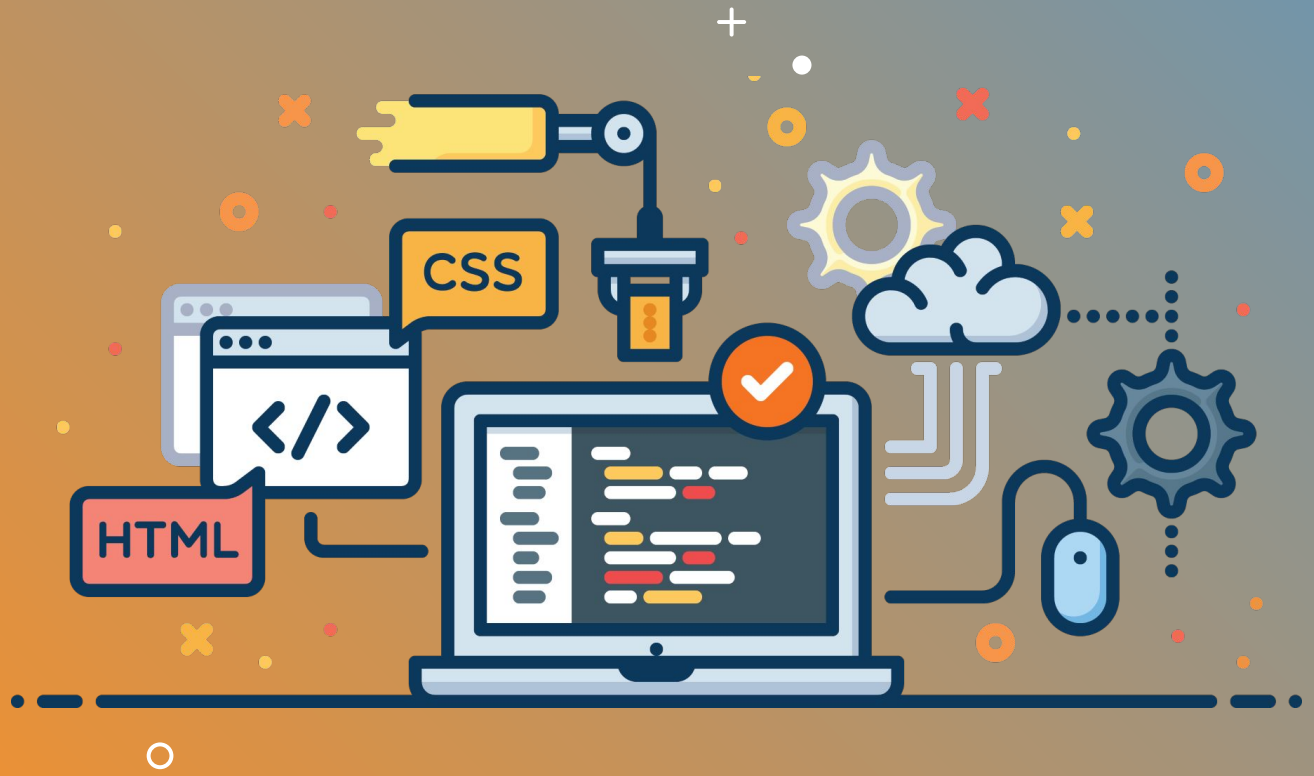
DESARROLLO DE APLICACIONES WEB FRONT END

Módulo 4

OTEC PLATAFORMA 5 CHILE

MÓDULO 4



CLASE 3

Introducción

XMLHttpRequest

Solicitud Asincrónica

Estados

Solicitud Sincrónica

Introducción

Ahora que ya hemos recorrido qué son los fetch y cómo están compuestas las URL, estamos en condiciones de trabajar con XMLHttpRequest.

Es necesario tener los conocimientos y las herramientas para el manejo de las distintas páginas que utilizaremos en nuestros proyectos futuros.

Arrancamos con nuestra clase nro 3.

XMLHttpRequest

XMLHttpRequest es un objeto nativo del navegador que permite hacer solicitudes HTTP desde JavaScript.

A pesar de tener la palabra “XML” en su nombre, se puede operar sobre cualquier dato, no solo en formato XML. Podemos cargar/descargar archivos, dar seguimiento y mucho más.

