TEXT CLASS REVIEW

TEMAS A TRATAR EN EL CUE

- Introducción a Ajax.
- ¿Cómo implementar Ajax a un proyecto?
- API.
- Consumo de API.
- Método Fetch.
- XHR

AJAX

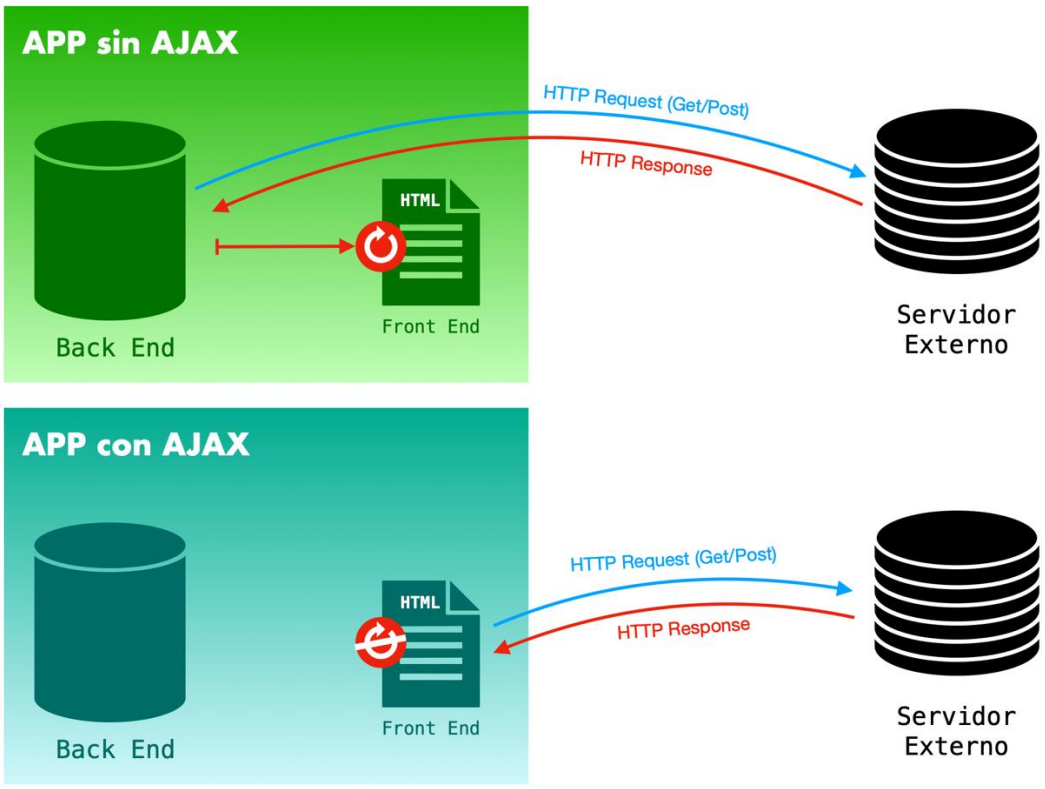
Hasta este momento, hemos aprendido acerca de las funcionalidades de la librería jQuery, y cómo usarlas para: simplificar nuestro desarrollo, manipular el contenido de nuestras páginas, y manejar los eventos instanciados por el usuario. Si bien todas estas funcionalidades son muy prácticas, ninguna de ellas nos permite extender las funcionalidades de JavaScript, más allá de la vista sobre la que estamos trabajando.

Ahora, veremos una potente capacidad de jQuery, la cual permite extender la funcionalidad de JavaScript en nuestros HTMLs, esta es AJAX, ¿de qué se trata?: es la capacidad de intercambiar información con un servidor, para actualizar parte del contenido de una página web sin tener que recargar toda la página, todo realizado de manera local. Las siglas AJAX vienen de "**A**synchronous **Ja**vaScript and **X**ML" o en castellano: JavaScript y XML Asíncrono.

Actualmente existen muchas aplicaciones populares que utilizan la tecnología AJAX. Por ejemplo, las aplicaciones de correo electrónico como Gmail, aplicaciones de mapas como Google Maps, e incluso algunas redes sociales como Facebook.

jQuery logra la funcionalidad de las llamadas AJAX, mediante una serie de métodos que permiten usar JavaScript para llevar a cabo peticiones HTTP de tipo `Get` y `Post`. Éstos nos permiten realizar consultas para recepcionar información en nuestras páginas web de tipo XML (como indica la sigla AJAX), HTML, o de tipo JSON. Es más, junto con estas funcionalidades que nos permiten acceder a información externa,

AJAX también otorga a nuestras páginas la capacidad de cargar información directamente a los elementos HTML de nuestras páginas.



