
UNIVERSIDAD ESTATAL AMAZÓNICA
VITA ET SAPIENTIA
UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

DESARROLLO DE APLICACIONES WEB

Unidad 2

Desarrollo del Lado del Cliente

Tema 2.1: JavaScript para aplicaciones web

Subtema 2.1.1: Fundamentos de JavaScript: manipulación del DOM y eventos.

DESARROLLO DE APLICACIONES WEB

Transformamos el mundo desde la Amazonía

Ing. Walter Rodrigo Núñez Zamora, Mgs.
DOCENTE - PERSONAL ACADÉMICO NO TITULAR OCASIONAL

UNIVERSIDAD ESTATAL AMAZÓNICA

SEMANA

5

DESARROLLO DE LA SEMANA 5

DEL LUN. 05 AL DOM. 11 DE ENERO / 2026

Resultado de aprendizaje: Aplicar técnicas de diseño de interfaces y desarrollo frontend y backend para implementar funcionalidades dinámicas y gestionar datos de manera eficiente.

CONTENIDOS

UNIDAD II : Desarrollo del lado del cliente

- Tema 2.1: JavaScript para aplicaciones web
  - Subtema 2.1.1: Fundamentos de JavaScript: manipulación del DOM y eventos

Subtema 2.1.1 ” Fundamentos de JavaScript: manipulación del DOM y eventos.”

DESARROLLO WEB

CARRERAS EN LÍNEA

¿Qué es JavaScript?

JavaScript (JS) es un lenguaje de programación de alto nivel, interpretado y basado en prototipos, diseñado principalmente para crear interactividad y dinamismo en páginas web. Es un componente esencial del desarrollo web junto con HTML y CSS.

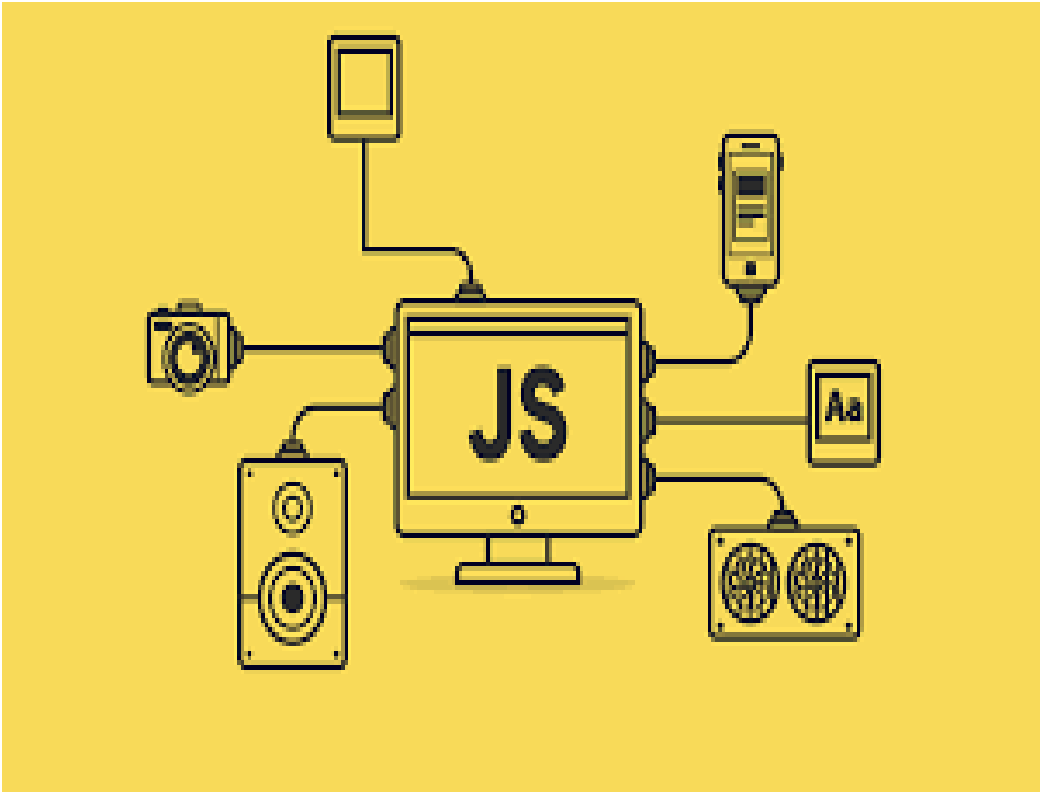
CARRERAS EN LÍNEA

Características de JavaScript?

