Universidad Autónoma de Santo Domingo

Facultad de Ciencia

Escuela de Informática

Asignatura:

Laboratorio Lenguaje de Programación III

Tema:

Realizar un resumen de AJAX y su uso -Coloque Ejemplo-

Estudiante:

Genesis Batista Mejía - 100572904

Maestro:

Radhames Silverio González

¿Qué es AJAX?

AJAX (Asynchronous JavaScript and XML) es una técnica de desarrollo web que permite a las aplicaciones web comunicarse con el servidor de manera **asíncrona**, lo que significa que se pueden enviar y recibir datos en segundo plano sin tener que recargar completamente la página web. Aunque su nombre menciona XML, en la práctica se usa comúnmente **JSON** como formato de intercambio de datos por ser más ligero y compatible con JavaScript.

¿Por qué es importante?

Antes de AJAX, cada vez que un usuario hacía clic o enviaba un formulario, la página completa se recargaba desde el servidor, lo cual generaba demoras e incomodidad. Con AJAX, sólo se actualiza la parte necesaria de la página, lo que proporciona una **experiencia más fluida e interactiva**, similar a las aplicaciones de escritorio.

Historia y evolución

AJAX no es un lenguaje nuevo, sino una combinación de tecnologías ya existentes:

- HTML y CSS para estructurar y diseñar la página.
- JavaScript para manejar eventos y lógica.
- DOM (Document Object Model) para actualizar contenido dinámicamente.
- XMLHttpRequest (XHR) o fetch() para las solicitudes HTTP.
- JSON (más usado que XML) como formato de intercambio de datos.
- Un servidor (PHP, Node.js, Python, etc.) para responder a las solicitudes.

Funcionamiento interno y tecnologías relacionadas

Ciclo de vida de una solicitud AJAX

1. El usuario hace una acción (clic, escritura, etc.).
2. JavaScript captura esa acción y crea una solicitud HTTP usando fetch() o XMLHttpRequest.
3. El servidor procesa la solicitud (consultas, cálculos, etc.).
4. El servidor responde con datos (generalmente en JSON).
5. JavaScript recibe los datos y los inserta en el DOM.
6. La página se actualiza dinámicamente, sin recargarla.

Sintaxis básica con fetch()

```
fetch('datos.json')
  .then(response => response.json())
  .then(data => {
    document.getElementById("resultado").innerHTML = data.nombre;
  })
  .catch(error => console.error("Error:", error));
```

Comparación: fetch() vs XMLHttpRequest

Característica	fetch()	XMLHttpRequest
Sintaxis	Moderna y concisa	Verbosa y más compleja
Promesas	Sí (por defecto)	No (requiere callbacks)
Soporte de streaming	Sí	No
Compatibilidad	Navegadores modernos	Compatible desde IE5+

Ventajas de usar AJAX

- Interfaz más dinámica y rápida.
- Menor carga para el servidor.
- Mejor experiencia para el usuario.
- Permite interacciones tipo aplicación (como Gmail, Facebook, etc.).

Desventajas y precauciones

- Requiere manejo cuidadoso de errores y estados.
- Complica el SEO si se abusa (páginas sin recarga completa).
- La lógica en el cliente puede volverse difícil de mantener si no se estructura bien.

Casos prácticos y aplicaciones de AJAX

HTML:

```
<button onclick="cargarUsuario()">Ver Usuario</button>
<div id="resultado"></div>
```

JavaScript:

```
function cargarUsuario() {
  fetch('usuario.json')
    .then(res => res.json())
    .then(data => {
      document.getElementById("resultado").innerHTML =
        `<p>Nombre: ${data.nombre}</p><p>Email: ${data.email}</p>`;
    });
}
```

JSON (usuario.json):

```
{
```

```
  "nombre": "Laura Gómez",
  "email": "laura@example.com"
}
```

AJAX revolucionó la forma en que se desarrollan aplicaciones web, al permitir crear interfaces ricas, rápidas y dinámicas. Aunque hoy en día muchos frameworks lo abstraen, entender cómo funciona AJAX es esencial para cualquier desarrollador web. Saber hacer solicitudes, manejar datos JSON, actualizar el DOM y tratar errores son habilidades clave en el desarrollo moderno.