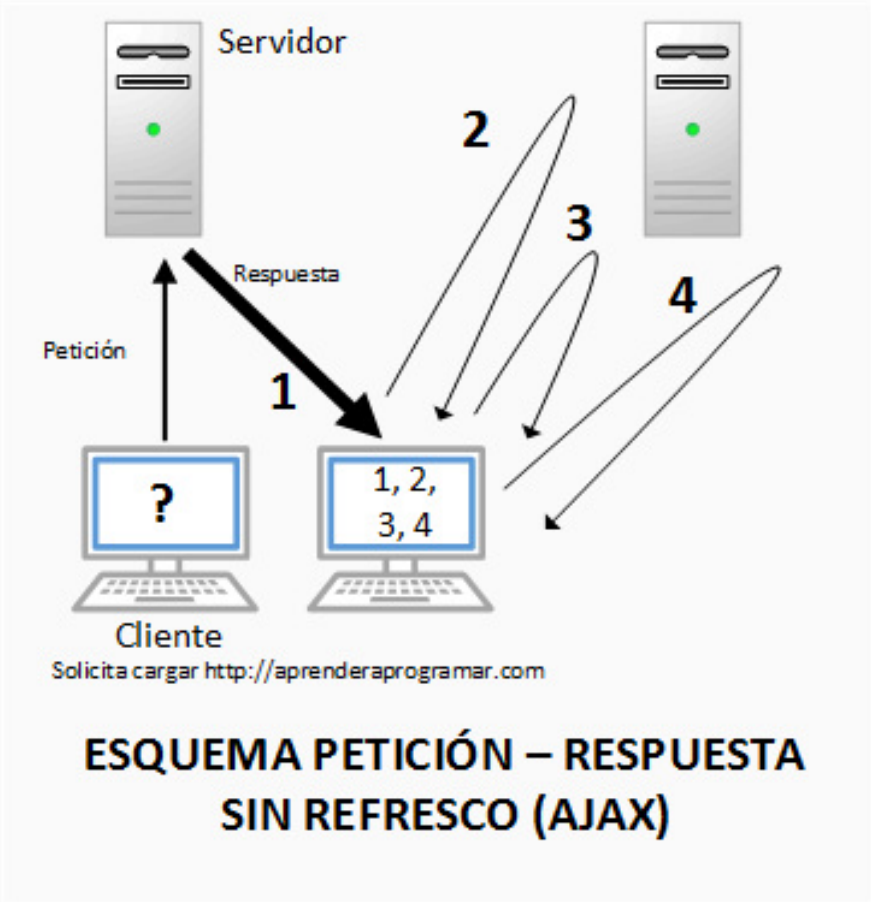
AJAX: Comunicación asíncrona con el servidor web

AJAX – Antes de AJAX

- La navegación y uso de aplicaciones web suponía que en cada petición al servidor se producía la carga de una nueva página en el navegador.
- Esto supone varios inconvenientes:
  - Cargar una página completa en cada acción del usuario.
  - Tiempo de respuesta más lento.
  - Transferencia de mayor cantidad de datos.
  - Bloqueo de la aplicación web hasta que se cargue la siguiente página.

AJAX - Definición

- Utilización conjunta de varias tecnologías (HTML, CSS, JS, XML, …) con el fin de construir aplicaciones web que se comuniquen de forma asíncrona con el servidor web.
- Con AJAX tenemos la posibilidad de hacer peticiones al servidor sin tener que volver a cargar la página completa.
  - Menor transferencia de datos
  - Sin bloquear la aplicación
  - Sin recarga de la página



AJAX – Peticiones al servidor

- Para peticiones al servidor se puede utilizar el objeto XMLHttpRequest o bien el más moderno método fetch().
- Las peticiones pueden llevar aparejadas el envío de datos.
- Por ejemplo, si se trata de una página con un formulario de registro de usuarios, los datos a enviar al servidor son los del usuario que se esté registrando (nombre de usuario, contraseña, fecha de nacimiento, correo electrónico, etc)

AJAX – fetch()

- Para utilizar el método fetch() debemos conocer:
  - La URL del servidor web que queremos invocar.
    - En dicha URL habrá un script/programa del servidor que procesará los datos que enviemos y nos devolverá una respuesta.
  - Los datos que necesitamos enviar para hacer la petición.
    - Dependiendo de la operación a realizar los datos a enviar serán distintos.
    - Es posible que haya peticiones al servidor que no envíen datos.
  - Los formatos de envío y recepción de datos.
    - Los datos se envían y reciben como cadenas de texto, lo que facilita el uso de JSON para enviar y recibir estructuras de datos complejas.
  - Método de la petición: GET o POST (hay más…)
    - GET – Peticiones orientadas a recuperar datos
    - POST – Peticiones orientadas a actualizar datos en el servidor

AJAX – Petición fetch() método GET I

```
/**
 * Realiza peticiones AJAX de tipo GET
 * @param {string} url
 * @param {FormData} parametros - Objeto FormData con los parámetros de la llamada
 * @returns response
 */
const peticionGET = async (url, parametros) => {
    // Creamos el objeto URL que contiene la dirección url de la petición
    // y los datos que enviamos con la petición
    let objetoURL = new URL(url);

    // Agregamos los datos de los parámetros que vienen en un objeto FormData
    for (let [clave, valor] of parametros) {
        objetoURL.searchParams.append(clave, valor);
    }
// continúa en la siguiente página
```

AJAX – Petición fetch() método GET II

```
// Finalmente hacemos la petición AJAX con el método fetch
    let respuestaServidor = await fetch(objetoURL, { method: "GET" });

    let response; // Variable para datos devueltos por el servidor o datos de error

    if (! respuestaServidor.ok) {  // Si no es una respuesta OK
        console.error("Error al acceder al acceder al servidor (STATUS != 200..299).");
        response = { error: 1,
                     mensaje: "Error al acceder al acceder al servidor (STATUS != 200..299)." };
    } else {
        response = await respuestaServidor.json();
    }

    return response;
}
```

AJAX – Petición fetch() método POST I

```
/**
 * Realiza peticiones AJAX de tipo POST
 * @param {string} url
 * @param {FormData} parametros - Objeto FormData con los parámetros de la llamada
 * @returns
 */
const peticionPOST = async (url, parametros) => {
    // Creamos el objeto URL que contiene la dirección url de la petición
    let objetoURL = new URL(url);

    let respuestaServidor = await fetch(objetoURL, {
        body: parametros,
        method: "POST"
    });

// continúa en la siguiente página
```

AJAX – Petición fetch() método POST II

```
    let response; // Variable para datos devueltos por el servidor o datos de error

    if (! respuestaServidor.ok) {  // Si no es una respuesta OK
        console.error("Error al acceder al acceder al servidor (STATUS != 200..299).");
        response = { error: 1,
                     mensaje: "Error al acceder al acceder al servidor (STATUS != 200..299)." };
    } else {
        response = await respuestaServidor.json();
    }

    return response;
}
```