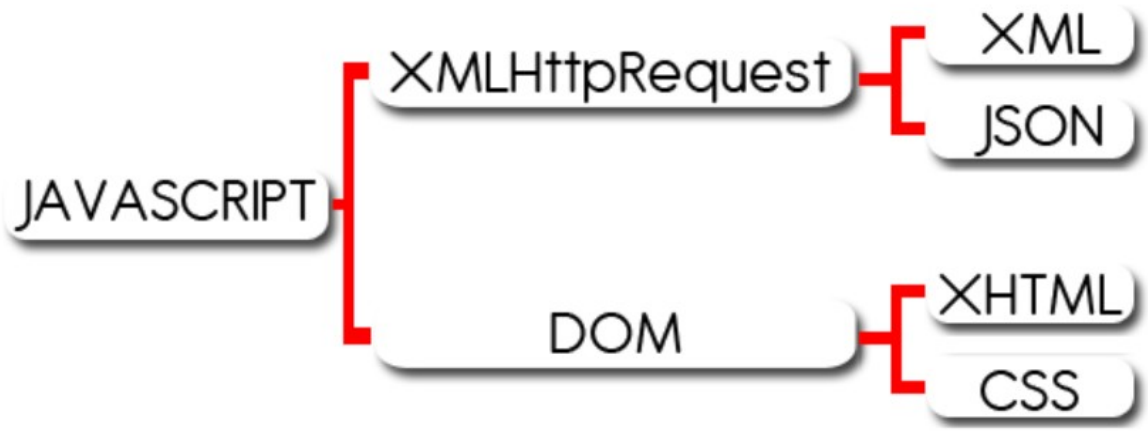
AJAX: Asynchronous Javascript And XML" (Javascript asíncrono y XML).

AJAX en si no es una tecnología, sino un conjunto de tecnologías. Permite comunicarse con el servidor, intercambiar datos y actualizar la página **sin tener que recargar el navegador.**

Las tecnologías presentes en AJAX son:

- XHTML y CSS para la presentación de la página.
- DOM para la manipulación dinámica de elementos de la página.
- Formatos de intercambio de información como JSON o XML.
- El objeto XMLHttpRequest, para el intercambio asíncrono de información (es decir, sin recargar la página).
- Javascript, para aplicar las anteriores tecnologías.



La forma de trabajar es la siguiente: JavaScript se encarga de unir todas las tecnologías. Para manipular la parte de representación de la página utiliza DOM (manipula el XHTML y CSS)..

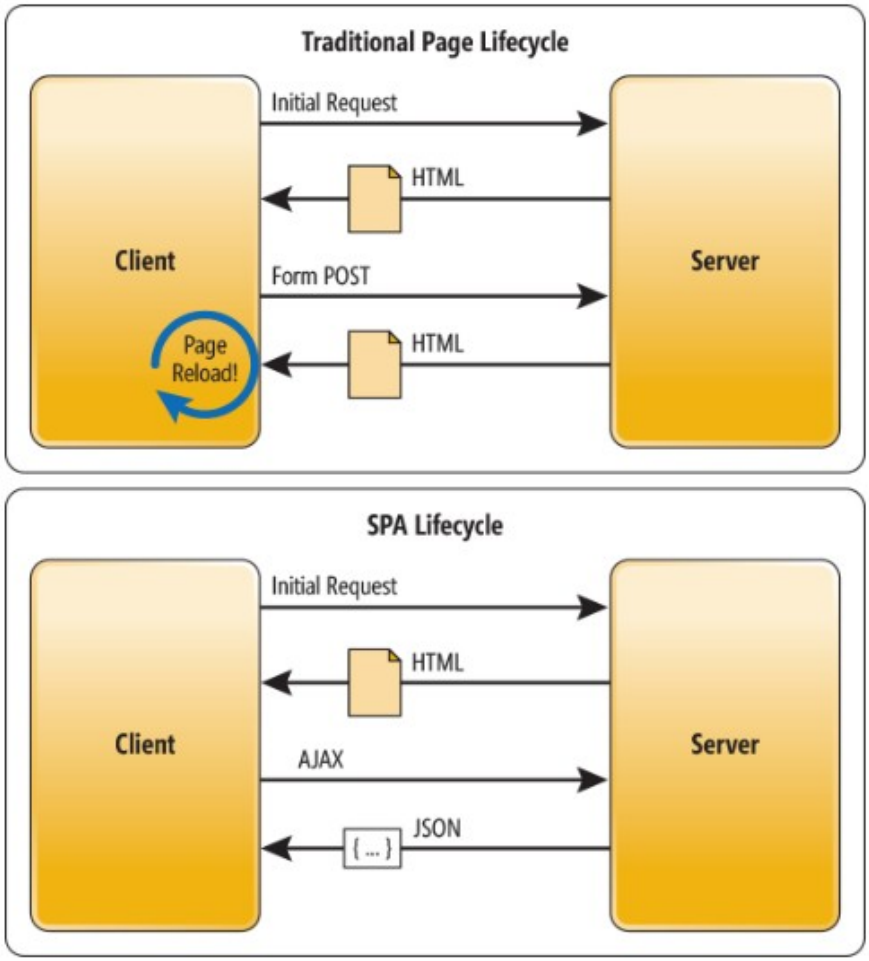
Para realizar peticiones asíncronas usa el objeto XMLHttpRequest. Este objeto intercambia información que son simplemente cadenas de texto. Lo más habitual es utilizar JSON o XML.

