Eventos:

- **load**: Se activa cuando la página y todos los recursos han sido completamente cargados.
- **submit**: Se activa cuando se envía un formulario.
- **scroll**: Se activa cuando la ventana o un elemento con barra de desplazamiento se desplaza.
- **focus**: Se activa cuando un elemento gana el foco.

Eventos de ratón:

- **click**: Se activa cuando se hace clic en un elemento.
- **dblclick**: Se activa cuando se hace doble clic en un elemento.
- **mousedown**: Se activa cuando se presiona un botón del ratón sobre un elemento.
- **mousemove**: Se activa cuando el ratón se mueve dentro de un elemento.
- **mouseup**: Se activa cuando se libera un botón del ratón sobre un elemento.
- **mouseover**: Se activa cuando el ratón entra en el área de un elemento.

Eventos de teclado:

- **keydown**: Se activa cuando se presiona una tecla. Es útil para capturar eventos en el momento exacto en que se presionan las teclas.
- **keyup**: Se activa cuando se libera una tecla. Es útil para capturar eventos después de que se ha terminado de presionar una tecla.

Eventos arrastrar y soltar:

1. establecer el atributo draggable=”true”.
2. capturar el evento drop.
3. invalidar el elemento dragover sobre el destino.

- **dragstart**: Cuando el arrastre comienza.
- **drag**: Continuamente durante el arrastre.
- **dragend**: Cuando el arrastre termina.
- **dragenter**: Cuando el elemento entra en una zona de destino.
- **dragover**: Cuando el elemento es arrastrado sobre una zona de destino (necesita `event.preventDefault()`).
- **dragleave**: Cuando el elemento sale de una zona de destino.
- **drop**: Cuando el elemento es soltado en una zona de destino (necesita `event.preventDefault()`).

- DataTransfer:

```
const caja1 = document.querySelector('.caja1');
const caja2 = document.querySelector('.caja2');
caja1.addEventListener('dragstart', function (e) {
  e.dataTransfer.setData('text', 'String enviado!!!');
});
caja2.addEventListener('drop', function (e) {
  e.preventDefault();
  console.log(e.dataTransfer.getData('text'));
});
caja2.addEventListener('dragover', function (e) {
  e.preventDefault();
});
```

x.addEventListener(“event”, function());
En el manejador de evento se puede incluir un objeto Event como primer parámetro

- target: elemento que produjo en el evento.
- currentTarget: elemento donde se captura.
- type: tipo de evento
- clientX, clientY: coordenadas
- key: tecla pulsada

Para evitar la propagación tendremos que poner event.stopPropagation();, solo si hay eventos en los padres.

FORMULARIOS:

Acceso a Formularios y Elementos

- **forms[]:** Array que contiene los formularios del documento.
    - Formas estándar de acceder:
        - document.forms[i]
        - document.getElementById("x")
        - document.querySelector("x")
- **elements[]:** Array de controles (input, textarea, select) de cada formulario.
    - Acceso a cada control:
        - foo.elements[i] o foo.elements['email']
        - foo[i] o foo['email']
    - También se pueden usar querySelector() o getElementById().

Controles de Formularios

- **input:** Tipos (type): text, button, checkbox, radio, submit, etc.
- **Textarea:** Para textos de varias líneas (cols, rows).
- **Select:** Listas desplegables (select con subetiquetas option).

Propiedades de Elementos de Formulario

- **type:** Indica el tipo de control (text, checkbox, button, etc.).
- **name:** Nombre del campo (variable enviada al servidor).
- **Value:** Valor del atributo value o del texto escrito por el usuario.
- **Checked:** Indica si un checkbox o radio está seleccionado.
- **selected:** Indica la opción seleccionada en un control select.

Eventos en Formularios

- **focus:** Cuando un elemento obtiene el foco.
- **blur:** Cuando un elemento pierde el foco.
- **change:** Cuando el usuario modifica el campo y pierde el foco.
- **input:** Cuando cambia el contenido de un elemento de tipo texto.
- **cut/copy/paste:** Acciones de cortar, copiar y pegar.
- **click:** Cuando se hace clic en cualquier elemento.
- **submit:** Cuando se envía el formulario (clic en submit o enter en input).

Enviar un Formulario

- **Control Attributes:**
  - Action: URL a la que se envían los datos.
  - method: Método de envío (GET o POST).
- **Enviar Programa:** También se puede enviar con form.submit().

Obtener Valores de Campos

- **Input y textarea:** Propiedad value.
- **RadioButtons:** Propiedad value del grupo, checked para estado.
- **Checkbox:** Acceso como array de nodos (form['name']), value, checked.
- **Listas desplegables:** Usar selectedIndex, array de nodos con value y selected.

Más Controles

- **Input Attributes:** Tipos adicionales como password, image, hidden, file, color, date, time, tel, email, range, etc.

Validación de Formularios

- HTML5 Attributes: minlength, maxlength, min, max, step, required, disabled, readonly, pattern.
- CSS Pseudoselectors: :valid, :invalid, :required.
- JavaScript Validation: checkValidity(), setCustomValidity().

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form Validation Example</title>
</head>
<body>
    <form id="myForm">
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <button type="submit">Submit</button>
    </form>

    <script>
        document.getElementById('myForm').addEventListener('submit', function(event) {
            const emailInput = document.getElementById('email');

            // Reiniciar cualquier mensaje de error personalizado
            emailInput.setCustomValidity('');

            // Validar el campo de correo electrónico
            if (!emailInput.checkValidity()) {
                // Si el campo no es válido, establecer un mensaje de error personalizado
                emailInput.setCustomValidity('Por favor, introduce una dirección de correo electrónico válida.');

                // Prevenir el envío del formulario
                event.preventDefault();
            }
        });

        // Añadir un listener para mostrar los mensajes de error personalizados
        document.getElementById('email').addEventListener('input', function() {
            this.setCustomValidity(''); // Resetear mensaje de error al cambiar el valor
        });
    </script>
</body>
</html>
```