

# Eventos DOM HTML de JavaScript

HTML DOM permite a JavaScript reaccionar ante eventos HTML.

## Reaccionando a eventos

Un código JavaScript se puede ejecutar cuando ocurre un evento, como cuando un usuario hace clic en un elemento HTML.

Para ejecutar código cuando un usuario hace clic en un elemento, agregue código JavaScript a un atributo de evento HTML:

onclick=	<i>JavaScript</i>
----------	-------------------

Ejemplos de eventos HTML:

- Cuando un usuario hace clic con el mouse
- Cuando una página web se ha cargado
- Cuando una imagen ha sido cargada
- Cuando el mouse se mueve sobre un elemento
- Cuando se cambia un campo de entrada
- Cuando se envía un formulario HTML
- Cuando un usuario pulsa una tecla

En este ejemplo, el contenido del elemento `<h1>` se cambia cuando un usuario hace clic en él:

## Ejemplo

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="this.innerHTML = 'Oops!'">Click on this text!</h1>

</body>
</html>
```

Inténtalo tú mismo "

En este ejemplo, se llama a una función desde el controlador de eventos:

## Ejemplo

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {
  id.innerHTML = "Oops!";
}
</script>

</body>
</html>
```

Inténtalo tú mismo "

# Atributos de eventos HTML

Para asignar eventos a elementos HTML, puede usar atributos de eventos.

## Ejemplo

Asignar un evento onclick a un elemento de botón:

```
<button onclick="displayDate()">Try it</button>
```

Inténtalo tú mismo "

En el ejemplo anterior, se ejecutará una función llamada displayDate cuando se haga clic en el botón.

# Asigna eventos usando el HTML DOM

El HTML DOM le permite asignar eventos a elementos HTML usando JavaScript:

## Ejemplo

Asignar un evento onclick a un elemento de botón:

```
<script>  
document.getElementById("myBtn").onclick = displayDate;  
</script>
```

Inténtalo tú mismo "

En el ejemplo anterior, una función llamada displayDate se asigna a un elemento HTML con id = "myBtn".

La función se ejecutará cuando se haga clic en el botón.

# Los eventos onload y onunload

Los eventos onload y onunload se activan cuando el usuario ingresa o sale de la página.

El evento de carga se puede utilizar para verificar el tipo de navegador y la versión del navegador del visitante, y cargar la versión correcta de la página web en función de la información.

Los eventos onload y onunload se pueden usar para tratar con cookies.

## Ejemplo

```
| <| body onload="checkCookies()">
```

Inténtalo tú mismo "

# El evento onchange

El evento onchange a menudo se usa en combinación con la validación de campos de entrada.

A continuación se muestra un ejemplo de cómo usar el cambio. Se llamará a la función upperCase () cuando un usuario cambie el contenido de un campo de entrada.

## Ejemplo

```
| <| input type="text" id="fname" onchange="upperCase()">
```

Inténtalo tú mismo "

# Los eventos onmouseover y onmouseout

Los eventos onmouseover y onmouseout se pueden usar para activar una función cuando el usuario pasa el mouse sobre o fuera de un elemento HTML:

Ratón sobre mí

Inténtalo tú mismo "

# Los eventos onmousedown, onmouseup y onclick

Los eventos onmousedown, onmouseup y onclick son todos elementos de un clic del mouse. Primero, cuando se hace clic en el botón del mouse, se desencadena el evento onmousedown; luego, cuando se suelta el botón del mouse, se activa el evento onmouseup, finalmente, cuando se completa el clic del mouse, se activa el evento onclick.

Click Me

Inténtalo tú mismo "

# Más ejemplos

[onmousedown y onmouseup](#)

Cambia una imagen cuando un usuario mantiene presionado el botón del mouse.

[onload](#)

Muestra un cuadro de alerta cuando la página ha terminado de cargarse.

[onfocus](#)

Cambia el color de fondo de un campo de entrada cuando obtiene el foco.

[Eventos del mouse](#)

Cambia el color de un elemento cuando el cursor se mueve sobre él.

---

## Referencia de objeto de evento HTML DOM

Para obtener una lista de todos los eventos HTML DOM, consulte nuestra [Referencia](#) completa de [objetos de DOM de HTML](#) .

---

## Ponte a prueba con ejercicios!

[Ejercicio 1 »](#) [Ejercicio 2»](#) [Ejercicio 3 »](#)

# JavaScript HTML DOM EventListener

## El método addEventListener ()

### Ejemplo

Agregue un detector de eventos que se active cuando un usuario haga clic en un botón:

```
document.getElementById("myBtn").addEventListener("click", displayDate);
```

Inténtalo tú mismo "

El método addEventListener () adjunta un controlador de eventos al elemento especificado.

El método addEventListener () adjunta un controlador de eventos a un elemento sin sobrescribir los controladores de eventos existentes.

Puede agregar muchos manejadores de eventos a un elemento.

Puede agregar muchos manejadores de eventos del mismo tipo a un elemento, por ejemplo dos eventos de "click".

Puede agregar detectores de eventos a cualquier objeto DOM, no solo a elementos HTML, por ejemplo al objeto window.

El método addEventListener () facilita el control de cómo reacciona el evento al bubbling.

Al utilizar el método addEventListener (), el código JavaScript se separa del código HTML, para una mejor legibilidad y le permite agregar detectores de eventos incluso cuando no controla el código HTML.