Como resultado obtenemos una navegación ágil, rápida y dinámica; y también la posibilidad de realizar cambios sobre una web sin necesidad de actualizarla.

Cuando interactuamos con servidores, podemos hacer uso de diferentes métodos HTTP para solicitar datos. Podemos crear, leer, actualizar y eliminar (CRUD) datos en los servidores utilizando verbos HTTP específicos como POST, GET, PUT/PATCH y DELETE.

- **GET**: Leer.
- **POST**: Crear.
- **PUT**: Actualizar.
- **DELETE**: Borrar.

Web tradicional vs Ajax:



En una aplicación Web clásica:

1. El cliente hace una petición al servidor.
2. El servidor recibe la petición.
3. El servidor procesa la petición y genera una nueva página con la petición procesada. (Ejemplo, se añade un post a un foro).
4. El cliente recibe la nueva página completa y la muestra.

En una aplicación Web AJAX:

1. El cliente hace una petición asíncrona al servidor.
2. El servidor recibe la petición.
3. El servidor procesa la petición y responde asíncronamente al cliente.
4. El cliente recibe la respuesta y con ella modifica dinámicamente los elementos afectados de la página sin recargar completamente la página.

Las aplicaciones Web AJAX son mejores ya que reducen la cantidad de información a intercambiar (no se envía la página entera, sino que se modifica solo lo que interesa) y a su vez al usuario final le da una imagen de mayor dinamismo, viendo una página web como una aplicación de escritorio.

El cliente es el programa que envía una solicitud, mientras que el servidor es el que recibe la solicitud. El servidor devuelve una respuesta en función de la validez de la solicitud. Si la solicitud tiene éxito, el servidor devuelve los datos en formato XML o JSON (JSON en la mayoría de los casos), y si la solicitud falla, el servidor devuelve un mensaje de error.

Las respuestas que devuelve el servidor suelen estar asociadas a códigos de estado (**status**). Estos códigos nos ayudan a entender lo que el servidor intenta decir cuando recibe una petición. Aquí tienes algunos de ellos y su significado:

- 100-199 denota una respuesta informativa.
- 200-299 **denota una solicitud exitosa.**
- 300-399 denota una redirección.
- 400-499 indica un error del cliente.
- 500-599 denota un error del servidor.

Códigos de respuesta (readyState)

- 0 al inicializarse el objeto.
- 1 al abrirse una conexión (al usar el método open).
- 2 al hacer una petición (uso de send).
- 3 mientras se está recibiendo información de la petición.
- 4 cuando la petición se ha completado.

Estado	Valor
READY_STATE_UNINITIALIZED	0
READY_STATE_LOADING	1
READY_STATE_LOADED	2
READY_STATE_INTERACTIVE	3
READY_STATE_COMPLETE	4

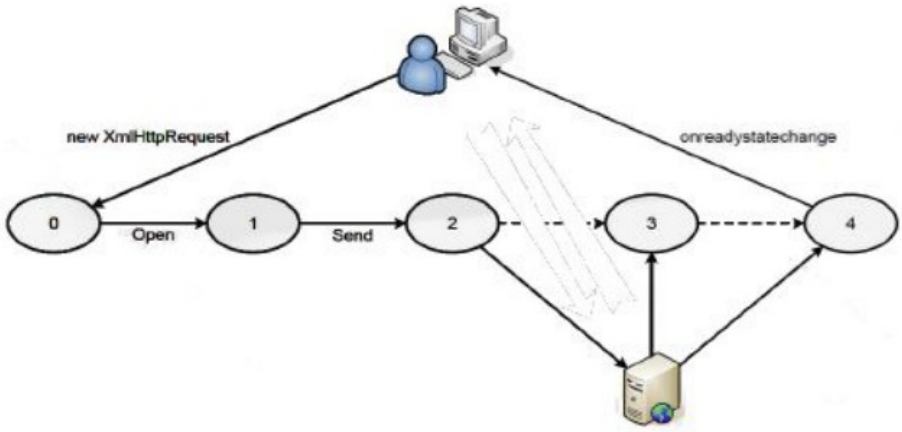
Métodos JS Nativo:

- XMLHttpRequest
- **API Fetch**

Librerías externas

- Jquery Ajax
- Axios
- …

Objeto XMLHttpRequest



¿Cómo lo aplicamos en código? El evento onreadystatechange cada vez que se produzca comprobará en que estado nos encontramos y hará lo planeado para ese estado. Generalmente el estado más utilizado es el 4, donde se ha completado la operación.