1. **Lenguaje del lado del cliente**:
   Corre directamente en el navegador, permitiendo interactividad en tiempo real.
   Ejemplo: Validación de formularios, menús dinámicos, efectos visuales.
2. **Lenguaje del lado del servidor** (Node.js):
   Permite construir aplicaciones web completas, manejando la lógica del servidor y las bases de datos.
3. **Dinámico y flexible**:
   Soporta paradigmas de programación como la orientación a objetos, funcional y basada en eventos.
4. **Multipropósito**:
   Desde desarrollo web, aplicaciones móviles y de escritorio, hasta programación para IoT (Internet de las cosas).
5. **Compatibilidad**:
   Es compatible con todos los navegadores modernos.

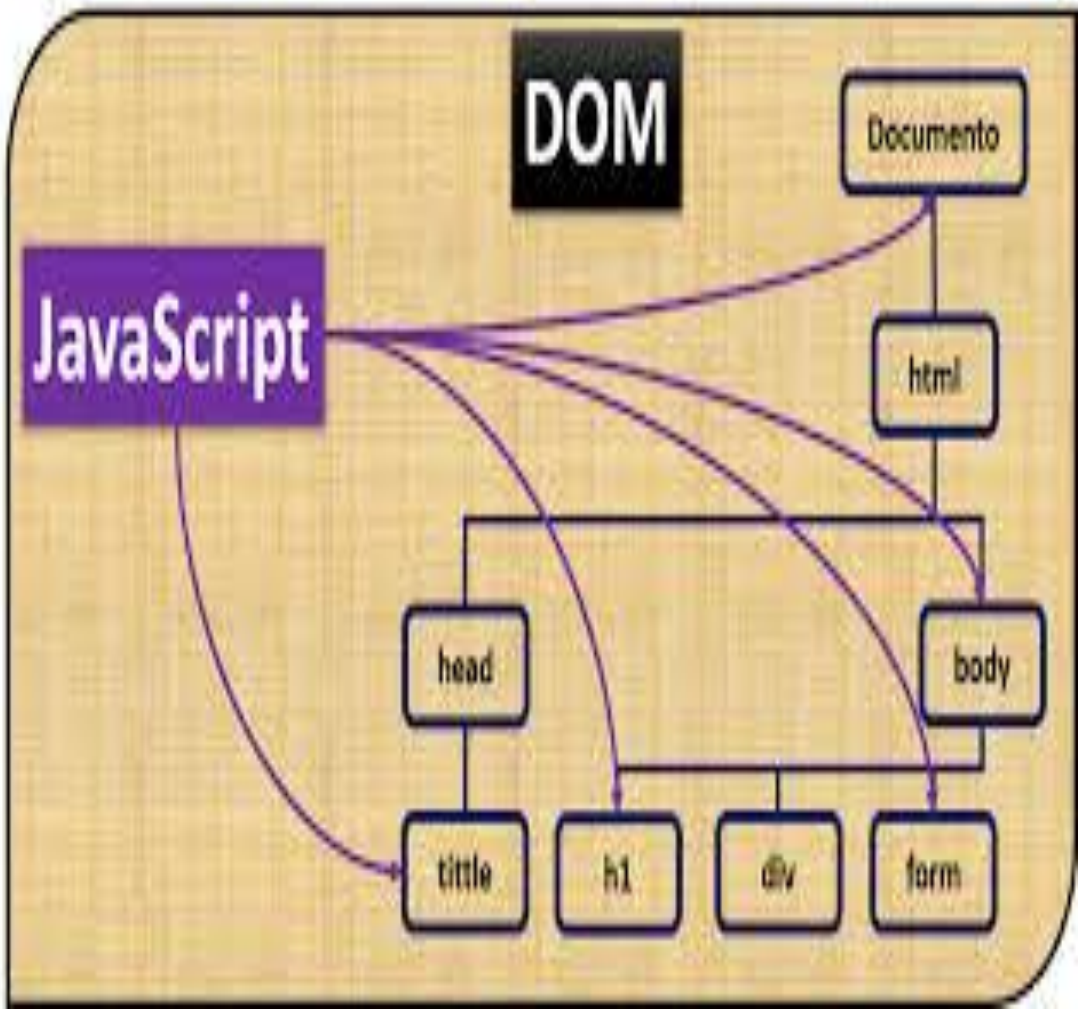
Ejemplo de JavaScript

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Ejemplo JavaScript</title>
</head>
<body>
   <h1 id="titulo">Hola, Mundo</h1>
   <button onclick="cambiarTexto()">Haz clic aquí</button>

