Introducción a AJAX
Las técnicas de Javascript asíncrono surgen ante la necesidad de solicitar información a los servidores web una vez que el documento HTML ya ha sido cargado en el navegador.
En lugar de realizar una conexión de un documento completo las páginas pueden requerir determinados datos concretos una vez cargadas. Estos datos pueden estar en formato texto, XML o JSON.

Resumen de lo que vamos a estudiar de AJAX

- Realizar peticiones asíncronas a un servidor para obtener información sobre archivos a descargar
- Peticiones asíncronas en paralelo
- Parametrizar solicitudes AJAX
- Subida de archivos mediante AJAX

Instalación de Lite Server
Para realizar los ejercicios necesitaremos un servidor al cual le solicitaremos determinados archivos como si fuera un servidor real.

Instalaremos un servidor web muy ligero implementado en Node que nos permitirá ver los cambios realizados en nuestro código al instante.

Si no disponemos de Node JS tendremos que instalarlo desde su web oficial o desde el repositorio de nuestra distribución de Linux. Es fundamental que instalemos también la herramienta de gestión de paquetes de Node de nombre npm.

En Linux:
```
apt install -y nodejs npm
```

Nos moveremos a la carpeta donde estarán los archivos que queremos que sirva nuestro servidor lite-server e inicializamos el proyecto de node:
```
npm init
```

Ya entonces podremos instalar el servidor con:
```
npm install lite-server --save-dev
```

Si observamos la página de lite-server comprobaremos que hay que copiar cierto trozo de código en el *package.json* que generó la creación del proyecto de node si aún no se encuentra relativo a la parte de scripts.

Para correr nuestro servidor basta ejecutar:
```
npm run dev
```

Se nos abrirá una ventana del navegador apuntando a localhost:3000 ofreciéndonos si la hubiese un *index.html*.

EJERCICIOS JQUERY Y AJAX

1. Utilizando el método *ajax* de jQuery recorre el documento HTML buscando todos los enlaces cuya clase sea “enlaces” y solicitando al servidor el *‘Content-Length’* de los mismos para añadirles su tamaño al lado.
   Pista: Utilizaremos el método *each* para recorrer todos los enlaces y *append* para asociarles el tamaño en formato legible.

2. **A)** Dado un archivo ejemplo.json que se encuentra alojado en la carpeta raíz del servidor realizar una solicitud get para mostrarlo por consola o en caso de que no exista un mensaje de error.

   B) Utilizando el servicio de https://jsonip.com y la función get de jQuery muestra en el párrafo denominado “objetivo” la dirección IP pública de nuestro router.

3. Dos los archivos *1.json* y *2.json* realiza un par de peticiones paralelas mediante la función *get* de jQuery que nos permita mostrar en el párrafo denominado “objetivo” el texto *“Jose Luis Núñez tiene el rol de administrador”*.

4. Utiliza el método *load* de jQuery para cargar en “parrafo1” y “parrafo2” el contenido del fichero de texto *1.txt* y lo que contiene la caja contenedor del fichero *2.html*.

5. Podemos serializar datos utilizando jQuery para enviar datos al servidor. Utiliza los metodos *serialize*() y *serializeArray*() sobre el objeto formulario para comprobar mediante consola cómo se enviarían los datos. Prueba también a crear un objeto en JavaScript y utilizar el método *param(obj)* para visualizar su codificación.

6. Create una cuenta en **OpenWeatherMap**, obtén una API key para obtener el tiempo actual y muestra en el párrafo identificado como “objetivo” la temperatura. Considera la posibilidad de que el servidor no esté disponible o la solicitud sea inválida para mostrar un mensaje de error.

7. Create una cuenta en **Freegeoip** y muestra en el párrafo “objetivo” datos consumidos de dicha API.

8. Create una cuenta en **IPAPI** y muestra en el párrafo “objetivo” tu dirección IP pública y el país en el que te encuentras.

9. Visita el sitio web https://github.com/toddmotto/public-apis y realiza algunos ejercicios de consumo de datos de diferentes APIs públicas.