.

EVENTOS DE TECLADO	
onkeydown	Se ha presionado (bajado) una tecla. Este evento se produce en el instante que un usuario presiona una tecla, independientemente que la suelte o no. Se produce en el momento de la pulsación.
onkeypress	Se ha pulsado una tecla ANSI. Ocurre un evento onkeypress cuando el usuario deja pulsada una tecla un tiempo determinado. Antes de este evento se produce un onkeydown en el momento que se pulsa la tecla.
onkeyup	Se produce cuando el usuario deja de apretar una tecla. Se produce en el momento que se libera la tecla.

EVENTOS DE ENFOQUE	
onblur	Se desata un evento onblur cuando un elemento pierde el foco de la aplicación. El foco de la aplicación es el lugar donde está situado el cursor, por ejemplo puede estar situado sobre un campo de texto, una página, un botón o cualquier otro elemento.
onfocus	El evento onfocus es lo contrario de onblur. Se produce cuando un

	elemento de la página o la ventana ganan el foco de la aplicación.
EVENTOS DE FORMULARIO	
onabort	Se cancela la carga de una imágen.
onchange	Este evento se produce cuando el usuario cambia el contenido de un elemento de formulario, tipo select, input o textarea.
onreset	Este evento está asociado a los formularios y se desata en el momento que un usuario hace clic en el botón de reset de un formulario.
onsubmit	Ocurre cuando el visitante aprieta sobre el botón de enviar el formulario. Se ejecuta antes del envío propiamente dicho.
EVENTOS DE CARGA DE PÁGINA	
onload	Este evento se desata cuando la página, o en Javascript 1.1 las imágenes, ha terminado de cargarse.
onunload	Al abandonar una página, ya sea porque se pulse sobre un enlace que nos lleve a otra página o porque se cierre la ventana del navegador, se ejecuta el evento onunload.
EVENTOS DE VENTANA	
onmove	Se mueve la ventana. Evento que se ejecuta cuando se mueve la ventana del navegador, o un frame.
onresize	La ventana del navegador cambia de tamaño. Evento que se produce cuando se redimensiona la ventana del navegador, o el frame, en caso de que la página los tenga.
ondragdrop	Este evento se produce cuando el usuario arrastra y suelta un objeto en la ventana del navegador.
OTROS EVENTOS	
onerror	Se produce un error en la carga de un elemento. Se produce cuando no se puede cargar un documento o una imagen y esta queda rota.

6.4 Métodos de evento disponibles en JavaScript

Los siguientes métodos de evento pueden utilizarse en JavaScript:

MÉTODOS DE EVENTO	FUNCIÓN QUE REALIZAN
blur()	Elimina el foco del objeto desde el que se llame
click()	Simula la realización de un *click* sobre el objeto desde el que se llame
focus()	Lleva el foco al objeto desde el que se llame
select()	Selecciona el área de texto del campo desde el que se llame
submit()	Realiza el envío del formulario desde el que se llame

Ejemplo:

```
<html>
<head>
     <title>Eventos</title>
     <script>
     <!--
          function Reacciona(campo) {
               alert("¡Introduzca un valor!");
               campo.focus();
          }
     //-->
     </script>
</head>
<body>
     <form method=post>
          <input type=text name=campo onFocus="Reacciona(this)">
     </form>
</body>
</html>
```

6.5 UTILIZACIÓN DE EVENTOS

En el caso de los eventos de JavaScript, habitualmente la condición que los activa es algún tipo de acción del usuario en el documento, por ejemplo pulsar un botón, situarse sobre un gráfico, abandonar un campo de edición, realizar una selección en una lista desplegable, etc. En todos estos casos resulta conveniente poder ejecutar algun código que proporcione una respuesta.

El código que se asocian a los eventos, a modo de rutinas de tratamiento de interrupción, son a menudo función definidas por el programador de páginas Web. Cuando la acción es realmente sencilla, se puede evitar la activación de una función escribiendo el código "en línea".

En este orden de cosas vamos a estudiar los siguientes apartados:

Escribir código on-line

El código se escribe directamente en el atributo que hace mención del evento:

```
<html>
<head>
</head>
<body>
   <form>
   <input type="text"  onBlur="if (this.value=='')
                alert('Debe rellenar el campo Nombre');">
   </form>
</body>
</html>
```

Ejemplo anterior (Escribir el código del evento directamente en el atributo que hace mención al evento**)**: **EventoOnLine.html**

Activar procedimientos manipuladores de eventos

En el caso de que la acción a realizar requiera de un número apreciable de instrucciones, resulta más cómodo, elegante y mantenible agruparlas en una función y llamar a dicha función en la definición del evento.

En el ejemplo siguiente se define una función, *ComprobarEdad()*, en la cabecera de la página Web y se la invoca desde el segundo campo de texto del formulario. La etiqueta HTML *<input type="text" name="txtEdad" onBlur="ComprobarEdad()">*, utiliza el evento onBlur para ejecutar la función *ComprobarEdad()* cuando el usuario abandona el campo de texto.

```
<html>
<head>
  <script language="javascript">
    <!--
    function ComprobarEdad()
      {
      if (isNaN(frm1.txtEdad.value))
        {
        alert ("Introduzca una edad correcta");
        frm1.txtEdad.value="";
        frm1.txtEdad.focus();
        }
      }
    -->
  </script>
</head>
<body onload="frm1.txtNombre.focus()">
  <form name="frm1" action="">
    Nombre: <input type="text" name="txtNombre" onBlur="if (this.value=='')
                alert('Debe rellenar el campo Nombre');">
    <br>
    Edad: <input type="text" name="txtEdad" onBlur="ComprobarEdad()">
    <br>
    <input type="submit" value="Enviar">
  </form>
</body>
</html>
```

La subrutina *ComprobarEdad()* simplemente comprueba que el valor del campo de texto es numérico; en caso contrario visualiza una ventana de aviso, inicializa el valor y sitúa el foco en el campo para que el usuario pueda vover a rellenarlo con facilidad.