   <script>
      function cambiarTexto() {
         document.getElementById("titulo").innerText = "¡Texto cambiado con
JavaScript!";
      }
   </script>
</body>
</html>
```

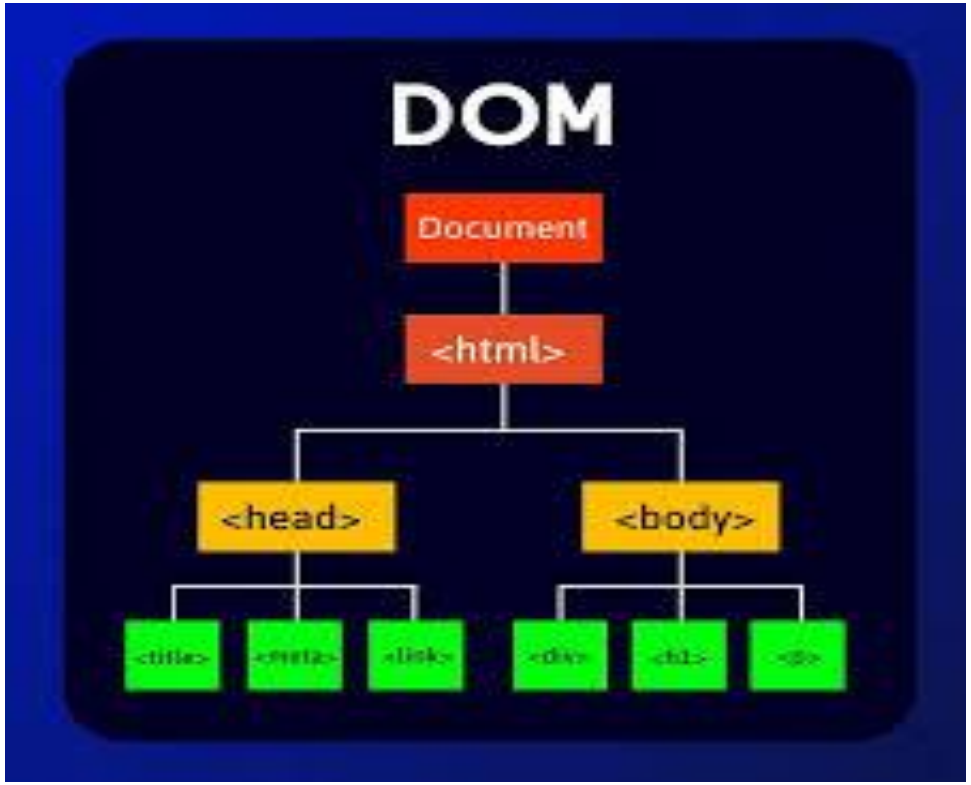


¿Qué es el DOM?

El DOM (Document Object Model) es una representación estructurada de un documento HTML o XML, que el navegador crea cuando carga una página web. Es una interfaz que permite a los lenguajes de programación, como JavaScript, interactuar con el contenido, estructura y estilos de una página.



Características del DOM JavaScript



1. **Estructura jerárquica:** Representa los elementos de la página como un árbol de nodos.
   - Cada etiqueta HTML es un nodo en el árbol.
2. **Interactividad:** Permite leer, modificar, añadir o eliminar elementos y atributos en tiempo real.
3. **Acceso programático:** Los lenguajes como JavaScript pueden manipular el DOM para actualizar la página sin recargarla.

Ejemplo de JavaScript

El DOM es el puente entre el HTML y JavaScript, permitiendo que las páginas sean dinámicas e interactivas.

Html:

```
<!DOCTYPE html>
<html>
<body>
   <h1 id="titulo">Hola, Mundo</h1>
   <button onclick="cambiarTexto()">Clic aquí</button>