Como se puede ver, AJAX es una importante funcionalidad que podemos implementar en nuestro desarrollo, gracias a la librería jQuery. De hecho, si no tuviésemos a disposición estos métodos, sería más complejo incorporar la funcionalidad AJAX a nuestros desarrollos. Esto se debe al hecho de que los navegadores utilizan distintas sintaxis para su implementación, lo que significa que tendríamos que escribir código AJAX regular, e ir probando uno por uno su uso efectivo con cada navegador. En cambio, con los métodos de jQuery, podemos usar el mismo código para implementarlo en todos los navegadores.

APIS Y FETCH

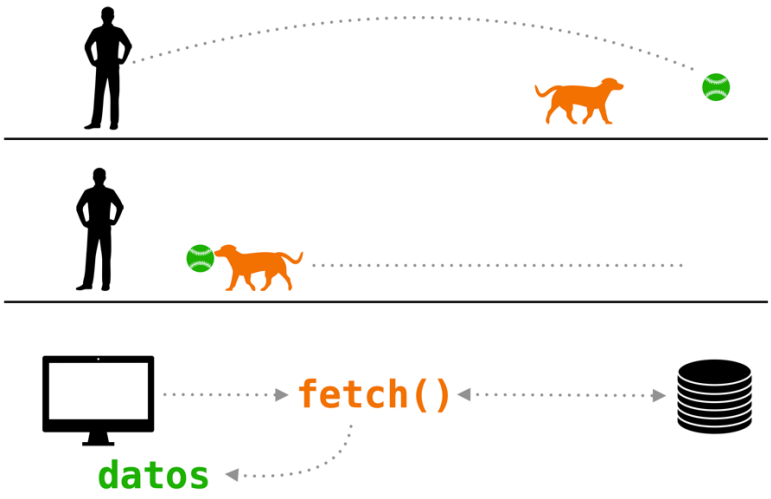
Según IBM, las APIs, o **I**nterfaz de **P**rogramación de **A**plicaciones, “permiten a las empresas abrir la funcionalidad de sus aplicaciones a desarrolladores externos, socios comerciales y departamentos internos dentro de una empresa. Esto permite que los servicios y productos se comuniquen entre sí, y

aprovechen los datos y la funcionalidad de los demás, a través de una interfaz documentada". De esta forma, las APIs permiten unificar el traspaso de información desde una fuente a otra, y hacer uso de ellas se ha vuelto tan popular que IBM también destacó que "El uso de las APIs se ha disparado durante la última década, hasta el punto de que muchas de las aplicaciones web más populares de la actualidad no serían posibles sin ellas".

Dada su importancia, vamos a analizar cómo se pueden utilizar estos recursos mediante JavaScript. Ahora bien, puede que te preguntes por qué veremos cómo consumir una API si ya lo hablamos cuando se trabajó con `jQuery.ajax()`. La razón es que ahora estudiaremos cómo consumir una API solo usando JavaScript puro. Hacerlo de esta forma requiere conocimiento de las funciones async, y de las Promesas. Dado que ahora manejamos estos conceptos, ya estamos listos para aprender cómo aumentar las capacidades de nuestras aplicaciones usando la API Fetch.

La API Fetch proporciona una forma sencilla y lógica de obtener recursos de forma asincrónica, a través del internet mediante el método `fetch()`.

ENTENDIENDO SEMANTICAMENTE EL FETCH



La palabra fetch se puede traducir a castellano como *"ir a buscar"*. Un ejemplo para ayudarnos a entender el `fetch()`, es cuando una persona juega a "ir por la pelota" con su mascota. En este caso, el dueño depende de su mascota para recuperar la pelota. Aunque éste sabe que su mascota traerá la pelota de

regreso, no tiene forma de saber cuánto tiempo tardará en recuperarla. De manera similar, los programas pueden usar el método `fetch()` para recuperar información de un servidor; y al igual que en el ejemplo, `fetch()` devuelve una promesa, lo que indica que, aunque no sabemos cuánto tiempo tomará en obtener información de un servidor, sabemos que nos devolverá los datos en algún momento.

Este tipo de funcionalidad se lograba anteriormente mediante el objeto `XMLHttpRequest`, pero a partir de ES6, los navegadores han podido realizar peticiones HTTP usando la API fetch. Aparte de ser una forma más nueva de consumir recursos a través de un navegador, ésta también es mucho más simple y limpia.

DIFERENCIA ENTRE API DEL BROWSER, API DE TERCEROS, LIBRERÍA JAVASCRIPT, FRAMEWORK JAVASCRIPT.