Ejemplo anterior (Escribir el código del evento en un procedimiento manipulador de eventos**)**: **ProcedimientoManipuladorEventos.html**

Configuración de manipuladores de eventos como propiedades del objeto

El objeto Document no dispone de la etiqueta HTML correspondiente. Por tanto, ¿cómo se configura un manipulador de eventos para este objeto?. El secreto es que el manipulador de eventos para cualquier objeto lo constituyen actualmente las propiedades de dichos objetos. Cuando establece un atributo de un manipulador de eventos en una etiqueta HTML, indirectamente está modificando la propiedad del manipulador de eventos correspondiente para el objeto subyacente.

Esto quiere decir que es posible establecer un manipulador de eventos para cada objeto utilizando una sentencia en un script con la sintaxis siguiente:

NombreObjeto.onNombreEvento = manipulador_eventos

por ejemplo:

document.onClick = manipulador_click

```
<html>
<head>
  <script language="javascript">
    <!--
    function miAlert()
      {
      alert("Gracias por pinchar en el botón.");
      }
    -->
  </script>
</head>
<body>
  <form name="frm1" action="">
    <input type="button" name="miBoton" value="pincha aquí">
    <br><br>
    <input type="button" name="miSegundoBoton"
            value="Deshabilita la captura de eventos"
            onclick="document.miFormulario.miBoton.onclick = null;">
  </form>
  <script language="javascript">
    document.miFormulario.miBoton.onclick = miAlert;
  </script>
</body>
</html>
```

Como se puede ver en el ejercicio, para deshabilitar la captura de eventos basta con poner:

document.onClick =null

Ejemplo anterior (Configurar un manipulador de eventos como propiedad del objeto correspondiente**):** **ManipuladorEventosComoPropiedadObjeto.html**

Lo mismo se puede lograr con los atributos *event* y *for* de la etiqueta *script*.

```
<html>
<head>
  <script language="javascript" event="onclick" for="document">
    <!--
    alert("Gracias por pinchar con el botón del ratón.");
    -->
  </script>
</head>
<body>
  <h2>Pincha en el cualquier parte del documento</h2>
</body>
</html>
```

Ejemplo anterior (Configurar un manipulador de eventos con los atributos event y for de la etiqueta script**):** **ManipuladorEventosAtributosEventFor.html**

El único inconveniente de utilizar los procedimientos explicados en este apartado es que no se puede usar para pasar los datos a una función que necesite uno o más argumentos. Si el manipulador de eventos acepta argumentos, debe usar el atributo del manipulador de eventos de la etiqueta HTML del objeto, como veremos en el siguiente apartado.

Uso de this como argumento para hacer referencia a un objeto de evento

Una de las cosas que lleva más tiempo cuando se trabaja con eventos, es la necesidad de hacer referencia a un objeto que ha sido causado por el evento que se ha disparado.

Existen varias maneras de hacer esto, pero la ruta más fácil es usar la palabra clave ***this***. Cuando se incluye como argumento de configuración del manipulador de eventos, hace referencia al objeto en el que se contiene el atributo, por ejemplo:

<img src="ejemplo.jpg" onClick="ManipuladorClick(this)>

El *atributo onClick* establece el manipulador de *eventos Click* para que sea una función llamada *ManipuladorClick()*. No solo esto, sino que *this* se envía como argumento, lo que significa que el actual *objeto image* se pasa a la función.

```
<html>
<head>
  <script language="javascript">
    <!--
    function ManipuladorClick(objeto)
     {
```

```
      alert(objeto.src);
      }
     -->
  </script>
</head>
<body onload="frm1.txtNombre.focus()">
  <h2>Pulse en la imagen para conocer su nombre</h2>
  <form name="frm1" action="">
    <img src="Imagenes/CasaColonPozo.jpg"
              onClick="ManipuladorClick(this)">
    <img src="Imagenes/CasaColonPapagayos.jpg"
              onClick="ManipuladorClick(this)">
    <img src="Imagenes/CasaColonPatio.jpg"
              onClick="ManipuladorClick(this)">
    <img src="Imagenes/CasaColonArcos.jpg"
              onClick="ManipuladorClick(this)">
  </form>
</body>
</html>
```

Ejemplo anterior (Uso de this como argumento de la llamada al manipulador de eventos**):** **ManipuladorEventosArgumentoThis.html**

Generar evento desde programa

Javascript permite al programador producir algunos eventos a través del código. De esta forma no se depende de que un cierto evento sólo lo puedan producir agentes externos.

La siguiente tabla muestra los eventos que pueden ser producidos por la llamada a un método y los objetos que poseen dichos métodos:

Objeto	click()	submit()	focus()	blur()	select()
button	✓				
reset	✓				
submit	✓				
radio	✓				
checkbox	✓				
form		✓			
text			✓	✓	✓
textarea			✓	✓	
select			✓	✓	

```
<html>
<head>
  <script language="javascript">
    <!--
    function Seleccionar()
      {
      txtTexto.select();
      txtTexto.focus();
      }
    -->
  </script>
</head>
<body onload="btnSeleccionar.focus()">
   <input type="text" name="txtTexto" value="Texto por defecto">
   <br>
   <input type="button" name="btnSeleccionar" value="SELECCIONA TEXTO"
                           onClick="Seleccionar()">
</body>
</html>
```

Ejemplo anterior (Genera eventos por programación**)**: **SeleccionarTexto.html**

6.6 UTILIZACIÓN DE EVENTOS EN ELEMENTOS DE FORMULARIO

En este apartado se presentan diversos ejemplos que ilustran la manera de consultar los datos que introduce y selecciona el usuario en un formulario. Aunque únicamente haremos uso de un grupo muy limitado de eventos, los ejemplos que aquí se aportan resultan especialmente interesantes debido a la utilidad que presentan las opciones de código de validación de campos de formularios.