   <script>
      function cambiarTexto() {
         document.getElementById("titulo").innerText = "¡Texto
modificado!";
      }
   </script>
</body>
</html>
```

Acceso al DOM con JavaScript

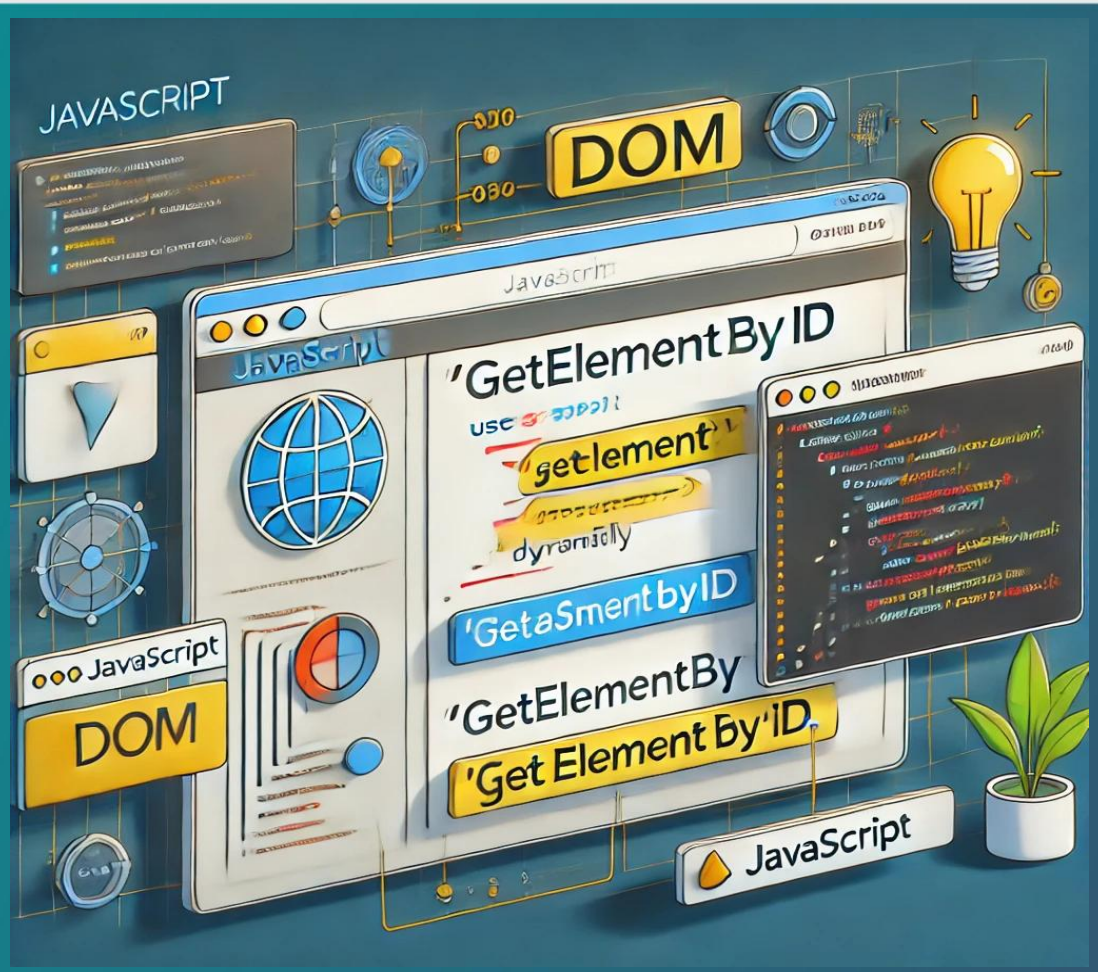
JavaScript utiliza el **DOM** para interactuar con los elementos de una página web. Esto permite acceder, modificar y actualizar dinámicamente el contenido y la estructura de la página.

1. **getElementById**

Accede a un elemento por su atributo id.

javascript

```
const titulo = document.getElementById("titulo");
titulo.innerText = "Nuevo texto";
```



Acceso al DOM con JavaScript

2. getElementsByClassName
Selecciona elementos que comparten una clase.

```
const elementos = document.getElementsByClassName("mi-clase"); elementos[0].style.color = "blue"; // Cambia el color del primer elemento
```

3. getElementsByTagName
Obtiene elementos por su etiqueta.

```
const parrafos = document.getElementsByTagName("p"); parrafos[0].innerHTML = "Texto actualizado"; // Modifica el primer párrafo
```

4. querySelector
Selecciona el primer elemento que coincida con un selector CSS.

