Introducción

A día de hoy, el formato JSON se usa masivamente, sobre todo en el mundo del desarrollo web. Aprenderemos aquí **qué es** y las **características** que lo han hecho el protagonista del mundo del desarrollo.

JSON (JavaScript Object Notation)

Es un formato de texto ligero, estándar y abierto que nace del lenguaje JavaScript. Se utiliza en general para el intercambio de datos.

Está basado en un sistema de generación de objetos en JavaScript: **JavaScript Object Notation**.

Es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

La información en formato JSON se transfiere mediante como respuesta a una petición de un servicio web, REST, llamada AJAX, API y en un fichero de texto plano con el formato JSON y extensión **.json**.

El “media type” que utiliza en Internet es: **application/json**.

Su identificador de tipo uniforme (Uniform Type Identifier) es: **public.json**

Características

- Es un formato liviano que implica una menor cantidad de datos para transmitir una misma información, comparándola con otros formatos como XML.
- Legible por el ser humano.
- Fácilmente convertible en objetos para su tratamiento en un desarrollo orientado a objetos, valga la redundancia.
- Simple tanto para procesos de lectura y escritura.
- Independiente del lenguaje que lo consuma (es otro formato de texto plano, al fin y al cabo).

Ofrece diferentes ventajas sobre XML pero también alguna carencia: **no puede detallar la información que transmite, al nivel de XML**.

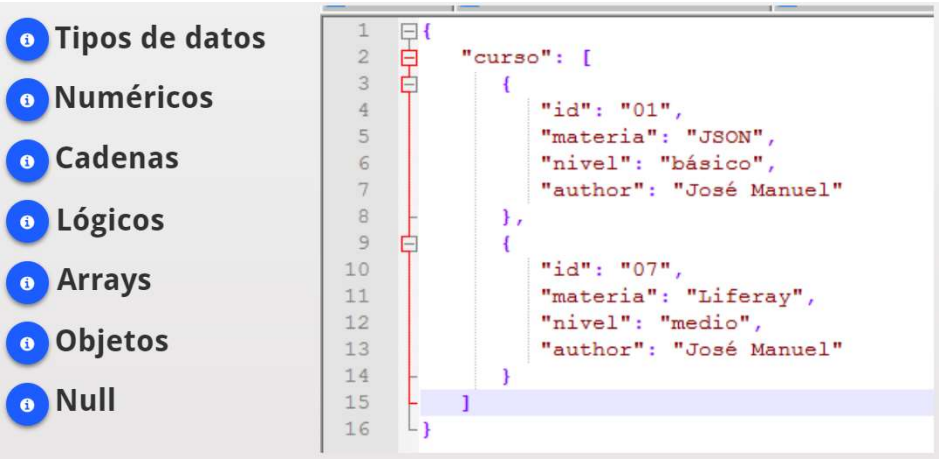
Esto hace que no sustituya a XML salvo en aquellas situaciones donde no se necesita ese nivel de detalle.

Formato JSON

¿Cómo se forma un contenido con sintaxis JSON? Aprendamos la base del porqué es ideal para intercambiar información en desarrollos orientados a objeto. JSON hereda de JavaScript su sintaxis (JavaScript Object Notation):

- Miembros separados por coma **,**.
- Llaves **{…}** para identificar los objetos.
- Corchetes **[…]** para identificar arrays.
- Los datos se guardan en pares **clave : valor**.
- Los identificadores de los miembros, se entrecomillan con comillas dobles.

Tipos de datos



Los valores de un par clave: valor o miembro de un objeto JSON pueden tener los siguientes valores:

Número:
En formato JavaScript, entero, flotante o doble: **3**, **7.3**, etc.

Cadena:
Caracteres Unicode encerrados entre comillas dobles que permiten escape de caracteres individuales.

Lógicos:
Valores **true** o **false**.

Array:
Delimitados entre corchetes, son una colección de miembros.

Objeto:
Delimitados entre llaves, pueden ser objetos miembro de otros objetos y arrays.

Espacio en blanco:
Podemos usarlo entre diferentes tokens o caracteres.

---

null:
Valor vacío.

Ejemplos

JSON

```
{
  "nombre": "Juan",
  "apellidos": "Suárez Iglesias",
  "edad": 25,
  "direccion": {
    "calle_y_num": "Acacias, nº34",
    "ciudad": "Valencia",
    "codigo_postal": "43005",
    "pais": "Colombia",
  },
  "telefono": [
    {
      "tipo": "fijo",
      "numero": "0034966432134"
    },
    {
      "tipo": "móvil",
      "numero": "0034677493826"
    }
  ]
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<nombre>Juan</nombre>
<apellidos>Suárez Iglesias</apellidos>
<edad>25</edad>
<direccion>
  <calle_y_num>Acacias, nº34</calle_y_num>
  <ciudad>Valencia</ciudad>
  <codigo_postal>43005</codigo_postal>
  <pais>Colombia</pais>
</direccion>
<telefono>
  <tipo>fijo</tipo>
  <numero>0034966432134</numero>
</telefono>
<telefono>
  <tipo>móvil</tipo>
  <numero>0034677493826</numero>
</telefono>
```

Herramientas JSON

Cualquier editor de textos puede servirnos para gestionar contenidos JSON. Sin embargo, el coloreado de sintaxis entre otras funcionalidades, permitirán una mayor productividad.

