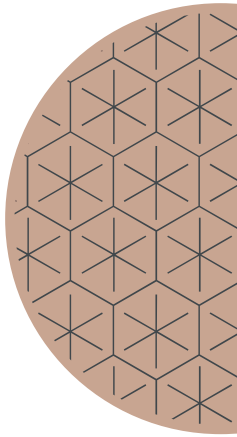
Gestión de eventos y formularios en JavaScript.

Gestión de eventos y formularios en JavaScript

En esta unidad veremos los siguientes conceptos:

- Eventos
- Objeto Event.
- Event handler y event Listener.
- Eventos más comunes.
- Validación de formularios.



Eventos

Eventos

¿Qué son los eventos? Bueno, los eventos de JavaScript son simplemente acciones que puede realizar el usuario o incluso el propio navegador. Cuando abrimos una página web, se producen muchos eventos, pero no somos conscientes de ellos.

Por ejemplo, se abre una página web, y se tarda un tiempo hasta que esa página termina de cargar su contenido. Otro ejemplo de evento es cuando un usuario hace clic en el botón de envío para enviar el formulario de registro.

Eventos

Otro ejemplo de evento es cuando un usuario hace clic en el menú del sitio web y obtiene la lista desplegable de elementos del submenú. La barra de búsqueda del sitio web es otro evento común. Cada vez que un usuario pulsa la tecla en su teclado, aparece el cuadro de autocompletar con útiles sugerencias.

Los eventos son una característica crucial de JavaScript. Sin ellos, nuestra página será aburrida y no interactiva.

Objeto evento

Todos los objetos asociados a las acciones heredan del objeto Event. Algunas de los métodos más útiles que tiene son los siguientes:

stopPropagation(): Evita la propagación de un evento durante el flujo de eventos

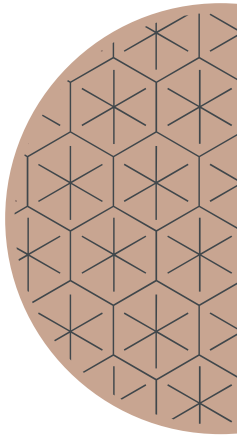
target: Devuelve el elemento que desencadenó el evento.

currentTarget: Devuelve el elemento cuyos oyentes de eventos desencadenaron el evento

Event Handler & Event Listener

Event Handler & Event Listener

Dos conceptos muy importantes cuando se trata de eventos en JavaScript. El manejador de eventos es simplemente una función. Esta función de JavaScript se ejecutará tan pronto como se active el evento.



Event Handler & Event Listener

Por otro lado, el escuchador de eventos se adjunta a un elemento en particular, y esperará y escuchará cualquier cambio que pueda ocurrir en ese elemento en particular.

Hay tres formas de asignar eventos a los elementos del DOM:

- Manejadores de eventos en línea
- Propiedades de los manejadores de eventos
- Escuchadores de eventos

Inline Event Handlers

Esto es lo más básico cuando se trata de manejadores de eventos.

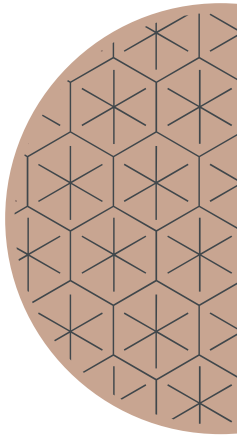
Id al GitHub y buscad el ejemplo de inline.

Añadir eventos de esta forma es fácil y sencillo, pero generalmente, no son muy utilizados hoy en día.

Event Handler Properties

La propiedad del manejador de eventos es muy similar a los eventos inline, con una sola excepción.

No necesitamos asociar directamente ningún atributo y valores a nuestro elemento HTML. Podemos conseguir el mismo resultado que en el ejemplo anterior con escribir las pocas líneas de código en nuestro archivo JavaScript; por lo tanto, esta propiedad inline ya no será necesaria dentro del botón.



Event Handler Properties

Ejercicio) Qué pasa cuando agregamos el archivo js en el head?

Event Listeners

Hasta ahora, hemos aprendido a manejar eventos de dos maneras diferentes, y la última es muy importante y se trata de los escuchadores de eventos.