```
const boton = document.querySelector(".boton-primario"); boton.style.backgroundColor = "red";
```

5. querySelectorAll
Selecciona todos los elementos que coincidan con un selector CSS.

```
const elementos = document.querySelectorAll(".elemento"); elementos.forEach(el => el.style.fontSize = "18px");
```

Modificación de Elementos del DOM

JavaScript permite modificar dinámicamente los elementos del DOM para cambiar el contenido, estilos, atributos o incluso la estructura de una página web

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Modificaciones del DOM</title>
</head>
<body>
   <h1 id="titulo">Texto original</h1>
   <img id="imagen" src="placeholder.jpg" alt="Imagen de
ejemplo">
   <button id="boton">Realizar cambios</button>
</body>
</html>
```

```
<script>
    document.getElementById("boton").addEventListener("click", () => {
      const titulo = document.getElementById("titulo");
      const imagen = document.getElementById("imagen");

      // Modificar texto o HTML
      titulo.innerText = "Nuevo texto"; // Solo texto
      titulo.innerHTML = "<b>Texto con formato</b>"; // Incluye etiquetas

      // Cambiar estilos
      titulo.style.color = "blue"; // Cambia el color
      titulo.style.fontSize = "18px"; // Cambia el tamaño de la fuente

      // Modificar atributos
      imagen.setAttribute("src", "nueva-imagen.jpg"); // Cambia el atributo src
      imagen.removeAttribute("alt"); // Elimina el atributo alt

      // Manejar clases
      titulo.classList.add("activo"); // Agregar clase
      titulo.classList.remove("activo"); // Quitar clase

      // Crear y eliminar elementos
      const nuevo = document.createElement("p"); // Crear un párrafo
      nuevo.innerText = "Este es un párrafo nuevo."; // Asignar contenido
      document.body.appendChild(nuevo); // Agregarlo al final del body
      titulo.remove(); // Eliminar el título
    });
  </script>
```

Creación de Elementos del DOM

La creación de elementos en el DOM permite agregar contenido dinámico a una página web, como nuevos párrafos, botones, imágenes o cualquier otro elemento HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Crear Elementos del DOM</title>
</head>
<body>
   <div id="contenedor">
      <h1>Título principal</h1>
   </div>
   <button id="crear">Crear Elemento</button>

</body>
</html>
```

```
<script>
    document.getElementById("crear").addEventListener("click", () => {
       // Crear un nuevo elemento (párrafo)
       const nuevoParrafo = document.createElement("p");
       nuevoParrafo.innerText = "Este es un párrafo nuevo.";

       // Establecer estilos al nuevo elemento
       nuevoParrafo.style.color = "blue";
       nuevoParrafo.style.fontSize = "16px";

       // Agregar el párrafo al contenedor
       const contenedor = document.getElementById("contenedor");
       contenedor.appendChild(nuevoParrafo);

       // Crear un botón adicional
       const nuevoBoton = document.createElement("button");
       nuevoBoton.innerText = "Botón dinámico";
       nuevoBoton.style.marginTop = "10px";

       // Agregar funcionalidad al botón
       nuevoBoton.addEventListener("click", () => {
          alert("¡Hiciste clic en el botón dinámico!");
       });

       // Agregar el botón al contenedor
       contenedor.appendChild(nuevoBoton);
    });
</script>
```

Eliminación de Elementos del DOM

JavaScript permite eliminar elementos del DOM para ajustar dinámicamente la estructura de la página. Esto es útil para ocultar, reemplazar o limpiar contenido de forma programada.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Eliminar Elementos del DOM</title>
</head>
<body>
   <div id="contenedor">
      <h1 id="titulo">Este es un título</h1>
      <p id="parrafo">Este es un párrafo.</p>
      <button id="eliminarTitulo">Eliminar Título</button>
      <button id="eliminarParrafo">Eliminar Párrafo</button>
   </div>

</body>
</html>
```