Los campos de formulario que se incluyen en los ejemplos son:

- Cajas de texto
- Cajas de verificación
- Áreas de texto
- Botones de radio
- Listas desplegables
- Botones
- Combinación de elementos de formulario

En este orden de cosas vamos a estudiar los siguientes apartados:

Evento click en botones, botones de radio y casillas de verificación

En este apartado se practica el empleo de botones de radio incorporando, además, una *casilla de verificación Descuento* que indica si un artículo está rebajado de precio. Vamos a suponer que no se admiten descuentos en el pago con tarjeta y que deseamos detectar esta situación tan pronto como se produzca (al seleccionar el usuario el pago con tarjeta o bien el *checkbox Descuento*, siempre que esté marcada esta opción).

La subrutina que detecta la situación expuesta es *ver_descuento();* se asocia al *evento onClick* de cada uno de los *radiobotones Pago* y al *evento onClick* de la *caja de verificación Descuento*. Además se prueba también el *evento onClick* del *botón submit*.

```
<html>
<head>
  <script language="javascript">
    <!--
    function radio_activo(grupo_radio)
      {
      // Repasar el grupo
      for (contador = 0; contador < grupo_radio.length; contador++)
        {
        // Al encontrar el botón activado, devolver el índice
```

```
        if (grupo_radio[contador].checked)
          {
          return contador
          }
        }
      // Si no hay ningún botón activado, devolver -1
      return -1
      }

    function ver_descuento()
      {
      var radio_seleccionado=radio_activo(formulario.Pago);
      if (formulario.Descuento.checked)
        {
        switch (radio_seleccionado)
          {
          case 0:
              alert("El pago con tarjeta no tiene descuento");
              break;
          case 1:
              alert("Descuento: 15%");
                 break;
          case 2:
             alert("Descuento: 5%");
                break;
          }
        }
      }

    function comprobar_radio()
      {
      var radio_seleccionado=radio_activo(formulario.Pago);
      switch (radio_seleccionado)
        {
        case 0:
           alert("El pago se hará con tarjeta de crédito");
               break;
        case 1:
           alert("El pago se hará en efectivo");
               break;
        case 2:
           alert("El pago se hará con un talón");
                break;
        }
      }
    -->
  </script>
</head>
<body>
   <form name="formulario">
   <h1>Forma de pago</h1>
   <input type="radio" name="Pago" value="Tarjeta"
           onclick="ver_descuento()" checked> Tarjeta de crédito<br>
   <input type="radio" name="Pago" value="Metálico"
           onclick="ver_descuento()"> Metálico<br>
   <input type="radio" name="Pago" value="Cheque"
           onclick="ver_descuento()"> Cheque<br><br>
   <input type="checkbox" name="Descuento"
           onclick="ver_descuento()">Descuento<br><br>
```

```
   <input type="Submit" onClick="comprobar_radio()">
   </form>
</body>
</html>
```

Ejemplo anterior (Comprobar evento click en radiobotones, casillas de verificación y botones**)**: **EventoClickFormulario.html**

Evento change en listas desplegables

El siguiente ejemplo define una *lista Marcas* con 6 opciones de marcas de coches sin la posibilidad de realizar selección múltiple.

El *evento onChange* activa la *subrutina VisualizarSeleccionado()*. Esta subrutina señala la opción que ha cambiado tras la nueva selección.

En el caso de las listas desplegables simples, el programador dispone de la matriz options, cuyos elementos se corresponden con las opciones definidas en la lista desplegable y la propiedad selectedIndex para señalar el índice de la opción seleccionada.

```
<html>
<head>
  <script language="javascript">
    <!--
    function VisualizarSeleccionado()
     {
     alert ("Ud. ha cambiado la selección a la marca " +
                frm1.Marcas.options[frm1.Marcas.selectedIndex].text);
     }
    -->
  </script>
</head>
<body>
   <h1>Selección de marcas</h1>
   <form name="frm1">
   <select name="Marcas" onchange="VisualizarSeleccionado()">
     <option> Audi </option>
     <option> BMW </option>
     <option> Ferrari </option>
     <option> Porsche </option>
   </select
   </form>
</body>
</html>
```

Ejemplo anterior (Comprobar evento change en listas desplegables**)**: **EventoChangeSelectFormulario.html**

Eventos change y select en áreas de texto

En el último ejemplo se definen dos *áreas de texto*. Cuando se realiza una selección (*onSelect*) en la primera, se copia el contenido en la segunda. Cuando se realiza una modificación (*onchange*) en la primera, se copia el contenido en la segunda área, pasando previamente todos los caracteres a mayúsculas.

```
<html>
<head>
<style type="text/css">
  textarea.Area  {color: olive;
                  font-size:20;
                  font-style:oblique;}
</style>
</head>
<body>
  <textarea id="Selecc1" class="Area" rows="4" cols="30"
            onchange="Selecc2.value=Selecc1.value.toUpperCase()"
            onselect="Selecc2.value=Selecc1.value">
  Origen:
  </textarea>
  <br><br>
  <textarea id="Selecc2" class="Area" rows="4" cols="30"
           style="color:blue;">
  Destino:
  </textarea>
</body>
</html>
```

Ejemplo anterior (Comprobar el evento change y select en áreas de texto**)**:
EventosSelectChangeFormulario.html

OTROS EJEMPLOS DE RUTINAS PARA APLICAR A FORMULARIOS

Colocar el foco en un determinado campo al cargar la página
Foco_campo_cargar_página.htm

Escribe un valor indicado en otra ventana, en el formulario abierto
Escribe_valor_otra_ventana_formu_abierto.htm

Comprueba que el valor ingresado en dos campos no sea el mismo
Valores_dos_campos_distintos.htm

Comprueba que el valor ingresado en dos campos sea igual
Valores_dos_campos_iguales.htm

Permite seleccionar solo una determinada cantidad de checkbox del mismo nombre
Selecciona_checkbox_mismo_nombre.htm

Selecciona todos los checkbox con un solo click o los deselecciona
Selecciona_num_determ_checkbox – click.htm

Selecciona todos los checkbox con un solo click o los deselecciona o invierte la selección
Selec_desel_inver_num_determ_checkbox – click.htm