Cómo enviar una solicitud GET en JavaScript utilizando XMLHttpRequest

```
const xhr = new XMLHttpRequest();
xhr.open("GET", "https://jsonplaceholder.typicode.com/users");
xhr.send();

xhr.addEventListener("readystatechange", (e) => {
  if (xhr.readyState !== 4) return;

  if (xhr.status >= 200 && xhr.status < 300) {
    console.log("éxito");

    let json = JSON.parse(xhr.responseText);
    console.log(json);

    json.forEach((el) => {
      const $li = document.createElement("li");
      $li.innerHTML = `${el.name} -- ${el.email} -- ${el.phone}`;
      $fragment.appendChild($li);
    });

    $xhr.appendChild($fragment);
  } else {
    console.log("error");
    let message = xhr.statusText || "Ocurrió un error";
    $xhr.innerHTML = `Error ${xhr.status}: ${message}`;
  }

  console.log("Este mensaje cargará de cualquier forma");
});
```

Cómo enviar una petición POST en JavaScript utilizando XMLHttpRequest

```
const xhr = new XMLHttpRequest();
xhr.open("POST", "https://jsonplaceholder.typicode.com/posts");

xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

const body = JSON.stringify({
  title: "Hello World",
  body: "My POST request",
  userId: 900,
});

xhr.addEventListener("readystatechange", (e) => {

   if (xhr.readyState !== 4) return;

   if (xhr.status >= 200 && xhr.status < 300) {
      console.log(JSON.parse(xhr.responseText));
   } else {
     console.log(`Error: ${xhr.status}`);
  }
};
xhr.send(body);
```

Cómo enviar una petición PUT en JavaScript utilizando XMLHttpRequest

```
const xhr = new XMLHttpRequest();
xhr.open("PUT", "https://jsonplaceholder.typicode.com/posts");

xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

const body = JSON.stringify({
  title: "Hello World",
  body: "My POST request",
  userId: 7,
});

xhr.addEventListener("readystatechange", (e) => {
   if (xhr.readyState !== 4) return;
   if (xhr.status >= 200 && xhr.status < 300) {
      console.log(JSON.parse(xhr.responseText));
   } else {
     console.log(`Error: ${xhr.status}`);
  }
};

xhr.send(body);
```

La información que se enviará al servidor se almacena en una variable llamada body. Contiene tres propiedades: title, body y userId.

Ten en cuenta que la variable body que contiene el objeto debe ser convertida en un objeto JSON antes de ser enviada al servidor. La conversión se realiza mediante el método **JSON.stringify().**

Para asegurarte de que el objeto JSON se envía al servidor, se pasa como parámetro al método send().

Cómo enviar una solicitud DELETE en JavaScript utilizando XMLHttpRequest

```
const xhr = new XMLHttpRequest();
xhr.open("DELETE", "https://jsonplaceholder.typicode.com/posts/3");
xhr.addEventListener("readystatechange", (e) => {
  var data = JSON.parse(xhr.responseText);
  if (xhr.status >= 200 && xhr.status < 300) {
    console.log(data);
  } else {
    console.log(`Error: ${xhr.status}`);
  }
};
xhr.send();
```

Ejercicios:

1.- Obtener el usuario con id=5 de jsonplaceholder.typicode.com/users.
Mostrar los datos del usuario en distintos <p> de un div de una página html: Nombre, usuario, correo y la dirección.

2.- Crear una página html con un <H1> POSTS </H1> y un botón. Cuando pulsemos un botón vamos a llamar a la api: jsonplaceholder.typicode.com/posts , y vamos a pintar el resultado en una **tabla**. Sacar en cada fila el title y el body, cada uno en una columna diferente.

3.- Crear una página html con un <h1> ¿Sí o No?. Tendrá un botón y un p para mostrar ahí la respuesta.
Cuando pulsemos el botón, se conectará con la api yesno.wtf/api para obtener la respuesta Si o No.
Crear otra página html2 igual a la anterior, tal que al pulsar el botón, cargaremos en el resultado la imagen que devuelva.

4.- Dada la api: jservice.io/api/clues , mostrar una lista con todas las pistas que aparecen: Category, question y answer. La etiqueta ul está creada en el html, las li no.

5.- Crear un fichero local json formado por un campo “students” que será un array de objetos literales. Dentro de cada uno habrá información de cada alumno: id: numero, nombre:cadena y notas, que será un array de 4 notas. Crear datos para 5 alumnos en el fichero.

Crear una tabla con la información de los alumnos con 4 columnas: ID , nombre, notas, que apareceran separadas por comas y la última columna será la media de notas.