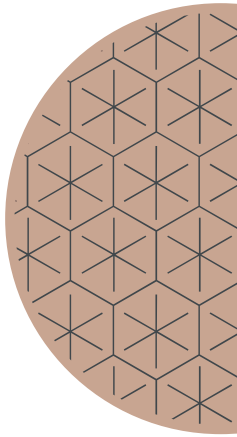
Los escuchadores de eventos escuchan un evento particular que ocurre en un elemento. Así que no vamos a asignar un evento directamente a un elemento, sino que utilizaremos el método addEventListener() que escuchará el evento que ocurra. El método addEventListener() puede tomar dos parámetros, el primero es el evento que estamos escuchando, y el segundo parámetro será la función callback.

Eventos comunes

Eventos más comunes de JavaScript

Hasta ahora, hemos cubierto las tres formas diferentes de manejar eventos en JavaScript, pero hay muchos más eventos en JavaScript que vale la pena mencionar.

No crearé ejemplos separados para ellos, pero crearé tablas para leer el tipo de evento y lo que hacen.



Eventos más comunes de JavaScript

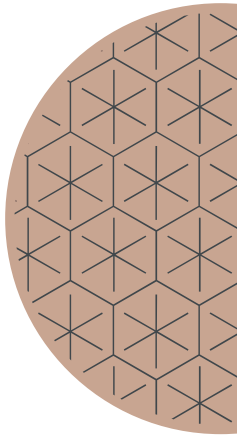
Como desarrollador profesional, necesitarás conocer los eventos del ratón porque son los que más utilizamos. Cada vez que movemos el ratón o hacemos clic en él, estamos realizando alguna acción, y todas las acciones que hacemos como usuarios son posibles eventos en JavaScript.

Eventos ratón

Event	Description
click	This event fires only when the mouse button is pressed and released
dbClick	This event fires only when an element is clicked twice
mousemove	This event fires only when a mouse pointer movies inside a particular element
mouseenter	This event fires only when a mouse pointer enters a particular element
mouseleave	This event fires only when a mouse pointer leaves the particular element

Eventos más comunes de JavaScript

El evento click es una combinación de dos eventos separados como los eventos mousedown y mouseup, y esto sucede cuando un usuario está presionando el botón del ratón o levantando el botón del ratón. Cuando usamos el mouseenter y el mouseleve, básicamente estamos recreando el efecto hover cuando tenemos el ratón sobre algún elemento.



Eventos teclado

Al igual que los eventos de ratón, tenemos eventos de teclado, y estos eventos se disparan cada vez que el usuario pulsa, levanta o mantiene pulsada una tecla concreta.

Event	Description
keypress	This event fires continuously as long as the key is pressed
keydown	This event fires only once when a key is pressed down
keyup	This event fires only once when a key is released

Eventos formulario

Cuando un usuario interactúa con un formulario, hay ciertos eventos que debemos tener en cuenta

Event	Description
focus	This event fires when an element like input field receives focus
submit	This event fires when an html form is submitted
blur	This event fires only when an element loses its focus

Validación formularios

Validación de formularios

Cuando se introducen datos, el navegador y/o el servidor web comprueban que los datos están en el formato correcto y dentro de las restricciones establecidas por la aplicación.

La validación realizada en el navegador se llama validación del lado del cliente, mientras que la validación realizada en el servidor se llama validación del lado del servidor. En esta unidad nos centraremos en la validación del lado del cliente.

Tipos de validación lado cliente

Hay dos tipos diferentes de validación del lado del cliente que encontrarás en la web:

- **La validación integrada de HTML**: se realiza usando atributos del input ,tiene mejor rendimiento que la de JavaScript, pero no es tan personalizable como la validación de JavaScript.
- **La validación de JavaScript** se codifica usando JavaScript. Esta validación es completamente personalizable, pero necesitas crearla entera.

Validación de formularios integrada

Se realiza utilizando atributos de validación en los inputs.