En el desarrollo web moderno XMLHttpRequest se usa por tres razones:

- Razones históricas: necesitamos soportar scripts existentes con XMLHttpRequest.
- Necesitamos soportar navegadores viejos, y no queremos polyfills (p.ej. para mantener los scripts pequeños).
- Necesitamos hacer algo que fetch no puede todavía, ej. rastrear el progreso de subida.

Solicitud Asincrónica

XMLHttpRequest tiene dos modos de operación: sincrónica y asíncrona.

Veamos primero la asíncrona, ya que es utilizada en la mayoría de los casos.

Para hacer la petición, necesitamos seguir 3 pasos:

1. **Crear el objeto XMLHttpRequest:**

   ```
   let xhr = new XMLHttpRequest();
   ```

   El constructor no tiene argumentos.

2. **Inicializarlo, usualmente justo después de new XMLHttpRequest:**

   ```
   xhr.open(method, URL, [async, user, password])
   ```

Solicitud Asincrónica

Este método especifica los parámetros principales para la petición:

- **method: método HTTP. Usualmente "GET" o "POST".**
- **URL: la URL a solicitar, una cadena, puede ser un objeto URL.**
- **async: si se asigna explícitamente a false, entonces la petición será asincrónica.**
- **user, password: usuario y contraseña para autenticación HTTP básica (si se requiere).**

Importante: hay que tener en cuenta que la llamada a open, contrario a su nombre, no abre la conexión. Solo configura la solicitud, pero la actividad de red solo empieza con la llamada del método send.

3. Enviar.

```
xhr.send([body])
```

Este método abre la conexión y envía la solicitud al servidor. El parámetro adicional body contiene el cuerpo de la solicitud.

Solicitud Asincrónica

Algunos métodos como GET no tienen un cuerpo. Y otros como POST usan el parámetro body para enviar datos al servidor.

4. Escuchar los eventos de respuesta xhr.

Estos son los tres eventos más comúnmente utilizados:

- **load: cuando la solicitud está; completa (incluso si el estado HTTP es 400 o 500), y la respuesta se descargó por completo.**
- **error: cuando la solicitud no pudo ser realizada satisfactoriamente, ej. red caída o una URL inválida.**
- **progress: se dispara periódicamente mientras la respuesta está siendo descargada, reporta cuánto se ha descargado.**

Solicitud Asincrónica

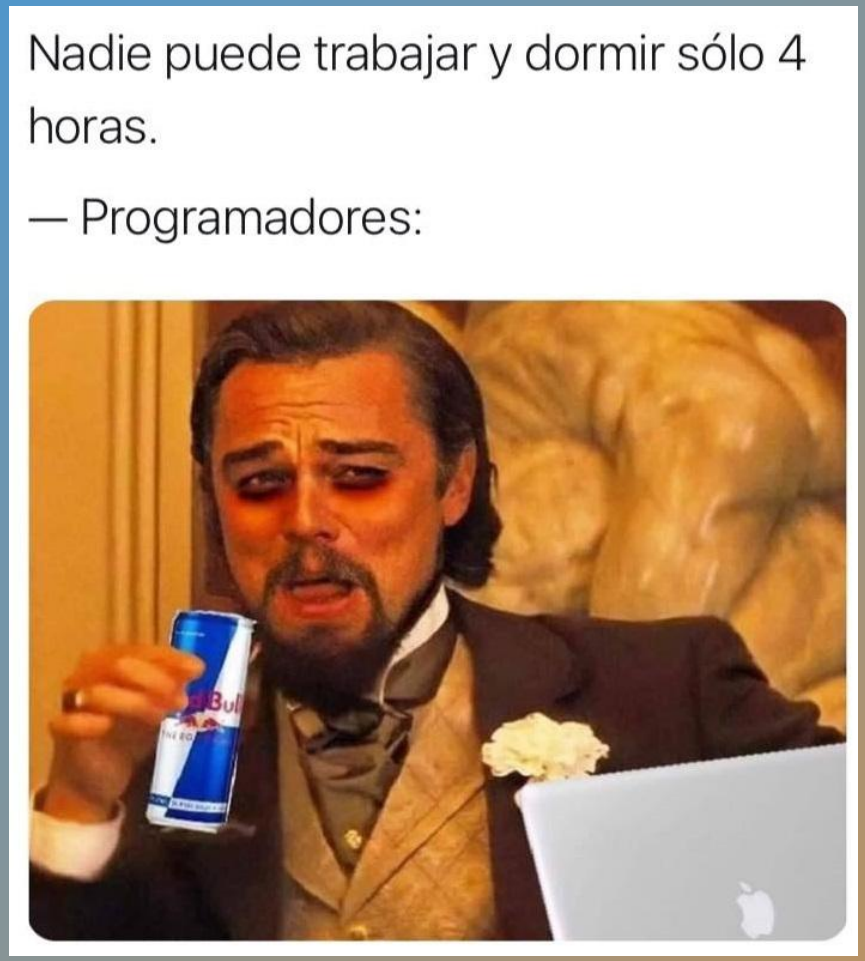
Una vez el servidor haya respondido, podemos recibir el resultado en las siguientes propiedades de xhr:

- **status: Código del estado HTTP (un número): 200, 404, 403 y así por el estilo, puede ser 0 en caso de una falla no HTTP.**
- **statusText: Mensaje del estado HTTP (una cadena): usualmente OK para 200, Not Found para 404, Forbidden para 403 y así por el estilo.**
- **response: El cuerpo de la respuesta del servidor.**

También podemos especificar un tiempo límite usando la propiedad correspondiente:

```
xhr.timeout = 10000; // límite de tiempo en milisegundos, 10 segundos
```

Si la solicitud no es realizada con éxito dentro del tiempo dado, se cancela y el evento timeout se activa.



Siempre es tiempo, para un buen meme...

Estados

XMLHttpRequest cambia entre estados a medida que avanza. El estado actual es accesible como xhr.readyState.

Todos los estados, como en la especificación:

```
UNSENT = 0; // estado inicial
OPENED = 1; // llamada abierta
HEADERS_RECEIVED = 2; // cabeceras de respuesta recibidas
LOADING = 3; // la respuesta está cargando (un paquete de datos es recibido)
DONE = 4; // solicitud completa
```

Un objeto XMLHttpRequest escala en orden 0 → 1 → 2 → 3 → ... → 3 → 4. El estado 3 se repite cada vez que un paquete de datos se recibe a través de la red.

Puedes encontrar oyentes del evento readystatechange en código realmente viejo, está ahí por razones históricas, había un tiempo cuando no existían load y otros eventos. Hoy en día los manipuladores load/error/progress lo hacen obsoleto.

```
xhr.onreadystatechange = function ()
{
 if (xhr.readyState == 3) {
   // cargando
 }
 if (xhr.readyState == 4) {
   // solicitud finalizada
 }
};
```

Solicitud Sincrónica

Si en el método open el tercer parámetro async se asigna como false, la solicitud se hace sincrónicamente.
En otras palabras, la ejecución de JavaScript se pausa en el send() y se reanuda cuando la respuesta es recibida. Algo como los comandos alert o prompt.

Aquí está el ejemplo, el tercer parámetro de open es false:

```
let xhr = new XMLHttpRequest();

xhr.open("GET",
"/article/xmlhttprequest/hello.txt", false);

try {
 xhr.send();
 if (xhr.status != 200) {
   alert(`Error ${xhr.status}:
${xhr.statusText}`);
 } else {
   alert(xhr.response);
 }
} catch (err) {
 // en lugar de onerror
 alert("Solicitud fallida");
}
```

Solicitud Sincrónica

Puede verse bien, pero las llamadas sincrónicas son rara vez utilizadas porque bloquean todo el JavaScript de la página hasta que la carga está completa. En algunos navegadores se hace imposible hacer scroll. Si una llamada síncrona toma mucho tiempo, el navegador puede sugerir cerrar el sitio web "colgado".

Algunas capacidades avanzadas de XMLHttpRequest, como solicitar desde otro dominio o especificar un tiempo límite, no están disponibles para solicitudes síncronas. Tampoco, como puedes ver, la indicación de progreso.

La razón de esto es que las solicitudes sincrónicas son utilizadas muy escasamente, casi nunca. Por lo tanto, no las usaremos.

MÓDULO 4

GRACIAS

Aquí finaliza la clase n°1 del módulo 4

OTEC PLATAFORMA 5 CHILE