Cuenta la cantidad de checkbox o radio que hay seleccionados
Cuenta_checkbox_radio_selecc.htm

Aumenta o disminuye un valor
Aumenta_disminuye_valor.htm

Cuenta la cantidad de palabras introducidas en un textarea
Numero_palabras_introd_textarea.htm

Muestra o esconde el botón "submit"
Muestra_esconde_submit.htm

Habilita o deshabilita el botón "submit"
Habilita_deshabilita_submit.htm

Mensaje de confirmación al borrar un formulario
Confirmación_borrar_formulario.htm

Mensaje de confirmación al enviar un formulario
Confirmación_enviar_formulario.htm

6.7 OTROS EVENTOS. UTILIZACIÓN DE LOS EVENTOS MÁS COMUNES

En este apartado se presentan ejemplos sencillos que hacen uso de los eventos que se utilizan más habitualmente.

En el apartado anterior únicamente utilizamos los eventos onClick, onChange y onSelect, puesto que el objetivo era habituarnos a su definición y utilización, a la vez que proporcionar una base en el uso y validación de elementos de formulario.

Los eventos que se explican en este apartado se dividen en los siguientes grupos:

Eventos de ratón

En el ejemplo que vemos a continuación se hace uso de los *eventos onClick* y de los *eventos de ratón onmouseOver* y *onmouseOut* para modificar varias características de DHTML definidas a través de los estilos en cascada (CSS).

```
<html>
<head>
<style type="text/css">
       p {color:green;
          font-family:serif;
      font-size:30;
      font-style:oblique;}
</style>
<script language="javascript">
  <!--
     function f_onmouseover_texto1()
       {
       texto1.style.color="red";
       texto1.style.fontSize="60";
       }

     function f_onmouseout_texto1()
       {
       texto1.style.color="blue";
       texto1.style.fontSize="30";
       }

     function f_onclick_texto2()
       {
       texto2.style.color="yellow";
       texto2.style.fontSize="60";
       }
```

```
    function f_onmouseout_texto2()
      {
      texto2.style.color="red";
      texto2.style.fontSize="30";
      }

    function f_onmouseover_texto3()
      {
      boton1.style.visibility="hidden";
      texto4.style.visibility="hidden";
      }

    function f_onmouseout_texto3()
      {
      boton1.style.visibility="visible";
      texto4.style.visibility="visible";
      }
  -->
</script>
</head>
<body>
 <h2 id="texto1" style="font-size:30; color:blue; font-family:sans-serif"
     onmouseover="f_onmouseover_texto1()"
     onmouseout="f_onmouseout_texto1()">
                    ¡Mueve el ratón hacia aquí!</h2>
 <p> Este es un párrafo y su estilo está definido en la cabecera
                    </p>
 <p id="texto2" style="color:red" onclick="f_onclick_texto2()"
     onmouseout="f_onmouseout_texto2()">Pincha una vez aquí</p>
 <p id="texto3" style="color:red; font-size:40;"
     onmouseover="f_onmouseover_texto3()"
     onmouseout="f_onmouseout_texto3()">
                    Mira como desaparecen el saludo y el botón</p>
 <p id="texto4" style="color:blue; font-size:50;">HOLA</p>
 <input type="button" name="boton1" value="El botón mágico">
 <h3>Estos estilos son comunes</h3>
 <h4> a muchas páginas</h4>
</body>
</html>
```

Ejemplo anterior (Manejo de algunos eventos de ratón: onClick, onMouseover y onMouseout**)**: **EventosMouseoverMouseout.html**

Eventos de teclado

Cuando pulsa una tecla, está haciendo dos cosas: en primer lugar pulsa la tecla y a continuación la suelta para pulsarla de nuevo. Cada una de esas acciones lanza su propio evento. Específicamente, pulsar la tecla lanza el evento keyDown, soltar la tecla lanza el evento keyUp y el final de una secuencia completa lanza el evento keypress.

El siguiente ejemplo establece manipuladores para los eventos keyDown y keyUp. El destino es el objeto text y los manipuladores son sentencias status del objeto window que visualizan mensajes en la barra de estado del navegador.

```
<body>
  <form>
  <b>Asegúrese de que el foco está dentro del cuadro,<br>
     pulse una tecla y observe la barra de estado.</b>
  <br>
  <input
       type="text"
       onKeyDown="window.status = 'La tecla está pulsada'"
       onKeyUp="window.status = 'La tecla no está pulsada'">
  </form>
</body>
```

Ejemplo anterior (Manejo de eventos de teclado: keyDown y keyUp**):**
EventosKeyDownKeyUp.html

Obtención de valores de teclas específicos

Como puede imaginar, el manejo de los eventos keyDown, keyUp y keyPress por sí mismos no le llevarían muy lejos. Lo que más interesa la mayoría de las veces es conocer exactamente qué tecla pulsa el usuario. Por ejemplo, si tiene un campo de formulario que sólo debe contener números, puede buscar las teclas no numéricas que se vayan a pulsar y cancelarlas.

Para **capturar el valor del ratón y/o pulsación del teclado** exacta que ha pulsado el usuario es necesario utilizar el **objeto event**, ***que Javascript emplea para almacenar datos sobre el objeto actual***. Por ejemplo, si javascript detecta uno de los eventos de teclado, usa el objeto event para almacenar un código que especifique la tecla que se ha pulsado.

Acceda al objeto Event pasándolo a la función del manipulador de eventos. El modo en que esto suceda dependerá de como defina el manipulador de eventos:

- Si utiliza un atributo dentro de una etiqueta HTML, incluya la palabra **event** de esta forma: **onKeyPress="manipulador_keypress(event)"**.
- Si utiliza la propiedad del manipulador de eventos del objeto, no hay ningún modo de enviar un argumento, sin embargo Javascript lo hará automáticamente.

