Peticiones Síncronas y Asíncronas con AJAX, Fetch y Axios en JavaScript

Autor: Cristian Mateo Lisintuña Correa

Web Avanzada

¿Qué son las peticiones síncronas y asíncronas?

1

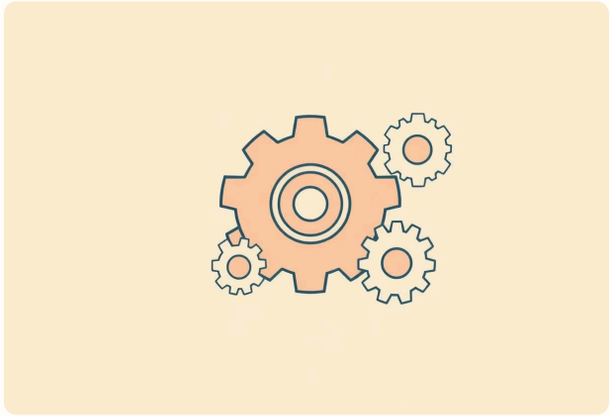
Síncronas

Bloquean la ejecución del código hasta recibir una respuesta completa, lo que puede congelar la interfaz de usuario y generar una experiencia de usuario deficiente. Imagine esperar por un documento mientras toda la aplicación se detiene.

2

Asíncronas

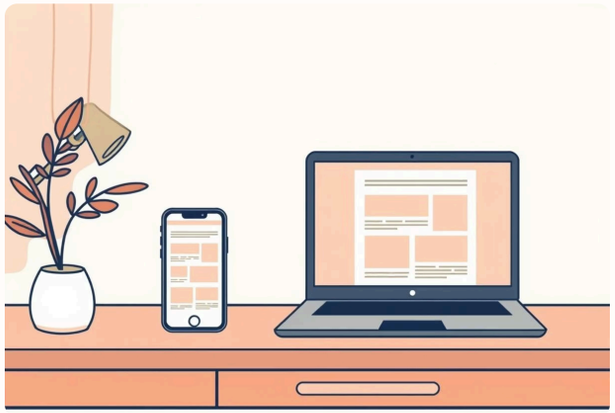
Permiten que la ejecución del programa continúe mientras se espera una respuesta. Esto mejora significativamente la fluidez y la interactividad de la aplicación, ya que el usuario puede seguir interactuando con la interfaz.

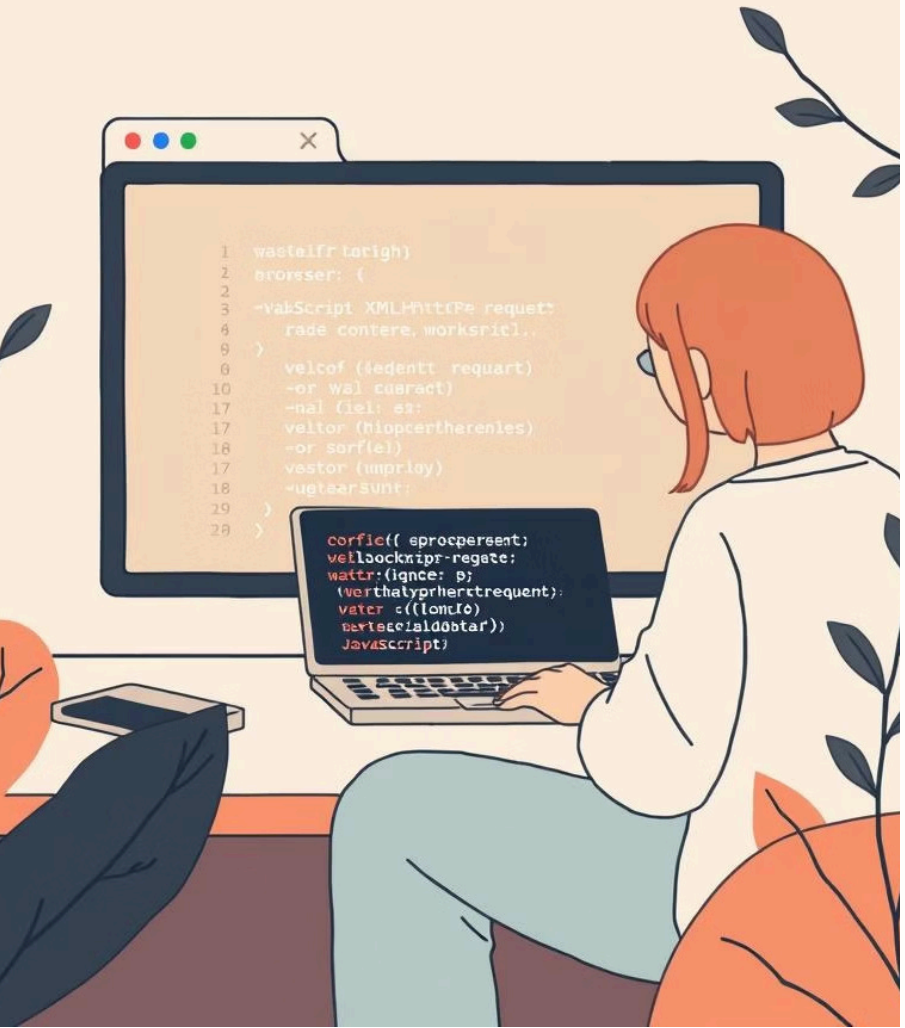


3

Importancia en la web

Son cruciales para evitar recargas completas de la página y para crear experiencias de usuario dinámicas y responsivas, fundamentales en las aplicaciones web actuales.





