perplexity

Quiero que me expliques el resto del código del mismo modo

Por supuesto, aquí va la explicación línea por línea del código completo explicado previamente, con un enfoque sencillo para una estudiante de bachillerato:

```
// Crear un nuevo objeto XMLHttpRequest
const xhr = new XMLHttpRequest();
```

- Aquí estamos creando un objeto llamado `xhr` que nos permite hacer una petición al servidor para pedir datos o enviar información. Es como preparar una carta para enviarla.

```
// Configurar la petición GET a una URL (por ejemplo, una API pública)
xhr.open('GET', 'https://jsonplaceholder.typicode.com/posts/1', true);
```

- Esta línea prepara la petición en sí. Le decimos que será una petición `GET` (es decir, que solo queremos obtener datos). Indicamos la dirección (URL) donde pediremos esos datos. El `true` significa que la petición será asíncrona (no bloquea el resto del código).

```
// Definir qué hacer cuando la respuesta esté lista
xhr.onreadystatechange = function() {
  if (xhr.readyState === 4 && xhr.status === 200) {
    // Parsear la respuesta JSON
    const datos = JSON.parse(xhr.responseText);
    // Mostrar los datos en consola
    console.log('Datos recibidos:', datos);
  }
};
```

- Aquí decimos qué hacer en cada cambio del estado de la petición.
- `xhr.readyState === 4` significa que la petición ya terminó y la respuesta llegó.
- `xhr.status === 200` indica que la respuesta fue exitosa (todo salió bien).
- Si ambas condiciones son ciertas, entonces convertimos el texto que llegó (`xhr.responseText`) a un objeto JavaScript con `JSON.parse()`, para poder usar esos datos fácilmente.
- Luego mostramos esos datos en la consola para que puedas verlos.

```
// Enviar la petición al servidor
xhr.send();
```

- Esta línea le dice que mande la petición al servidor. Hasta ahora solo estábamos preparándola, pero ahora comienza la comunicación real.

```
console.log('La petición se ha enviado, continúo ejecutando código...');
```

- Finalmente, este mensaje se muestra inmediatamente después de enviar la petición. Sirve para demostrar que nuestro código no se detuvo para esperar la respuesta, sino que sigue ejecutándose mientras llega la información.

En pocas palabras, este código pide información a un servidor sin detener el programa, espera a que llegue la respuesta, la transforma para poder utilizarla, y continúa trabajando mientras tanto sin bloqueos.

¿Quieres que te explique paso a paso cómo ver esto funcionando en tu navegador?