En cualquier caso, asegúrese de configurar la función para que acepte el objeto event como argumento:

```
function manipulador_keypress(objeto_event)
 {
 sentencias
 }
```

Para que la captura del objeto event se pueda realizar en todos los navegadores usaremos event.wich || event.keycode

```
<html>
   <head>
   <script language="javascript">
     <!--
     function ManejarKeyPress(tecla)
       {
         if (document.all)
           {
           // Probablemente, Internet Explorer 4 y posteriores
           status = String.fromCharCode(event.keyCode)
           return true
           }
         else if (document.getElementById)
           {
           // Probablemente, Netscape 6 y posteriores
           status = String.fromCharCode(tecla.which)
           return true
           }
         else if (document.layers)
           {
           // Probablemente Netscape 4
           status = String.fromCharCode(tecla.which)
           return true
           }
       }
     -->
   </script>
   </head>
   <body>
    <form>
     <b>Haga clic dentro de este cuadro de texto y pulse una tecla.<br>
     La tecla pulsada aparece en la barra de estado.</b>
     <br>
     <input
        type="text"
        onKeyPress="return ManejarKeyPress(event)">
     </form>
   </body>
   </html>
```

Ejemplo anterior (Manejo de eventos de ratón: onClick, onKeyPress**):**
EventoKeyPress.html

Cómo decidir qué botón utilizar

En Internet Explorer 4, el código del botón se almacena en ***event.button***.

Códigos de botón reconocidos por Internet Explorer	
Código de botón	**Generado al pulsar**
0	Nada
1	Botón izquierdo
2	Botón derecho
3	Botones izquierdo y derecho
4	Botón central
5	Botones izquierdo y central
6	Botones derecho y central
7	Los tres botones

Una buena manera de colocar estos códigos para usarlos correctamente es considerar un dispositivo que impida pulsar el botón derecho para evitar copias de texto o imágenes.

```
<html>
<head>
<script language="javascript">
  <!--
  if (document.all)
    {
    }
  else if (document.getElementById)
    {
    document.captureEvents(Event.MOUSEDOWN)
    }
  else if (document.layers)
    {
    document.captureEvents(Event.MOUSEDOWN)
    }

  document.onmousedown = manejar_mousedown

  function manejar_mousedown(evento_raton)
    {
    var click_derecho_no_permitido =
            "¡Lo sentimos; el clic derecho no está permitido!"
    if (document.all)
      {
      // Probablemente, Internet Explorer 4 y posteriores
      if (event.button == 2 || event.button == 3)
        {
        alert(click_derecho_no_permitido)
        return false
        }
      }
    else if (document.getElementById)
      {
```

```
      // Probablemente, Netscape 6 y posteriores
      if (evento_raton.which == 3)
        {
        alert(click_derecho_no_permitido)
        return false
        }
      }
    else if (document.layers)
      {
      // Probablemente Netscape 4
      if (evento_raton.which == 3)
        {
        alert(click_derecho_no_permitido)
        return false
        }
      }
    }
  -->
</script>
</head>
<body>
  <h2>Cueva pintada de gáldar</h2>
  <img name="imagen1" src="Imagenes/1.jpg">
  <img name="imagen2" src="Imagenes/3.jpg">
  <img name="imagen3" src="Imagenes/2.jpg">
</body>
</html>
```

Ejemplo anterior (Inhabilitar el botón derecho del ratón**):** **ClickDerecho.html**

El script de arriba funciona fenomenal con Explorer, pero da algunos problemillas con Firefox, concretamente, sale el aviso pero después aparece el menú contextual. Sugerimos que se utilice dicho script, además de añadir este evento a la etiqueta body: oncontextmenu="return false" Quedaría <body oncontextmenu="return false"> Esto funciona perfectamente al menos con los navegadores más modernos.

Ejemplo anterior (Inhabilitar el botón derecho del ratón incluso en Mozilla Forefox**):** **ClickDerecho3.html**

Eventos de enfoque

Este ejemplo permite acumular en un área de texto los párrafos seleccionados en una página provista de pares (párrafo, botón de selección del párrafo) y cambiar los colores de los botones usando los *eventos onFocus y onBlur*. En definitiva, un botón puede tomar los siguientes colores:

- **Estándar** (habitualmente gris): cuando el usuario todavía no se ha situado sobre ese botón.
- **Orchid**: cuando el usuario está situado sobre el botón.
- **Skyblue**: cuando el usuario ya se situó sobre ese botón, pero no lo seleccionó.
- **Rojo**: cuando el usuario ha seleccionado (doble click) ese botón.

```
<html>
```

```
<head>
<style type="text/css">
      p {color:green;
         font-family:serif;
      font-size:20;
      font-style:oblique;}
</style>
<script language="javascript">
  <!--
 function f_onfocus(Boton)
   {
   if (Boton.style.backgroundColor!="red")
     Boton.style.backgroundColor="orchid";
   }

 function f_onblur(Boton)
   {
   if (Boton.style.backgroundColor!="red")
     Boton.style.backgroundColor="skyblue";
   }

 function AniadirASeleccion(Boton)
   {
   if (Boton.style.backgroundColor!="red")
     {
     Boton.style.backgroundColor="red";
     Seleccion.innerText=Seleccion.innerText+eval(Boton.value).innerText;
     }
   }
  -->
</script>
</head>
<body>
 <p id="P1">¿Qué hacíais en el bosque?</p>
 <input type="button" value="P1"
     onfocus="f_onfocus(this)"
     onblur="f_onblur(this)"
     ondblclick="AniadirASeleccion(this)">
 <p id="P2">
       Buscábamos comida y bebida, pues nos moríamos de hambre.</p>
 <input type="button" value="P2"
     onfocus="f_onfocus(this)"
     onblur="f_onblur(this)"
     ondblclick="AniadirASeleccion(this)">
 <p id="P3">Pero, en definitiva. ¿Qué asunto os trajo al
             bosque?-preguntó el rey enojado.</p>
 <input type="button" value="P3"
     onfocus="f_onfocus(this)"
     onblur="f_onblur(this)"
     ondblclick="AniadirASeleccion(this)">
 <p id="P4">
          Thorin cerró entonces la boca y no dijo nada más.</p>
 <input type="button" value="P4"
     onfocus="f_onfocus(this)"
     onblur="f_onblur(this)"
     ondblclick="AniadirASeleccion(this)">
 <p style="background-color='skyblue'">
           El hobbit (J.R.R. Tolkien)</p>
 <br>
```

```
 <textarea id="Seleccion" rows="7" cols="50">
   El hobbit:
 </textarea>
</body>
</html>
```

