XML

XML (o eXtensible Markup Language) és un format de text humanament llegible que s'utilitza per l'emmagatzematge i l'intercanvi de dades estructurades, especialment a través d'internet.

És un llenguatge de marques, igual que l'HTML, però l'XML permet crear les teves pròpies etiquetes.

Sintaxi de les dades emmagatzemades en format XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<cotxes>
    <cotxe matricula="E-0198-LYX">
        <combustible>diesel</combustible>
        <canviManual>false</canviManual>
        <cv>170</cv>
        <companyia>
            <marca>opel</marca>
            <model>insignia</model>
        </companyia>
        <kilometres/>
    </cotxe>
    <cotxe matricula="E-5213-DWL">
        <combustible>gasolina</combustible>
        <canviManual>true</canviManual>
        <cv>140</cv>
        <companyia>
            <marca>jeep</marca>
            <model>wrangler</model>
        </companyia>
        <kilometres>70000</kilometres>
    </cotxe>
</cotxes>
```

Figura 1 : estructuració dades XML

Eina online per verificar la correcció del format XML

https://codebeautify.org/xmlvalidator

Ús d’un servidor pel treball amb fitxers XML

El navegador web no ens permetrà obrir un arxiu XML que no provingui d’un servidor HTTPS, ja que és una vulneració de les polítiques de seguretat de CORS (Cross-Origin Requests).

Per tant, per poder obrir els fitxers, haurem d’instal·lar i arrancar un servidor ben senzill. Això ho podem fer amb la instrucció:

```
npm install -g http-server
```

Abans però haurem de tenir accés al sistema de gestió de paquets npm. Si no hi tenim accés, instal·larem Node.js, que l’inclou. Ens el podem baixar des de:

https://nodejs.org/en/

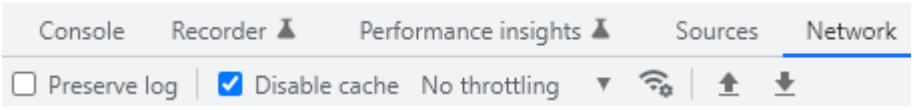
Un cop instal·lat http-server obrim la terminal de comandes de Node.js, ens ubiquem al directori on tenim el projecte i arranquem el servei introduïnt:

```
http-server
```

Si ara obrim el navegador al localhost i al port 8080 (el servidor ofereix IPs i ports alternatius) podrem accedir als nostres fitxers i executar-los des del nostre petit servidor HTTP i d’aquesta manera ja no tindrem restriccions d’accés.

http://127.0.0.1:8080/books.html

Haureu de refrescar el contingut del servidor o deshabilitar la caché (amb Google Chrome ho podeu fer des de Network / Disable cache).



De forma alternativa, podem instal·lar l’extensió **Live Server** de Visual Studio Code.

Obrir un fitxer XML des d'un servidor

El JavaScript que necessitem usar per llegir les dades d'un fitxer XML arranca des d'un fitxer HTML, així l'importem:

```
<html>
	<script type="text/javascript" src="xml.js"></script>
</html>
```

Al JavaScript hi hem d'especificar:

```
1.	var xhr = new XMLHttpRequest();
2.	xhr.onreadystatechange = function()
3.	{
4.		if (this.readyState == 4 && this.status == 200) {
5.			ops(this);
6.		}
7.	};
8.	xhr.open("GET", "cotxes.xml", true);
9.	xhr.send();
```

Aquest procediment és asíncron. Aquesta és l'ordenació segons l'odre en el que s'executen:
Línia 1: Creem un objecte XMLHttpRequest d'AJAX.
Línia 2: Aquesta funció s'executa cada vegada que canvia l'estat de l'objecte XMLHttpRequest.
Línia 8: Obre el fitxer .xml especificat.
Línia 9: Aquesta sol·licitud canvia l'estat de l'objecte mitjançant un esdeveniment.
Línia 4: Valida que la resposta ha arribat i que es correcta.
Línia 5: Envia a la funció l'objecte XMLHttpRequest amb el contingut obtingut del fitxer .xml.

Formatejar l'objecte XMLHttpRequest

Un cop emplenat l'objecte XMLHttpRequest amb les dades que necessitem li podem aplicar els següents mètodes:

Convertir en string:

```
var stringResponse = xhr.responseText
```

Convertir en objecte DOM XML:

```
xhr.responseXML
```

Per altra banda, també podem convertir un string en format XML en objecte DOM XML de la següent manera:

```
var parserXML = new DOMParser()
var xmlDoc = parserXML.parseFromString(stringResponse,"text/xml")
```

Per tal de treballar amb l'objecte DOM XML prèviament s'ha d'obtenir el contingut del document:

```
xmlDoc.documentElement
```

O n'obtenim el primer element de forma genèrica:

```
xmlDoc.childNodes[0]
```

Obtenir dades concretes a partir de l'objecte XML DOM obtingut

Obtenir tots els nodes:
xmlDoc.querySelectorAll("*")

Obtenir la col·lecció d'elements que hi ha dins l'etiqueta «combustible» (*Figura 1*):
var elem = *xmlDoc* .getElementsByTagName("combustible")

Obtenir l'atribut «matricula» del primer «cotxe» (*Figura 1*):
xmlDoc.getElementsByTagName("cotxe")[0].getAttribute("matricula")

Obtenir les propietats dels elements

«elem» és una col·lecció d'elements de manera que per iterar-los tots haurem d'usar un bucle.
for(var i=0; i < *elem*.length; i++) { ... }

Obtenir el contingut textual del node:
elem[i].nodeValue

Obtenir el contingut textual del node i dels seus descendents:
elem[i].textContent

Obtenir els atributs del node:
elem[i].attributes

Obtenir el nom del node:
elem[i].nodeName

Obtenir el tipus del node:
elem[i].nodeType

Abans d'accedir a les dades d'un node hem de saber de quin tipus és per tractar-lo en conseqüència.

Aquests són els tipus de node més habituals:

Tipus de node	
Element	1
Attribute	2
Text	3
Comment	8
Document	9

Navegar pel DOM

Mètodes que permeten navegar pel DOM. Es poden dividir en dos grups, els que retornen el contingut del node i els que retornen elements node. Tingues en compte que el primer tipus pren els espais, tabulacions, salts de línia, etc com a nous nodes.

Obtenir el contingut del primer fill del node:
elem[i].firstChild

Obtenir el primer element fill del node:
elem[i].firstElementChild

Obtenir el contingut de l'últim fill del node:
elem[i].lastChild

Obtenir l'últim element fill del node:
elem[i].lastElementChild

Obtenir el contingut dels nodes fills:
elem[i].childNodes

Obtenir els elements fills del node:
elem[i].children

Obtenir el contingut del node pare:
elem[i].parentNode

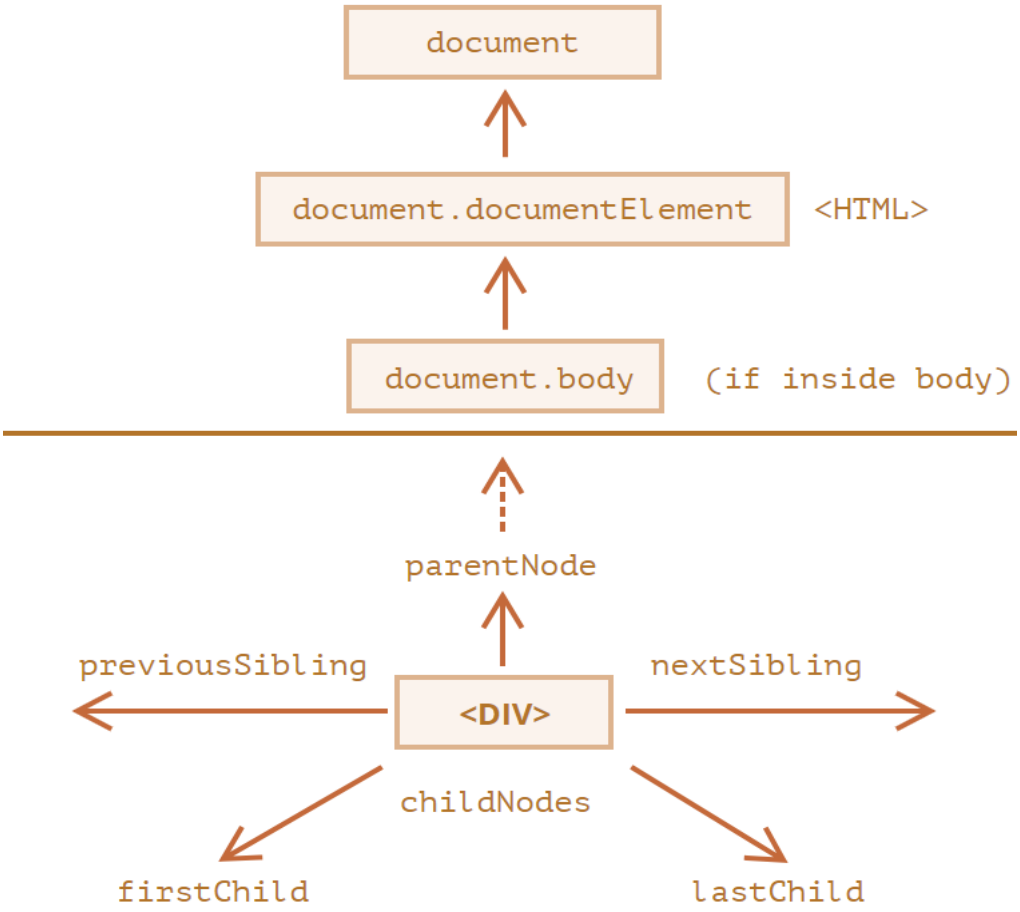
Obtenir l'element node pare:
elem[i].parentElement

Obtenir el contingut del node germà anterior:
elem[i].previousSibling

Obtenir el contingut del node germà següent:
elem[i].nextSibling

Obtenir l'element node germà anterior:
elem[i].previousElementSibling

Obtenir l'element node germà següent:
elem[i]. nextElementSibling



Exemple navegació pel DOM

Tenim el fitxer cotxes.xml on hi emmagatzemem informació tècnica sobre alguns cotxes.

Hem d'extreure aquesta informació i mostrar-la al navegador.