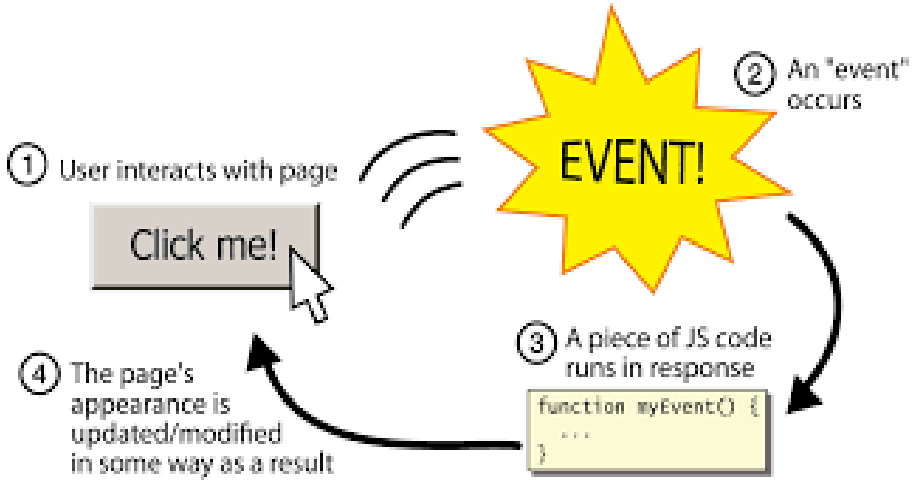
```
<script>
    // Eliminar el título con el método remove()
    document.getElementById("eliminarTitulo").addEventListener("click", () => {
      const titulo = document.getElementById("titulo");
      titulo.remove(); // Elimina el título
    });

    // Eliminar el párrafo con el método removeChild()
    document.getElementById("eliminarParrafo").addEventListener("click", () => {
      const contenedor = document.getElementById("contenedor");
      const parrafo = document.getElementById("parrafo");
      contenedor.removeChild(parrafo); // Elimina el párrafo
    });
  </script>
```

Introducción a Eventos

Los eventos son acciones o sucesos que ocurren en el navegador o en una página web, como hacer clic en un botón, mover el mouse, escribir en un campo de texto, o cargar una página. JavaScript permite reaccionar a estos eventos para crear aplicaciones web interactivas.

Cómo funcionan los eventos

1. **Evento**: Es la acción que se produce (por ejemplo, un clic).
2. **Manejador de eventos (Event Listener)**: Es el código que responde al evento.



Ejemplo practico eventos

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Eventos en JavaScript</title>
</head>
<body>
    <button id="miBoton">Haz clic aquí</button>
    <input type="text" id="miCampo" placeholder="Escribe algo">
    <p id="mensaje"></p>
</body>
</html>
```

CARRERAS EN LÍNEA

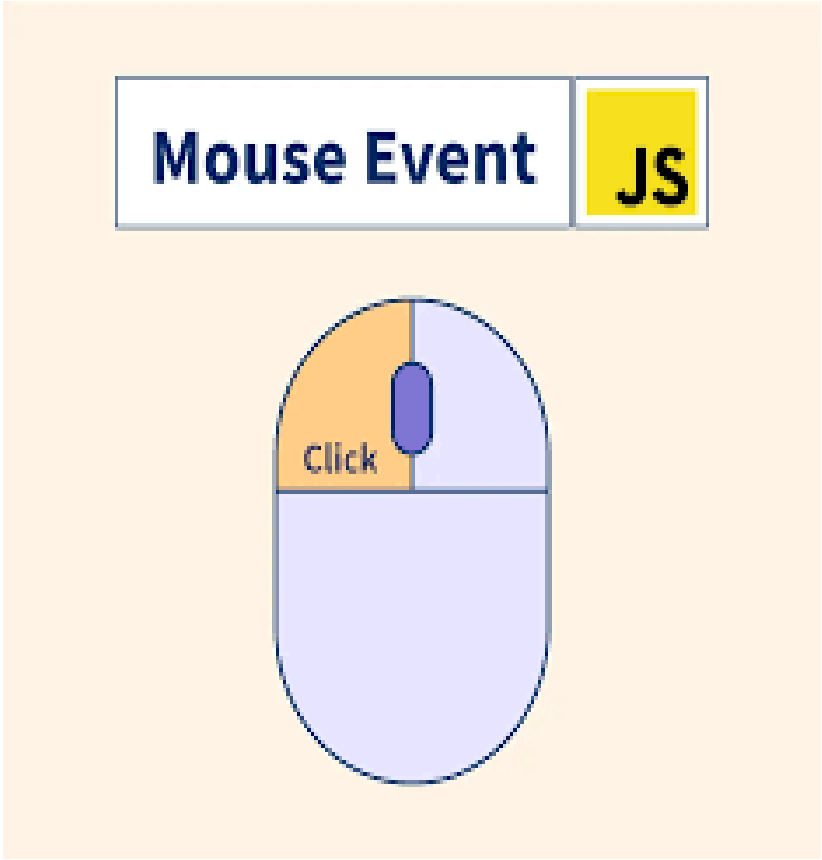
```
<script>
    // Evento de clic
    const boton = document.getElementById("miBoton");
    boton.addEventListener("click", () => {
        alert("¡Botón clickeado!");
    });