Ejemplo anterior (Manejo de algunos eventos de enfoque: onFocus y onBlur**):**
EventosFocusBlur.html

Eventos de ventana

El ejemplo de este apartado nos muestra la manera de visualizar un mensaje de bienvenida y de despedida junto a la página Web al cargarla y descargarla. Para ello hacemos uso de los eventos onLoad y onUnload asociados a la ventana (window), por lo que lo situamos dentro de la etiqueta <body>.

```
<body onload='alert("Bienvenido a la página de Pepito Grillo");'
      onunload='alert("Gracias por visitar nuestra página");'>
   <h1>Pepito grillo</h1>
   Datos de Pepito
</body>
```

Ejemplo anterior (Manejo de eventos de ventana: onLoad y onUnload**):**
EventosLoadUnload.html

El siguiente ejemplo utiliza el evento onResize, en el cual actualizamos dos cajas de texto con la posición horizontal y vertical de la esquina inferior derecha de la ventana (respecto al origen de la pantalla).

```
<body onResize='X.value=event.screenX;Y.value=event.screenY'>
  <h2>Posiciones de la esquina inferior derecha de la ventana</h2>
  X: <input type="text" name="X" size="4"><br>
  Y: <input type="text" name="Y" size="4">
</body>
```

Ejemplo anterior (Manejo de algunos eventos de ventana: onResize**):**
EventoResize.html

Otros eventos

El siguiente ejemplo muestra cómo asociar una acción al evento onAbort, que se activa cuando detenemos la carga de una imagen o no llega a cargarse (habitualmente usando el botón "detener" del navegador o pulsando la tecla "escape"). En este caso visualizamos un mensaje indicando que la carga

ha sido interrumpida y rellenamos un área de texto con una breve explicación del contenido de la imagen; para ello utilizamos la *subrutina manejar_abort()* asociada al *evento onAbort* definido en la *etiqueta <img>.*

```
<html>
<head>
<script language="javascript">
  <!--
  function manejar_abort(imagen)
    {
    var mensaje_abort =
         "La siguiente imagen no se ha cargado por completo:\n\n"
    mensaje_abort += imagen + "\n\n"
    mensaje_abort += "¿Desea recargar la página?"
    if (confirm(mensaje_abort))
      {
      location.reload()
      }
    }
  -->
</script>
</head>
<body>
 <img src="Imagenes/Ejemplo.gif"  style="widt:200000px"
         onAbort="manejar_abort(this.src)">
</body>
</html>
```

Ejemplo anterior (Manejo de otros eventos: onAbort**):** **EventoAbort.html**

El último ejemplo de este apartado muestra la definición y funcionamiento del *evento onError*. En este caso se asocia el evento a la *etiqueta <img>*, de tal manera que si existe un error en la carga de la imagen se activa el evento. En el ejemplo intentamos cargar la imagen *"noexiste.gif"*, que obviamente no existe, por lo que se activa el *evento onError* y se visualiza el mensaje *"¡Error! La siguiente imagen no ha podido cargarse:".*

```
<html>
<head>
<script language="javascript">
  <!--
  var contador_recarga = 0
  var maxima_recarga = 5

  function manejar_error(imagen)
    {
    // ¿Hemos llegado al máximo de recargas?
    if (contador_recarga < maxima_recarga)
      {
      // Si no es así, incrementamos el contador de recargas
      contador_recarga++
      // Intentamos cargar la imagen de nuevo
      document.images["mala"].src = imagen
      }
    else
      {
      // Si ya no se puede seguir recargando, mostramos un mensaje
```

```
      //(en la práctica, puede que lo mejor
      // sea dejarlo aquí y no molestar más)
      alert("¡Error! La siguiente imagen no ha podido cargarse:\n\n" +
          document.images["mala"].src)
      }
    }
  -->
</script>
</head>
<body>
  <img src="noexiste.gif" name="mala"
         onError="manejar_error(this.src)">
</body>
</html>
```

Ejemplo anterior (Manejo de otros eventos: onError**):** **EventoError.html**

6.8 MODELO DE EVENTOS DEL INTERNET EXPLORER (objeto event)

El **objeto event** está disponible únicamente mientras se da un evento. Esto quiere decir que podremos usarlo dentro de un manejador de evento, pero no en cualquier otra parte del código. A pesar de que todas las propiedades de event están disponibles, hay algunas propiedades que sólo tienen sentido cuando se da un determinado evento y no otro.

Pasamos a ver las propiedades más destacadas que nos ofrece este objeto:

Objeto Event del Internet Explorer	
Propiedad	**Descripcion**
altKey	Nos dice el estado de la tecla **ALT**
button	Nos dice qué botón ha sido pulsado
cancelBubble	Valor booleano que indica si se permite o no el burbujeo de suscesos
clientX	Nos dice la coordenada **x** del ratón relativa al área cliente de la ventana, excluyendo barras de scroll y otros elementos decorativos
clientY	Idem para la coordenada **y** del ratón
ctrlKey	Nos dice el estado de la tecla **CTRL**
ctrlLeft	Nos dice el estado de la tecla **LEFT izquierda**
fromElement	Nos da una referencia del objeto desde el que el usuario está moviendo el ratón
keyCode	Nos dice el **código Unicode** asociado a la tecla que causó el evento
offsetX	Nos dice la coordenada **x** del ratón relativa al objeto que lanzó el evento
offsetY	Idem para la coordenada **y** del ratón

repeat	Nos dice si el evento **onKeyDown** está siendo repetido
returnValue	Podremos asignar un valor de retorno para el evento asignándolo a esta propiedad
screenX	Nos dice la coordenada **x** del ratón relativa a la pantalla del usuario
screenY	Idem para la coordenada **y** del ratón
shiftKey	Nos dice el estado de la tecla **SHIFT**
srcElement	Nos devuelve una referencia al objeto que ha iniciado el evento
toElement	Nos da una referencia del objeto hacia el que el usuario está moviendo el ratón
type	Nos da el nombre del evento
x	Nos da la coordenada **x** del ratón (en pixels) relativa al elemento padre
y	Idem con la coordenada **y** del ratón