- **required:** Especifica si es necesario rellenar un campo del formulario antes de enviarlo.
- **minlength y maxlength:** Especifica la longitud mínima y máxima de los datos textuales (cadenas).
- **min y max:** Especifica los valores mínimos y máximos de los tipos de entrada numérica.
- **type:** Especifica si los datos deben ser un número, una dirección de correo electrónico o algún otro tipo específico preestablecido.
- **pattern:** Especifica una expresión regular que define un patrón que deben seguir los datos introducidos.

Validación de formularios mediante JavaScript

La mayoría de los navegadores admiten la API de validación de restricciones, que consta de un conjunto de métodos y propiedades disponibles en las siguientes interfaces DOM de elementos de formulario:

- HTMLButtonElement (representa un elemento <button>)
- HTMLFieldSetElement (representa un elemento <fieldset>)
- HTMLInputElement (representa un elemento <input>)
- HTMLOutputElement (representa un elemento <output>)
- HTMLSelectElement (representa un elemento <select>)
- HTMLTextAreaElement (representa un elemento <textarea>)

Validación de formularios mediante JavaScript

La API de validación de restricciones pone a disposición de los elementos anteriores las siguientes propiedades.

- **validationMessage**: Devuelve un mensaje localizado que describe las restricciones de validación que el control no satisface (si las hay). Si el control no es candidato a la validación de restricciones (willValidate es falso) o el valor del elemento satisface sus restricciones (es válido), esto devolverá una cadena vacía.

Validación de formularios mediante JavaScript

validity: Devuelve un objeto ValidityState que contiene varias propiedades que describen el estado de validez del elemento. A continuación se enumeran algunas de las más comunes:

- **patternMismatch:** Devuelve true si el valor no coincide con el patrón especificado, y false si coincide. Si es verdadero, el elemento coincide con la pseudoclase CSS :invalid.
- **tooLong:** Devuelve true si el valor es más largo que la longitud máxima especificada por el atributo maxlength, o false si es más corto o igual al máximo. Si es verdadero, el elemento coincide con la pseudoclase :invalid CSS.

Validación de formularios mediante JavaScript

- **tooShort**: Devuelve verdadero si el valor es más corto que la longitud mínima especificada por el atributo minlength, o falso si es mayor o igual que el mínimo. Si es verdadero, el elemento coincide con la pseudoclase CSS inválida.
- **rangeOverflow**: Devuelve true si el valor es mayor que el máximo especificado por el atributo max, o false si es menor o igual que el máximo. Si es verdadero, el elemento coincide con las pseudoclases CSS :invalid y :out-of-range.

Validación de formularios mediante JavaScript

- **rangeUnderflow**: Devuelve true si el valor es menor que el mínimo especificado por el atributo min, o false si es mayor o igual que el mínimo. Si es verdadero, el elemento coincide con las pseudoclases CSS :invalid y :out-of-range.
- **typeMismatch**: Devuelve true si el valor no está en la sintaxis requerida (cuando el tipo es email o url), o false si la sintaxis es correcta. Si es true, el elemento coincide con la pseudoclase CSS :invalid.

Validación de formularios mediante JavaScript

- **valid**: Devuelve true si el elemento cumple con todas sus restricciones de validación, y por lo tanto se considera válido, o false si falla alguna restricción. Si es verdadero, el elemento coincide con la pseudoclase CSS válida; en caso contrario, con la pseudoclase CSS inválida.
- **valueMissing**: Devuelve true si el elemento tiene un atributo requerido, pero no tiene valor, o false en caso contrario. Si es true, el elemento coincide con la pseudoclase CSS inválida.

Validación de formularios mediante JavaScript

La API de validación de restricciones también pone a disposición los siguientes métodos en los elementos anteriores y en el elemento formulario.

- **checkValidity():** Devuelve true si el valor del elemento no tiene problemas de validez; false en caso contrario. Si el elemento es inválido, este método también dispara un evento inválido en el elemento.
- **reportValidity():** Informa de los campos no válidos mediante eventos. Útil en combinación con preventDefault() en un manejador de eventos onSubmit
- **setCustomValidity(message):** Añade un mensaje de error personalizado al elemento.Esto permite utilizar código JavaScript para establecer un fallo de validación distinto a los ofrecidos por las restricciones de validación HTML estándar.

FIN