    // Evento de teclado
    const campo = document.getElementById("miCampo");
    campo.addEventListener("keydown", (event) => {
        const mensaje = document.getElementById("mensaje");
        mensaje.innerText = `Tecla presionada: ${event.key}`;
    });
</script>
```

Tipos Comunes de Eventos en JavaScript

1. Eventos del Mouse

Se desencadenan por interacciones con el mouse.

- **click:** Cuando se hace clic en un elemento.
- **dblclick:** Cuando se hace doble clic.
- **mouseover:** Cuando el puntero pasa sobre un elemento.
- **mouseout:** Cuando el puntero abandona un elemento.
- **mousedown:** Cuando se presiona un botón del mouse.
- **mouseup:** Cuando se suelta un botón del mouse.



Ejemplo:

```
const boton = document.getElementById("miBoton");
boton.addEventListener("click", () => { alert("Hiciste clic en el botón"); });
```

Tipos Comunes de Eventos en JavaScript

2. Eventos del Teclado

Detectan cuando el usuario interactúa con el teclado.

- **keydown:** Cuando una tecla es presionada.
- **keyup:** Cuando una tecla es liberada.
- **keypress:** (Obsoleto) Similar a keydown, pero solo para caracteres imprimibles.

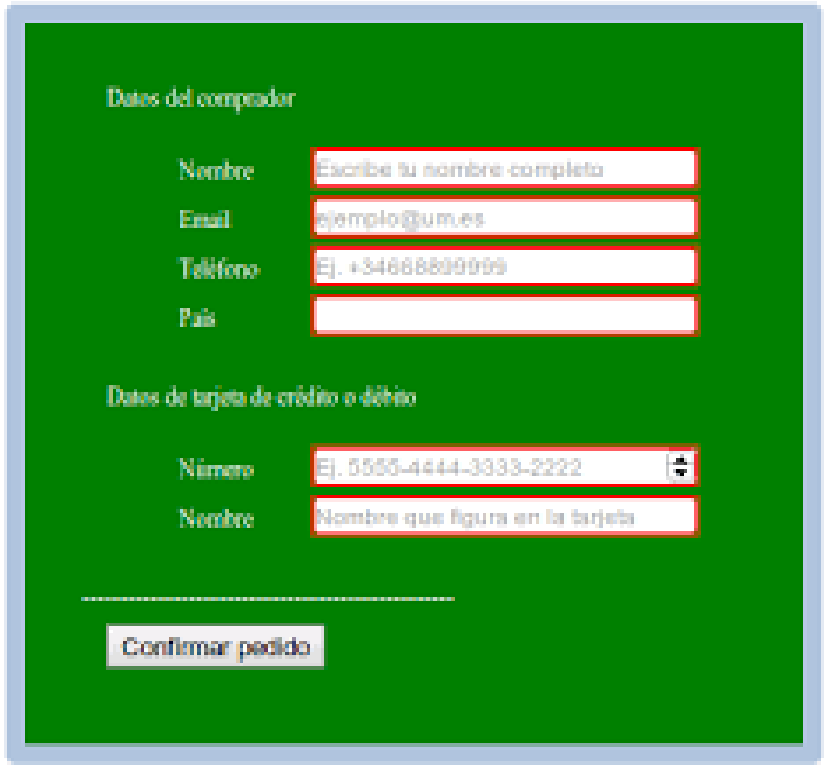
Ejemplo:
```
document.addEventListener("keydown", (event) => {
console.log(`Tecla presionada: ${event.key}`); });
```

Tipos Comunes de Eventos en JavaScript

3. Eventos de Formulario

Se activan en elementos como inputs, selects y formularios.

- **submit:** Cuando se envía un formulario.
- **change:** Cuando el valor de un campo cambia.
- **input:** Mientras se escribe en un campo de texto.
- **focus:** Cuando un elemento recibe foco.
- **blur:** Cuando un elemento pierde el foco.



Ejemplo:

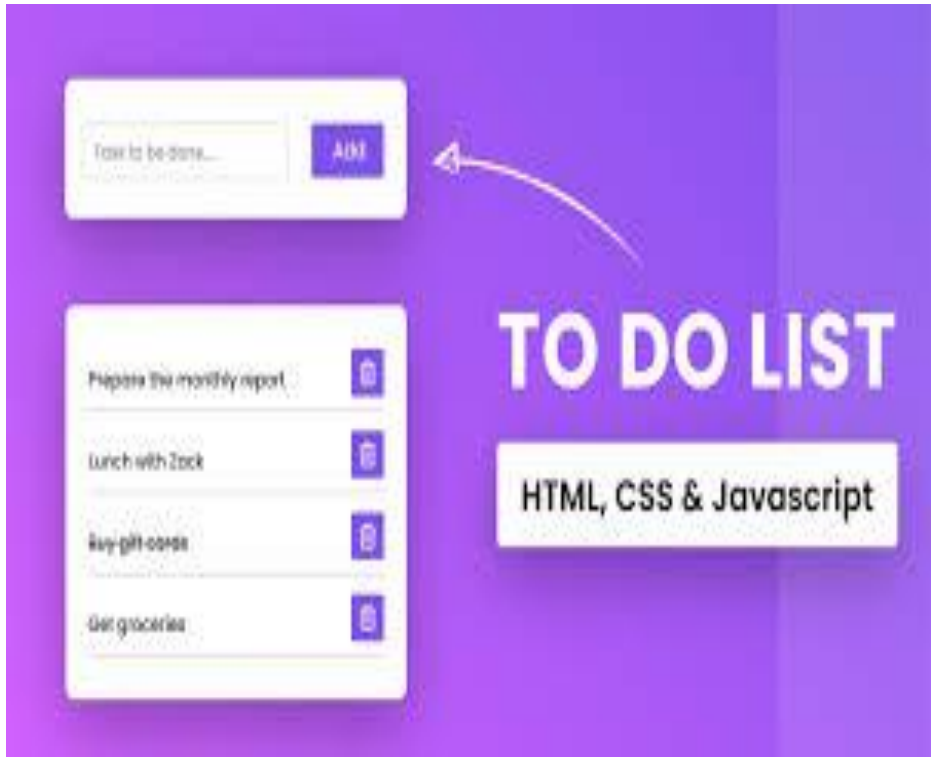
```
Const formulario = document.getElementById("miFormulario");