AJAX: El inicio de la comunicación asíncrona en la web

AJAX, acrónimo de **A**synchronous **J**avaScript **a**nd **X**ML, fue una revolución en su momento al permitir la actualización de contenido web sin necesidad de recargar la página completa. Utiliza el objeto **XMLHttpRequest (XHR)** para hacer estas peticiones.

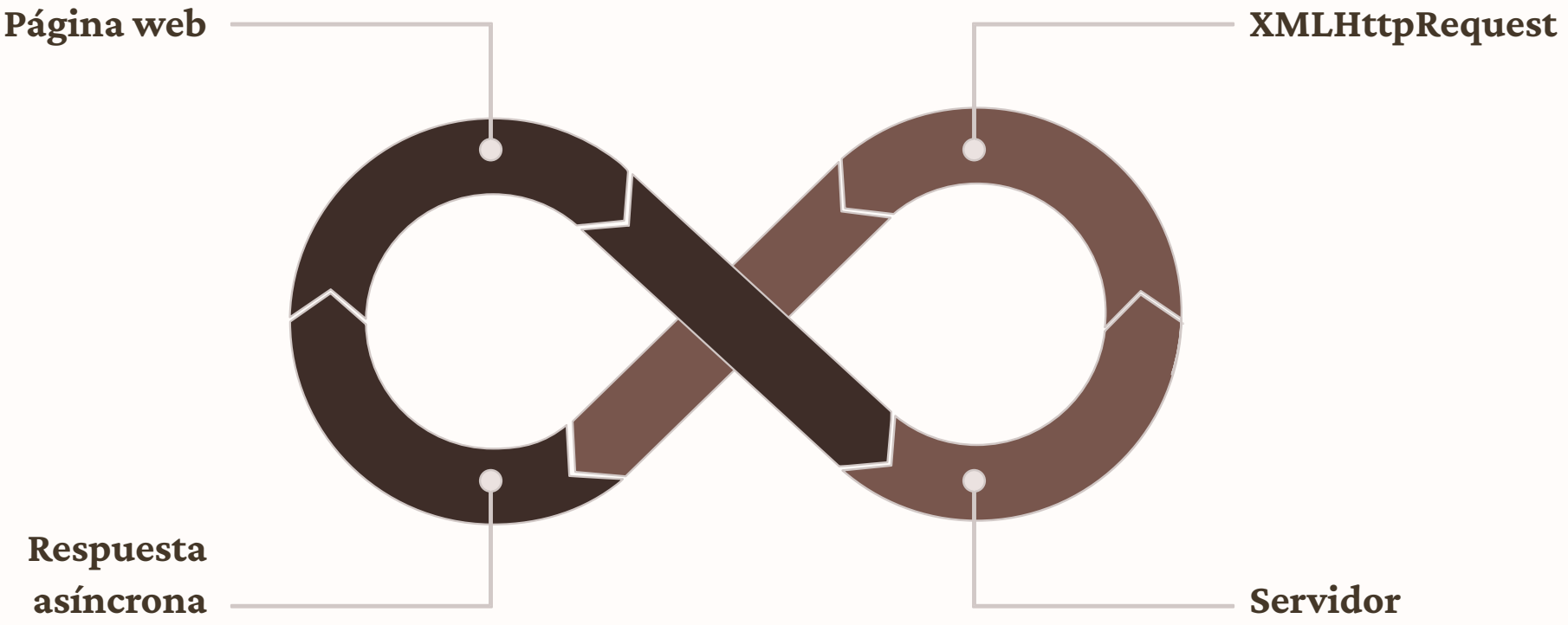
Ventajas de su origen

- Actualizaciones parciales de la página.
- Mejora inicial de la experiencia de usuario.
- Base para aplicaciones web dinámicas.

Limitaciones

- Manejo complejo de eventos.
- Código propenso a ser verboso.
- Dificultad para mantener proyectos grandes.

El Flujo de una Petición Asíncrona con XHR



Este diagrama ilustra cómo una página web inicia una petición a través de XMLHttpRequest, que se comunica con el servidor, y luego recibe una respuesta de forma asíncrona, permitiendo que el usuario siga interactuando con la interfaz.

Fetch API: La evolución moderna y nativa

Introducida con ES6, Fetch API ofrece una interfaz más potente y flexible para realizar peticiones HTTP. Su diseño, basado en **Promesas**, resulta en un código más limpio y fácil de comprender, superando las limitaciones de AJAX tradicional.

Sintaxis Simplificada

Utiliza `.then()` para encadenar operaciones asíncronas y `async/await` para una lectura secuencial.

Basado en Promesas

Facilita el manejo de errores y la composición de código asíncrono.

Nativa del navegador

No requiere librerías externas, está disponible directamente en los navegadores modernos.

Ejemplo básico de Fetch con GET:

```
fetch('https://api.example.com/data')
 .then(response => response.json())
 .then(data => console.log(data))
 .catch(error => console.error('Error:', error));
```