TEMA 8. JSON y XML .

Desarrollo Web en Entorno Cliente.

Profesor: Juan José Gallego García

Índice :

JSON

¿Qué es JSON?

Es un tipo de sintaxis para almacenar e intercambiar datos en formato de texto. Las siglas JSON significan **J**ava**S**cript **O**bject **N**otation (notación de objetos de JS).

Sintaxis

La sintaxis es casi idéntica que para un objeto JS: `{nombre:"Juan", apellido:"Gallego"}` formato JSON: `{"nombre":"Juan", "apellido":"Gallego"}` , en este último la clave debe de ir entre comillas dobles, todo ello encerrado entre comillas simples (como cadena) estaría preparado para usarlo como intercambio.

```
var cadenaJSON='{"nombre":"Juan", "apellido":"Gallego"}';
```

Para almacenar JSON en archivos con extensión **.json**, para usarlo en intercambio el tipo MIME es "**application / json**"

Tipos de valores que admite JSON

Los valores que permite el formato JSON pueden ser : Un string, un number, otro objeto JSON, un array, un booleano, o null.

```
var objJSON = {
    "alumno":"Juan" , // String
    "curso":2 , // Number
    "modulos": ["m1","m2","m3"] , // Array
    "domicilio":{"calle":"Cerro","num":3} // Otro JSON
    };
```

```
var cadenaJSON = `{
    "alumno":"Juan" ,
    "curso":2 ,
    "modulos": ["m1","m2","m3"] ,
    "domicilio":{"calle":"Cerro","num":3}
    }`;
```

```
var objJS = {
    alumno:"Juan" ,
    curso:2 ,
    modulos: ["m1","m2","m3"] ,
    domicilio:{"calle":"Cerro","num":3}
    };
```

objJSON → Objeto con sintaxis JSON.
cadenaJSON → Cadena con sintaxis JSON para intercambio. Se puede observar que se usa “ ` “ para incluir los saltos de línea de la cadena.
objJS → Objeto con sintaxis JavaScript. JS puede referenciar igualmente objJSON como objeto.

Métodos JSON.parse( ) y JSON.stringify( )

Para el intercambio de datos con un servidor se debe usar formato texto en un formato determinado por ejemplo JSON, por ello es necesario enviar/recibir los datos como cadena. La función JSON.parse( ), convierte una cadena recibida desde el servidor en un objeto JS, por el contrario JSON.stringify( ), convierte a cadena un objeto JS para poder ser enviado al servidor u otro dispositivo de intercambio.

```
// Suponemos cadena recibida del servidor
var cadenaJSON=`{
    "alumno":"Juan" ,
    "curso":2 ,
    "modulos": ["m1","m2","m3"] ,
    "domicilio":{"calle":"Cerro","num":3}
    }`;
var objJS=JSON.parse(cadenaJSON); // Convierte en objeto JS
alert (objJS.alumno); // Muestra nombre de alumno.
```

```
var objJS={     // Objeto JS
    alumno:"Juan" ,
    curso:2 ,
    modulos: ["m1","m2","m3"] ,
    domicilio:{calle:"Cerro",num:3}
    };
var cadenaJSON=JSON.stringify(objJS);
alert (cadenaJSON); // Muestra cadena JSON.
```

Ejemplo de intercambio con archivo PHP

El siguiente ejemplo recibe datos JSON del servidor y los convierte en objeto JS

```
<!DOCTYPE html>
<html>
<body>
<h2>Recibe datos en JSON desde un servidor por PHP con AJAX.</h2>
<p id="demo"></p>
<script>
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    myObj = JSON.parse(this.responseText);
    document.getElementById("demo").innerHTML = myObj.nombre;
  }
};
xmlhttp.open("GET", "json1.php", true);
xmlhttp.send();
</script>
</body>
</html>
```

```
//  json1.php
<?php
header("access-control-allow-origin: *");
$myObj = new stdClass();
$myObj->nombre = "Juan";
$myObj->ciudad = "Ubrique";
$myJSON = json_encode($myObj);
echo $myJSON;
?>
```

Ver que se ha añadido en el archivo PHP la cabecera (header) para permitir llamadas AJAX entre distintos dominios . Ver CORS y JSONP

Ejemplo de almacenamiento local (navegador) en JSON.

Podemos usar JSON para guardar y recuperar información en el navegador en forma de objetos JS .

```
<script>

// Guardar datos
var objJS={alumno:"Juan" , curso:2 };
var cadenaJSON=JSON.stringify(objJS);
localStorage.setItem("datos", cadenaJSON);

// Recuperar datos
var datos = localStorage.getItem("datos");
obj = JSON.parse(datos);
alert (obj.alumno); // Muestra "Juan"

</script>
```

XML

¿Qué es XML?