formulario.addEventListener("submit", (event) => {
event.preventDefault(); // Evita el envío del formulario
alert("Formulario enviado"); });
```

To-Do List con JavaScript

Una To-Do List (lista de tareas) es una aplicación interactiva ideal para practicar el manejo de eventos, el DOM y las funciones dinámicas de JavaScript.

Características

1. Agregar tareas.
2. Marcar tareas como completadas.
3. Eliminar tareas.



To-Do List con JavaScript

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>To-Do List</title>
  <style>
    li.completed { text-decoration: line-through; color: gray; }
    .deleteBtn { background: red; color: white; cursor: pointer; }
  </style>
</head>
<body>
  <h1>To-Do List</h1>
  <input id="taskInput" type="text" placeholder="Escribe una tarea...">
  <button id="addTaskBtn">Agregar</button>
  <ul id="taskList"></ul>

</body>
</html>
```

```
<script>
    const taskInput = document.getElementById("taskInput");
    const addTaskBtn = document.getElementById("addTaskBtn");
    const taskList = document.getElementById("taskList");

    addTaskBtn.addEventListener("click", () => {
      const taskText = taskInput.value.trim();
      if (taskText === "") return;

      const li = document.createElement("li");
      li.innerText = taskText;

      const deleteBtn = document.createElement("button");
      deleteBtn.innerText = "Eliminar";
      deleteBtn.classList.add("deleteBtn");
      deleteBtn.addEventListener("click", () => li.remove());

      li.addEventListener("click", () => li.classList.toggle("completed"));

      li.appendChild(deleteBtn);
      taskList.appendChild(li);
      taskInput.value = "";
    });

    taskInput.addEventListener("keydown", (event) => {
      if (event.key === "Enter") addTaskBtn.click();
    });
  </script>
```


JUNTOS TRANSFORMAMOS
EL MUNDO DESDE
LA AMAZONÍA