Dependiendo de lo que necesitemos de JSON, podemos encontrarnos herramientas, muchas de ellas en línea que nos proveerán entre otras cosas de verificación de la sintaxis, generación de contenido, etc.

En la práctica, el software que desarrollemos será el que deba leer, procesar y generar contenidos JSON.

Editores y otras herramientas

Editores

Cualquier editor es válido para generar objetos JSON. Desde el bloc de notas de Windows, pasando por nano o vim en entornos Linux, hasta otros mucho más sofisticados como UltraEdit o el ya clásico Notepad++.

Editores online

Los editores en línea nos permiten prescindir de tener una herramienta adicional instalada en nuestro sistema:

JSON Editor Online, JSON formatter, Code Beautify, Clean CSS, JSON Viewer

Validadores

Cualquier editor capaz de reconocer el formato JSON para algo más que colorear la sintaxis, nos servirá como validador del código escrito en este formato. Sin embargo, una vez más disponemos de validadores online:

JSON Lint, JSON Formatter & Validator, Freeformatter, CodeBeautify

Generadores

Esta herramienta de la que encontramos muchas posibilidades online, nos permite entre otras cosas generar código JSON que nos puede ser útil para muy diversos fines como la realización de pruebas:

JSON Generator, ObjGen, Mockaroo, Online JSON Data Generator, The One Generator, Minecraft JSON Generator

Disponemos de otras posibilidades como conversores entre formatos. De éstos, encontramos gran cantidad en línea, como http://www.csvjson.com/csv2json

Lenguajes

JSON es “agnóstico” en cuanto a los lenguajes que lo utilizan. Se trata, en cualquier caso, de un fichero de texto que podemos procesar de mil formas.

Sin embargo, su formato concreto, hace que lenguajes como JavaScript lo pueda tratar con suma facilidad y otros, como PHP, Python, etc., puedan usarlo rápidamente para desarrollos orientados a objeto.

El objeto JSON en JavaScript

JSON está totalmente relacionado con JavaScript. En principio, un contenido JSON será tratado como objeto en JavaScript directamente siempre y cuando no lo asignemos como cadena.

En caso contrario, nos veremos obligados a realizar conversiones de cadena a objeto JSON y viceversa. De hecho, es una situación muy común en llamadas AJAX y servicios REST.

El objeto JSON de JavaScript provee de algunos métodos que realizan estas conversiones tan útiles.

```
var strJson = "{
  "nombre" : "Yo",
  "nota" : "sobresaliente"
}";
var objJson = JSON.parse(strJson);
console.log(objJson.nombre);
var strCadena = JSON.stringify(objJson);
console.log(strCadena);
```

Un poco más de información sobre el objeto JSON: https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/JSON

Lenguajes que usan JSON

Tratándose de un formato de texto plano, en principio, cualquier lenguaje con capacidad para leer y escribir en ficheros de texto, puede usarlo.

La mayor ventaja la obtienen aquellos lenguajes con orientación a objetos como Java, PHP, Python, JavaScript, etc., que tienen más fácil tratar la información en formato JSON.

Java: hay diferentes librerías pero org.json es de las más conocidas.

PHP: con el uso de las funciones **json_decode()** y **json_encode()**, se facilita enormemente el trabajo con JSON.

Python: nuevamente, importar una librería es suficiente: import json.

- JavaScript: JSON se puede usar directamente con JavaScript si el objeto JSON no se ha recibido como cadena. En caso contrario, haremos uso del objeto JSON de JavaScript.
- Go: una pequeña introducción: https://gobyexample.com/json.

Por supuesto existen más lenguajes, pero gran parte de ellos, a día de hoy, o bien pueden acceder directamente o disponen de alguna librería que provee de la funcionalidad para hacerlo.

Uso

Las características de JSON lo hacen idóneo para diferentes funciones: desde configuración de proyectos y herramientas, consumo de servicios a registros en bases de datos.

Aplicaciones de JSON

A grandes rasgos, el uso de JSON se puede aplicar a cualquier situación donde tengamos que intercambiar información, pero:

- Enviar y recibir información de servicios web.
- Enviar y recibir información de servicios REST.
- Uso de GraphQL.
- Enviar y recibir información de llamadas AJAX.
- Uso en NoSQL.
- Configuraciones de aplicaciones y herramientas.
- Generación del esqueleto de proyectos.
- Gestión de información en JavaScript mediante **localStorage**, **sessionStorage** e **indexedDB**.
- ¿Necesidades y desarrollos personalizados?

Las posibilidades son muchísimas.