Siglas en inglés de eXtensible Markup Language, traducido como "Lenguaje de Marcado Extensible" o "Lenguaje de Marcas Extensible", es un meta-lenguaje para almacenar e intercambiar datos como texto con una estructura jerárquica .

```
<?xml version="1.0" encoding="UTF-8"?>
 <cine>
    <película categoría="acción">
           <título idioma="inglés">Mad Max</título>
            <director>George Miller</director>
            <estreno>15 mayo 2015</estreno>
            <reparto>Charlize Theron</reparto>
     </película>
     <película categoría="animación">
           <título idioma="inglés">Inside Out</título>
           <reparto>Amy Poehler</reparto>
</película>
 </cine>
```

Ejemplo de documento XML
(ej1.xml)

Para manejar documentos XML con JS se usa el DOM, con casi las mismas propiedades y métodos usados para el DOM de HTML. Por tanto podemos obtener, crear, modificar y borrar elementos del documento XML.

Un documento XML está compuesto principalmente por nodos elementos, nodos atributos y nodos textos. Por ejemplo en el XML anterior el nodo raíz cine posee dos nodos elementos película y a su vez película posee un nodo elemento título que contiene un nodo texto “Mad Max”, y un nodo atributo; idioma.

Propiedades (principales) del DOM del XML
Siendo x un nodo elemento cualquiera:

- **x.nodeName** - el nombre de x
- **x.nodeValue** - el valor de x
- **x.parentNode** - el nodo padre de x
- **x.childNodes** - los nodos hijos de x
- **x.childNodes.length** - número de elementos de x
- **x.attributes** - los atributos de los nodos x

Métodos (principales) del DOM del XML
Siendo x un nodo elemento cualquiera:

- **x.getElementsByTagName (*nombre*)** - obtener todos los elementos con un nombre de etiqueta especificada.
- **x.appendChild (*nodo*)** - añadir un nodo secundario a x
- **x.removeChild (*nodo*)** - eliminar un nodo hijo de x

Tomando como referencia el documento ej1.xml anterior lo guardamos en cadena :

```
var parser = new DOMParser();
var xmlDoc = parser.parseFromString(cadena,"text/xml");
alert(xmlDoc.getElementsByTagName("película").length); // Muestra 2
alert(xmlDoc.getElementsByTagName("título")[0].childNodes[0].nodeValue); // Muestra 'Mad Max'
alert(xmlDoc.getElementsByTagName("título")[0].attributes[0].nodeValue); // Muestra 'inglés'
alert(xmlDoc.getElementsByTagName("título")[0].getAttribute("idioma")); // Muestra 'inglés'
xmlDoc.getElementsByTagName("título")[0].childNodes[0].nodeValue="Mad Max 2"; // Cambia el valor del nodo
xmlDoc.getElementsByTagName("título")[0].setAttribute("idioma","francés"); // Cambia el valor del atributo
alert(xmlDoc.getElementsByTagName("película")[0].getElementsByTagName("reparto")[0].childNodes[0].nodeValue); //
Con la línea anterior se muestra 'Charlize Theron'
var x=xmlDoc.getElementsByTagName("película")[0];
xmlDoc.documentElement.removeChild(x); // Borra el primer nodo 'película' y todos sus hijos.
var cine= xmlDoc.getElementsByTagName("cine")[0];
var peli= xmlDoc.createElement("película"); // Crea nodo película.
var tit=xmlDoc.createElement("título");  // Crea nodo título.
var textTit= xmlDoc.createTextNode("Mad Max3"); // Crea un nodo texto
tit.appendChild(textTit); // Añade el texto al título
peli.appendChild(tit); // Añade el título a la peli
cine.appendChild(peli); // Añade la peli al nodo raíz 'cine'
```

```
// Para listar todos los títulos.
var x = xmlDoc.getElementsByTagName('título');
var txt="";
for (i of x) {
    txt += i.childNodes[0].nodeValue + "\n";
}
alert(txt);
```

Como prueba , vamos a escribir el código para listar los actores (reparto) de cada película.

Podríamos haber cargado el archivo XML externo con Ajax en lugar de asignar el contenido XML a la variable como string, se verá en el siguiente tema.

JSON vs XML

El siguiente ejemplo compara la estructura de un array de 3 objetos (alumnos) en JSON y XML

```
{"alumnos":[
  { "nombre":"Juan", "apellido":"García" },
  { "nombre":"Pedro", "apellido":"Pérez" },
  { "nombre":"Pepe", "apellido":"Blanco" }
]}
```

```
<alumnos>
  <alumno>
    <nombre>Juan</nombre> <apellido>García</apellido>
  </alumno>
  <alumno>
    <nombre>Pedro</nombre> <apellido>Pérez</apellido>
  </alumno>
  <alumno>
    <nombre>Pepe</nombre> <apellido>Blanco</apellido>
  </alumno>
</alumnos>
```

Algunas ventajas de JSON sobre XML:

- No usa cierre de etiqueta, por tanto es más corto.
- Es más rápido de leer y escribir.
- Puede hacer uso de arrays.
- JSON se puede analizar con una simple función (parse) de JS convirtiendo una cadena en objetos, XML necesita un analizador para XML, por ejemplo a través del DOM teniendo que usar varios métodos, esta ventaja de JSON es muy destacable para peticiones AJAX.

Bibliografía

- LibrosWeb
- W3Schools
- Developer Mozilla Docs