Puede eliminar fácilmente un detector de eventos utilizando el método removeEventListener ().

# Sintaxis

`element. addEventListener(event, function, useCapture);`

El primer parámetro es el tipo del evento (como "click" o "mousedown").

El segundo parámetro es la función a la que queremos llamar cuando ocurre el evento.

El tercer parámetro es un valor booleano que especifica si se debe usar event bubbling o event capturing. Este parámetro es opcional.

Tenga en cuenta que no usa el prefijo "on" para el evento; use "click" en lugar de "onclick".



# Agregar un controlador de eventos a un elemento

## Ejemplo

Alerta "¡Hola mundo!" cuando el usuario hace clic en un elemento:

```
| element|.addEventListener("click", function(){ alert("Hello World!"); });
```

Inténtalo tú mismo "

También puede referirse a una función externa "nombrada":

## Ejemplo

Alerta "¡Hola mundo!" cuando el usuario hace clic en un elemento:

```
| element|.addEventListener("click", myFunction);
```

```
| function myFunction() {  
|   alert ("Hello World!");  
| }
```

Inténtalo tú mismo "

# Agregue muchos manejadores de eventos al mismo elemento

El método `addEventListener()` le permite agregar muchos eventos al mismo elemento, sin sobrescribir eventos existentes:

## Ejemplo

```
element.addEventListener("click", myFunction);  
element.addEventListener("click", mySecondFunction);
```

Inténtalo tú mismo "

Puede agregar eventos de diferentes tipos al mismo elemento:

## Ejemplo

```
element.addEventListener("mouseover", myFunction);  
element.addEventListener("click", mySecondFunction);  
element.addEventListener("mouseout", myThirdFunction);
```

Inténtalo tú mismo "

# Agregar un controlador de eventos al objeto window

El método `addEventListener()` le permite agregar detectores de eventos en cualquier objeto HTML DOM como los elementos HTML, el documento HTML, el objeto `window` u otros objetos que admitan eventos, como el objeto `xmlHttpRequest`.

## Ejemplo

Agregue un detector de eventos que se active cuando un usuario cambie el tamaño de la ventana:

```

| window.addEventListener("resize", function(){
|     document.getElementById("demo").innerHTML = sometext;
| });
```

Inténtalo tú mismo "

# Pasar parámetros

Al pasar valores de parámetros, use una "función anónima" que llame a la función especificada con los parámetros:

## Ejemplo

```
element.addEventListener("click", function(){ myFunction(p1, p2); });
```

Inténtalo tú mismo "

# Event Bubbling or Event Capturing?

Hay dos formas de propagación de eventos en HTML DOM, event bubbling y event capturing.

La propagación de eventos es una forma de definir el orden de los elementos cuando ocurre un evento. Si tiene un elemento `<p>` dentro de un elemento `<div>`, y el usuario hace clic en el elemento `<p>`, ¿qué evento de "clic" de elemento se debe manejar primero?

Con event bubbling, el evento del elemento más interno se maneja primero y luego el externo: el evento de clic del elemento `<p>` se maneja primero, luego el evento click del elemento `<div>`.

Con event capturing, el evento del elemento externo se maneja primero y luego el interno: el evento click del elemento `<div>` se manejará primero, luego el evento click del elemento `<p>`.

Con el método `addEventListener()` puede especificar el tipo de propagación utilizando el parámetro `"useCapture"`:

```
addEventListener( event, function, useCapture);
```

El valor predeterminado es falso, que utilizará la propagación de propagación, cuando el valor se establece en verdadero, el evento usa la propagación de captura.

## Ejemplo

```
document.getElementById("myP").addEventListener("click",  
myFunction, true);  
document.getElementById("myDiv").addEventListener("click",  
myFunction, true);
```

Inténtalo tú mismo "

# El método removeEventListener ()

El método removeEventListener () elimina los controladores de eventos que se han adjuntado con el método addEventListener ():






## Ejemplo

```
element.removeEventListener("mousemove", myFunction);
```

Inténtalo tú mismo "

# Soporte del navegador

Los números en la tabla especifican la primera versión del navegador que admite completamente estos métodos.

Method					
<code>addEventListener()</code>	1.0	9.0	1.0	1.0	7.0
<code>removeEventListener()</code>	1.0	9.0	1.0	1.0	7.0

Nota: Los métodos `addEventListener()` y `removeEventListener()` no son compatibles con IE 8 y versiones anteriores, Opera 6.0 y versiones anteriores. Sin embargo, para estas versiones de navegador específicas, puede usar el método `attachEvent()` para adjuntar un manejador de eventos al elemento y el método `detachEvent()` para eliminarlo:

```
element. attachEvent(event, function);
```

```
element.detachEvent(event, function);
```

## Ejemplo

Solución de navegador cruzado:

```
var x = document.getElementById("myBtn");
if (x.addEventListener) {           // For all major browsers, except IE 8
    and earlier
    x.addEventListener("click", myFunction);
} else if (x.attachEvent) {        // For IE 8 and earlier versions
    x.attachEvent("onclick", myFunction);
}
```

Inténtalo tú mismo "

# Referencia de objeto de evento HTML DOM

Para obtener una lista de todos los eventos HTML DOM, consulte nuestra [Referencia](#) completa de [objetos de DOM de HTML](#) .