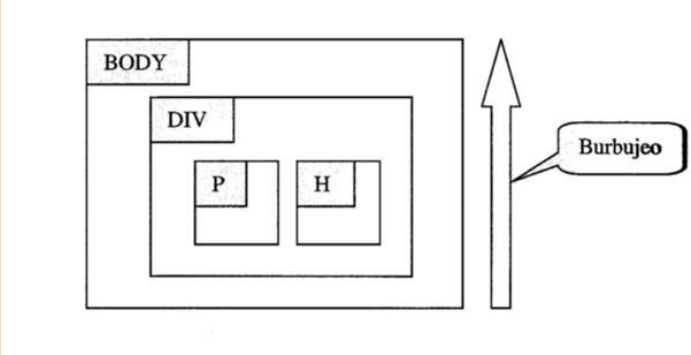
Todas las propiedades salvo repeat son de lectura/escritura y por tanto, su valor se puede leer y/o establecer.

El modelo de eventos del DOM (Document Object Model, jerarquía de objetos normalizada por W3C y que sigue, en este tema, Explorer y no Netscape) proporciona con este fin un concepto llamado burbujeo de eventos ('event bubbling'). Habitualmente a cada elemento le asociamos una serie de eventos que consideramos necesarios, sin embargo, en algunas ocasiones resulta conveniente jerarquizar la asignación de manejadores de eventos.

Por ejemplo en la siguiente sección de código presentamos la situación en la que un texto de cabecera <h1> no tiene asociado ningún manejador de eventos. En este caso al realizar una pulsación con el ratón, se activa el manejador de su elemento padre . El evento onkeypress del elemento <body> también se activa cuando pulsamos una tecla ANSI estando situados en cualquiera de los dos texto (Hola, Adiós).

```
<html>
<body onkeypress="alert('BODY')">
<div onclick="alert('DIV')">
<p onclick="alert('P')">Hola</p>
<h1>Adiós</h1>
</div>
</body>
</html>
```

Por otra parte, en las situaciones en las que tanto un elemento como alguno de sus ancestros tienen un mismo tipo de manejador de eventos asociado, por defecto, después de ejecutarse el manejador de eventos del elemento, se van ejecutando los manejadores (del mismo tipo) de los ancestros. Esta situación la encontramos en el ejemplo anterior con el evento onclick de <p> y de ; al pulsar el botón izquierdo del ratón en "Hola", aparecen sucesivamente ventanas con los mensajes: P y DIV.



Si se desea cortar el mecanismo de burbujeo de sucesos en alguna etapa de la ejecución de los manejadores de eventos, basta con utilizar la propiedad cancelBubble del objeto event de la siguiente manera: window.event.cancelBubble = true; En nuestro ejemplo se debe cambiar la línea de definición del párrafo para evitar la ejecución del manejador asociado a la etiqueta ;. La etiqueta ; se convierte en: Hola

El siguiente ejercicio aporta un ejemplo práctico en el que se puede apreciar la utilidad del burbujeo de sucesos. Podemos escribir diversas páginas conteniendo los párrafos de un libro con un tamaño de letra pequeño, para que una página abarque el mayor contenido posible de información. A la vez, gracias a las características de DHTML, será posible ampliar el tamaño de un párrafo determinado sólo con situarse sobre el mismo.

```
<html>
<head>
<style type="text/css">
       p {color:green;
          font-family:serif;
      font-size:20;
      font-style:oblique;}
</style>
<script language="javascript">
  <!--
     function f_onmouseover()
       {
       Elemento=window.event.srcElement;
       Elemento.style.color="blue";
       Elemento.style.fontSize="40";
       }

     function f_onmouseout()
       {
       Elemento=window.event.srcElement;
       Elemento.style.color="green";
       Elemento.style.fontSize="20";
       }
  -->
</script>
</head>
<body>
<div onmouseover="f_onmouseover()" onmouseout="f_onmouseout()">
  <p>¿Qué hacíais en el bosque?</p>
  <p>Buscábamos comida y bebida, pues nos moríamos de hambre.</p>
  <p>Pero, en definitiva. ¿Qué asunto os trajo al
              bosque?-preguntó el rey enojado.</p>
```

```
   <p>Thorin cerró entonces la boca y no dijo nada más.</p>
      <br>
   <p style="background-color='skyblue'"
           onmouseover="this.style.backgroundColor='cyan'"
           onmouseout="this.style.backgroundColor='skyblue'"
           >El hobbit (J.R.R. Tolkien)</p>
</div>
 </body>
 </html>
```