- **JAVASCRIPT**: lenguaje de programación interpretado, es decir, que no requiere de compilación, sino que es analizado por otro programa, específicamente los navegadores web. Éste es utilizado principalmente para ejecutar acciones del lado del cliente (aunque también se puede ocupar desde el servidor, como Node). Su principal objetivo es añadir interactividad a las páginas web, y que las acciones de los usuarios influyan en la información presentada en la página.

- **APIS DEL BROWSER**: construcciones integradas en el navegador, creadas con el lenguaje JavaScript, y que permiten implementar funcionalidad más fácilmente.
  Dentro de las API´s del browser más comunes se destacan:
  - API DOM: para manipular HTML y CSS.
  - Fetch API: para obtener datos desde un servidor externo.

- **APIS DE TERCEROS**: construcciones integradas en plataformas de terceros, que permiten usar algunas de las funcionalidades de esa plataforma en tus páginas web.
  Dentro de las API´s del browser más comunes se destacan:
  - API de Twitter.
  - API de googleMaps.
  - API de Facebook.
  - Youtube API.

- **LIBRERÍAS JAVASCRIPT**: por lo general, uno o más archivos JavaScript que contienen funciones personalizadas que puedes añadir a tu página web, para acelerar o habilitar la escritura de funcionalidades comunes. Por ejemplo: jQuery, Mootools y React.
- **FRAMEWORKS JAVASCRIPT**: son el siguiente paso a las librerías, como Angular y Ember, y suelen ser paquetes de HTML, CSS, JavaScript, y otras tecnologías que se instalan y luego se usan para escribir una aplicación web completa desde cero. La diferencia clave entre una librería y un framework, es la "Inversión del control". Cuando se llama a un método desde una librería, el desarrollador tiene el control. Con un framework el control se invierte: es él quien llama al código del desarrollador.

CUANDO USAR UNA U OTRA APIS DE TERCEROS

Cambios: debemos tener en consideración que, al estar utilizando servicios de terceros en nuestro desarrollo, se puede cambiar la utilización de la API, ya sea por mejora de la misma o por cuestiones de seguridad; y si no estamos al tanto de estos cambios, podemos llegar a perder el acceso. También existe la posibilidad de que, al momento de adquirir el acceso a la API, éste se procesara de manera gratuita, y que después este servicio sea de pago.

Control de disponibilidad: si bien es cierto que el uso de API´s nos ahorra tiempo en desarrollo de las aplicaciones o funcionalidades de los sitios web, también lo es que no deja de ser un servicio de terceros, del cual no podemos tener control, por lo que si éste deja de funcionar de manera repentina, no podríamos solucionarlo y tendríamos que esperar a que los desarrolladores resuelvan el problema.

Conocimiento: otro de los puntos para determinar qué API utilizar, es el nivel de conocimiento que se tenga en la implementación de éstas; algunas son más complejas de emplear, por lo que puede provocar retrasos y mal desempeño de la misma.

XHR EN JAVASCRIPT

XMLHttpRequest (XHR) es una API en JavaScript que permite a los navegadores interactuar con los servidores para recuperar datos de una URL sin tener que recargar la página completa. Aunque Fetch API es más moderna y recomendada, XHR sigue siendo importante y útil para comprender cómo funcionan las solicitudes HTTP en JavaScript.

Características de XHR:

- Asincronía: Permite enviar y recibir datos sin bloquear la ejecución del código.
- Compatibilidad: Soportado por la mayoría de los navegadores.
- Versatilidad: Puede manejar diferentes tipos de datos como texto, JSON, XML, etc.

Pasos para Usar XMLHttpRequest:

1. Crear un Objeto XMLHttpRequest:
   Para comenzar, se crea una nueva instancia de 'XMLHttpRequest'.

```
// Crear una nueva instancia de XMLHttpRequest
const xhr = new XMLHttpRequest();
```

2. Configurar la Solicitud:
   Se configura la solicitud especificando el método HTTP (como GET o POST) y la URL de la API que se desea consumir.

```
// Configurar la solicitud: método GET y URL de la API
xhr.open('GET', 'https://jsonplaceholder.typicode.com/posts', true);
```

3. Enviar la Solicitud:

   La solicitud se envía al servidor.

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

4. Manejar la Respuesta:

   Se define una función para manejar la respuesta del servidor. Esto se hace estableciendo un manejador de eventos para el evento onreadystatechange.

```
// Definir una función para manejar la respuesta del servidor
xhr.onreadystatechange = function() {
    // Verificar si la solicitud está completa y fue exitosa
    if (xhr.readyState === 4 && xhr.status === 200) {
        // Convertir la respuesta JSON en un objeto JavaScript
        const data = JSON.parse(xhr.responseText);

        // Procesar los datos recibidos (aquí, simplemente los mostramos en la consola)
        console.log(data);

    }
};
```