La funció ha de ser dinàmica de manera que si les etiquetes que inclouen els cotxes canvien continuï funcionant.

En primer lloc obtenim la informació de l'XML usant: *XMLHttpRequest*.
Obtenim l'etiqueta arrel i iniciem l'algoritme a partir d'ella, i ho fem de la següent manera:
Mostrem el contingut de l'etiqueta i els seus atributs i després fem el mateix pels seus fills. Un cop finalitzada la branca passem a la següent etiqueta i repetim la seqüència.

Pots consultar l'exemple resolt aquí: https://github.com/xbaubes/DAM/tree/main/MP4-Llenguatges-marques-SGI/XML/exercici0

ACTIVITATS

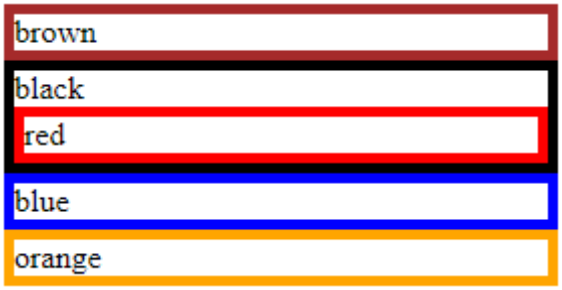
1. Usa com a base: https://github.com/xbaubes/DAM/tree/main/MP4-Llenguatges-marques-SGI/XML/exercici1

Obté la informació del fitxer books.xml.

Al fer click al botó, mostrar a div#showResults, en forma de taula amb capçalera, tota la informació dels llibres en que coincideixin amb la informació sol·licitada des de input#intro.
El camp de cerca varia segons el camp seleccionat a select#ops.

2. A partir d'un fitxer XML amb etiquetes de colors de fins a dos nivells com aquest: https://github.com/xbaubes/DAM/tree/main/MP4-Llenguatges-marques-SGI/XML/exercici-2-createHTML

Has de crear un algorisme que converteixi aquestes dades en elements HTML.
Per cada etiqueta de color hauràs de crear un div, a aquest div hi has d'escriure el color de l'etiqueta i assignar-li un contorn visible d'aquest color. Tal com es mostra a la imatge:



Utilitza la següent capçalera per la funció des d'on crearàs els DIV's:
function createDiv (pareDOM, fillXML)

A aquesta funció li has de passar l'element d'on ha de penjar el div que crearàs i el contingut d'aquest div que has obtingut de l'XML. Has de retornar l'element div creat.
Dins de la funció hi has de crear un div, assignar-li border i contingut i afegir-lo com a fill de l'element passat.

Hi ha vàries diferències respecte d'aquest exercici:
https://github.com/xbaubes/DAM/tree/main/Llenguatges-marques/XML/exercici-0
Els tags tenen noms diferents, potser hauràs d'utilitzar la propietat **nextElementSibling** per moure't entre germans del fitxer XML.
Els elements s'han de crear, no anar afegint contingut a un div.

BIBLIOGRAFIA I WEBGRAFIA

«AppsLoveWorld». https://www.appsloveworld.com/free-online-sample-xml-api-for-testing-purpose
«Quackit». https://www.quackit.com/xml/tutorial/
«MDN Web Docs». https://developer.mozilla.org/es/docs/Web/HTTP/CORS
«Microsoft». Microsoft docs books.xml
«W3 Schools». https://www.w3schools.com/xml/