Ejemplo anterior (Burbuejo de sucesos**):** **Burbujeo.html**

En este ejemplo (último de la saga del hobbit) se permite trasladar al área del texto, tanto el contenido de cada párrafo (al pulsar en la tecla T) como la indicación del número de párrafo dentro de la página (al pulsar en una tecla diferente de T). El valor de la tecla pulsada lo obtenemos del objeto event.

```
<html>
<head>
<style type="text/css">
 p {color:green;
    font-family:serif;
    font-size:20;
    font-style:oblique;}
</style>
<script language="javascript">
  <!--
  function AniadirASeleccion()
    {
    Boton=window.event.srcElement;
    Tecla=window.event.keyCode;
    if (Boton.style.backgroundColor!="red")
       Boton.style.backgroundColor="red";
    if (Tecla==84 || Tecla==116) <!-- t,T -->
       Seleccion.innerText=Seleccion.innerText+eval(Boton.value).innerText;
    else
       Seleccion.innerText=Seleccion.innerText+Boton.value;
    }
  -->
</script>
</head>
<body onload="Seleccion.innerText=''">
  <div onkeypress="AniadirASeleccion()">
    <p id="P1">¿Qué hacíais en el bosque?</p>
    <input type="button" value="P1">
    <p id="P2">Bucábamos comida y bebida, pues nos moríamos de hambre.
                  </p>
    <input type="button" value="P2">
    <p id="P3">Pero, en definitiva. ¿Qué asunto os trajo al
                  bosque?-preguntó el rey enojado.</p>
    <input type="button" value="P3">
    <p id="P4">Thorin cerró entonces la boca y no dijo nada más.
                  </p>
    <input type="button" value="P4">
  </div>
```

```
  <p style="background-color='skyblue'">El hobbit (J.R.R. Tolkien)
                  </p>
  <br>
  <textarea id="Seleccion" rows="7" cols="50">El hobbit:
                  </textarea>
</body>
</html>
```

Ejemplo anterior (Burbujeo de sucesos y uso del objeto event**)**: **ObjetoEvent.html**

6.9 MODELO DE EVENTOS PARA LOS NAVEGADORES QUE SIGUEN LOS ESTANDARES. PROPIEDADES DEFINIDAS POR DOM (objeto event)

La siguiente tabla recoge las propiedades definidas para el objeto event en los navegadores que siguen los estándares:

Objeto event (navegadores estándares DOM)		
Propiedad/Método	**Devuelve**	**Descripción**
`altKey`	Boolean	Devuelve true si se ha pulsado la tecla ALT y false en otro caso
`bubbles`	Boolean	Indica si el evento pertenece al flujo de eventos de *bubbling*
`button`	Número entero	El botón del ratón que ha sido pulsado. Posibles valores: 0 – Ningún botón pulsado 1 – Se ha pulsado el botón izquierdo 2 – Se ha pulsado el botón derecho 3 – Se pulsan a la vez el botón izquierdo y el derecho 4 – Se ha pulsado el botón central 5 – Se pulsan a la vez el botón izquierdo y el central 6 – Se pulsan a la vez el botón derecho y el central 7 – Se pulsan a la vez los 3 botones
`cancelable`	Boolean	Indica si el evento se puede cancelar
`cancelBubble`	Boolean	Indica si se ha detenido el flujo de eventos de tipo *bubbling*
`charCode`	Número entero	El código unicode del carácter correspondiente a la tecla pulsada

Objeto event (navegadores estándares DOM)		
Propiedad/Método	**Devuelve**	**Descripción**
`clientX`	Número entero	Coordenada X de la posición del ratón respecto del área visible de la ventana
`clientY`	Número entero	Coordenada Y de la posición del ratón respecto del área visible de la ventana
`ctrlKey`	Boolean	Devuelve true si se ha pulsado la tecla CTRL y false en otro caso
`currentTarget`	Element	El elemento que es el objetivo del evento
`detail`	Número entero	El número de veces que se han pulsado los botones del ratón
`eventPhase`	Número entero	La fase a la que pertenece el evento: 0 – Fase capturing 1 – En el elemento destino 2 – Fase bubbling
`isChar`	Boolean	Indica si la tecla pulsada corresponde a un carácter
`keyCode`	Número entero	Indica el código numérico de la tecla pulsada
`pageX`	Número entero	Coordenada X de la posición del ratón respecto de la página
`pageY`	Número entero	Coordenada Y de la posición del ratón respecto de la página
`preventDefault()`	Función	Se emplea para cancelar la acción predefinida del evento
`relatedTarget`	Element	El elemento que es el objetivo secundario del evento (relacionado con los eventos de ratón)

Objeto event (navegadores estándares DOM)		
Propiedad/Método	**Devuelve**	**Descripción**
`screenX`	Número entero	Coordenada X de la posición del ratón respecto de la pantalla completa
`screenY`	Número entero	Coordenada Y de la posición del ratón respecto de la pantalla completa
`shiftKey`	Boolean	Devuelve true si se ha pulsado la tecla SHIFT y false en otro caso
`stopPropagation()`	Función	Se emplea para detener el flujo de eventos de tipo *bubbling*
`target`	Element	El elemento que origina el evento
`timeStamp`	Número	La fecha y hora en la que se ha producido el evento
`type`	Cadena de texto	El nombre del evento

Al contrario de lo que sucede con Internet Explorer, la mayoría de propiedades del **objeto event de DOM** son de sólo lectura. En concreto, solamente las siguientes propiedades son de lectura y escritura: `altKey`, `button` y `keyCode`.

6.10 NUEVOS ATRIBUTOS DE EVENTO EN HTML5

En el siguiente enlace podemos consultar los nuevos atributos de evento incorporados en HTML5 y que pertenecen a los eventos siguientes:

- Eventos para el objeto Window
- Eventos de Formulario (Form)
- Eventos de teclado (Keyboard)
- Eventos de ratón (Mouse)
- Eventos para vídeos, imágenes y audio (Media)

http://www.w3schools.com/tags/ref_eventattributes.asp

6.11 EVENTOS Touch (HTML 5) de JAVASCRIPT (eventos de toque)

Con el fin de proporcionar soporte de calidad para usuarios de interfaces táctiles, los eventos táctiles o eventos de toque dan la posibilidad de interpretar la actividad de los dedos en pantallas táctiles o trackpads. https://developer.mozilla.org/es/docs/DOM/Touch_events

Un vistazo a los nuevos eventos Touch de JAVASCRIPT:
http://www.desarrolloweb.com/articulos/eventos-touch-javascript.html

Desarrollo web con tecnología Multitouch:
http://www.html5rocks.com/es/mobile/touch/#toc-events

Eventos Touch (de toque) - DOM | MDN: (2015)
https://developer.mozilla.org/es/docs/DOM/Touch_events