LENGUAJES DE MARCAS

Tratamiento de documentos JSON

Autores: Diana Bautista / Óscar Villar
Revisado: Javier García

Índice de contenido

1. Introducción

Introducción

- **JSON** (acrónimo de **JavaScript Object Notation**, «notación de objeto de JavaScript») es un formato de texto sencillo para el intercambio de datos
- Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que resulta mucho más sencillo escribir un analizador sintáctico (parser) para él.
- Gran aceptación por la comunidad de desarrolladores AJAX
- Si bien se tiende a considerar JSON como una *alternativa* a XML, lo cierto es que no es infrecuente el uso de JSON *y* XML en la misma aplicación
- Una aplicación de cliente que integra datos de Google Maps con datos meteorológicos en SOAP necesita hacer uso de ambos formatos.

Introducción

.json es un archivo que contiene una serie de datos estructurados en formato de texto y se usa para transferir información entre sistemas. Es importante decir que, a pesar de su origen estar en el lenguaje JavaScript, JSON no es un lenguaje de programación.

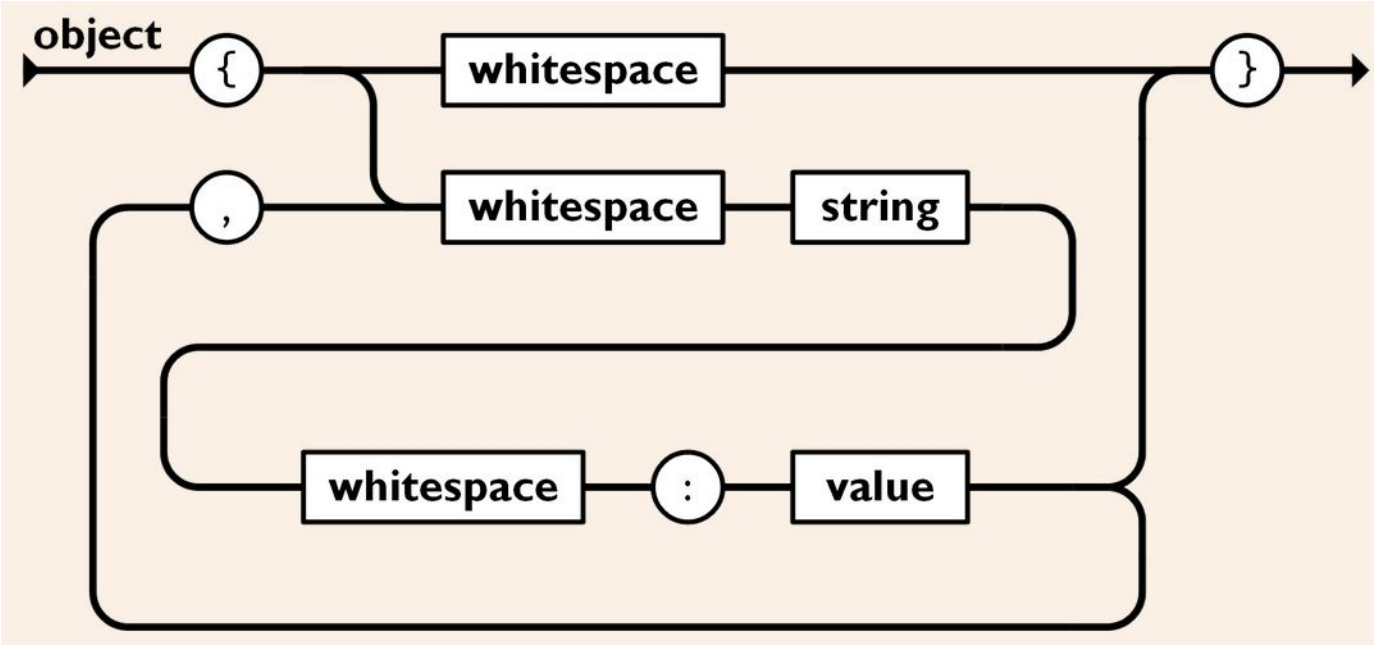
JSON es una notación para la transferencia de datos que sigue un estándar específico. Por eso, puede emplearse en diferentes lenguajes de programación y de sistemas.

Los datos contenidos en un archivo en formato JSON deben estructurarse por medio de una colección de pares con nombre y valor o deben ser una lista ordenada de valores. Sus elementos tienen que contener:

- **Clave**: corresponde al identificador del contenido. Por eso, debe ser una string delimitada por comillas.
- **Valor**: representa el contenido correspondiente y puede contener los siguientes tipos de datos: string, array, object, number, boolean o null.

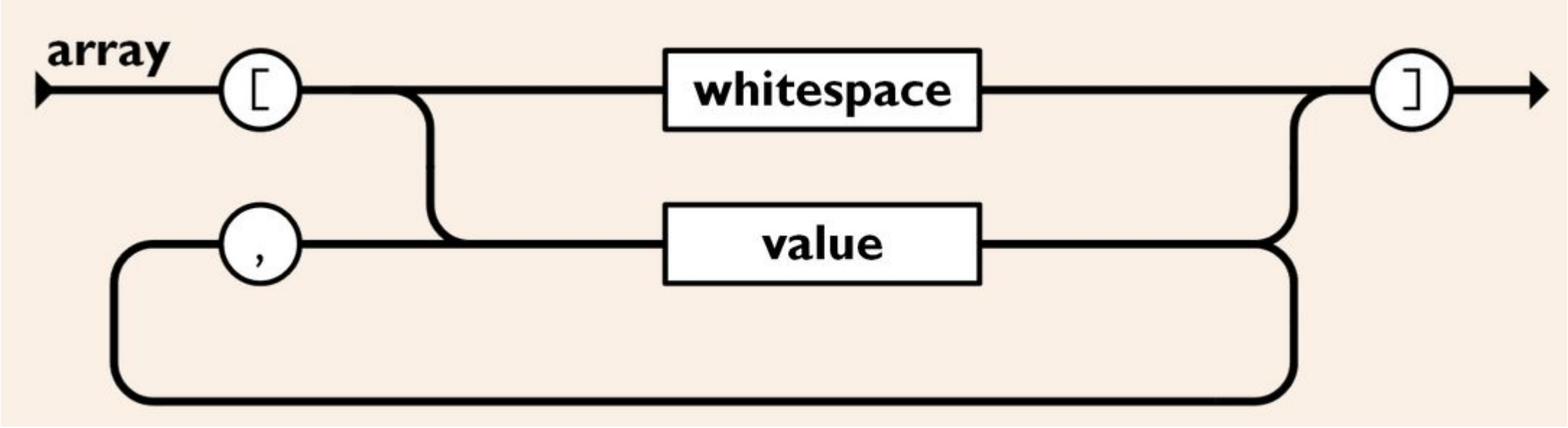
Reglas de sintaxis JSON

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con {llave de apertura y termine con }llave de cierre. Cada nombre es seguido por :dos puntos y los pares nombre/valor están separados por ,coma.



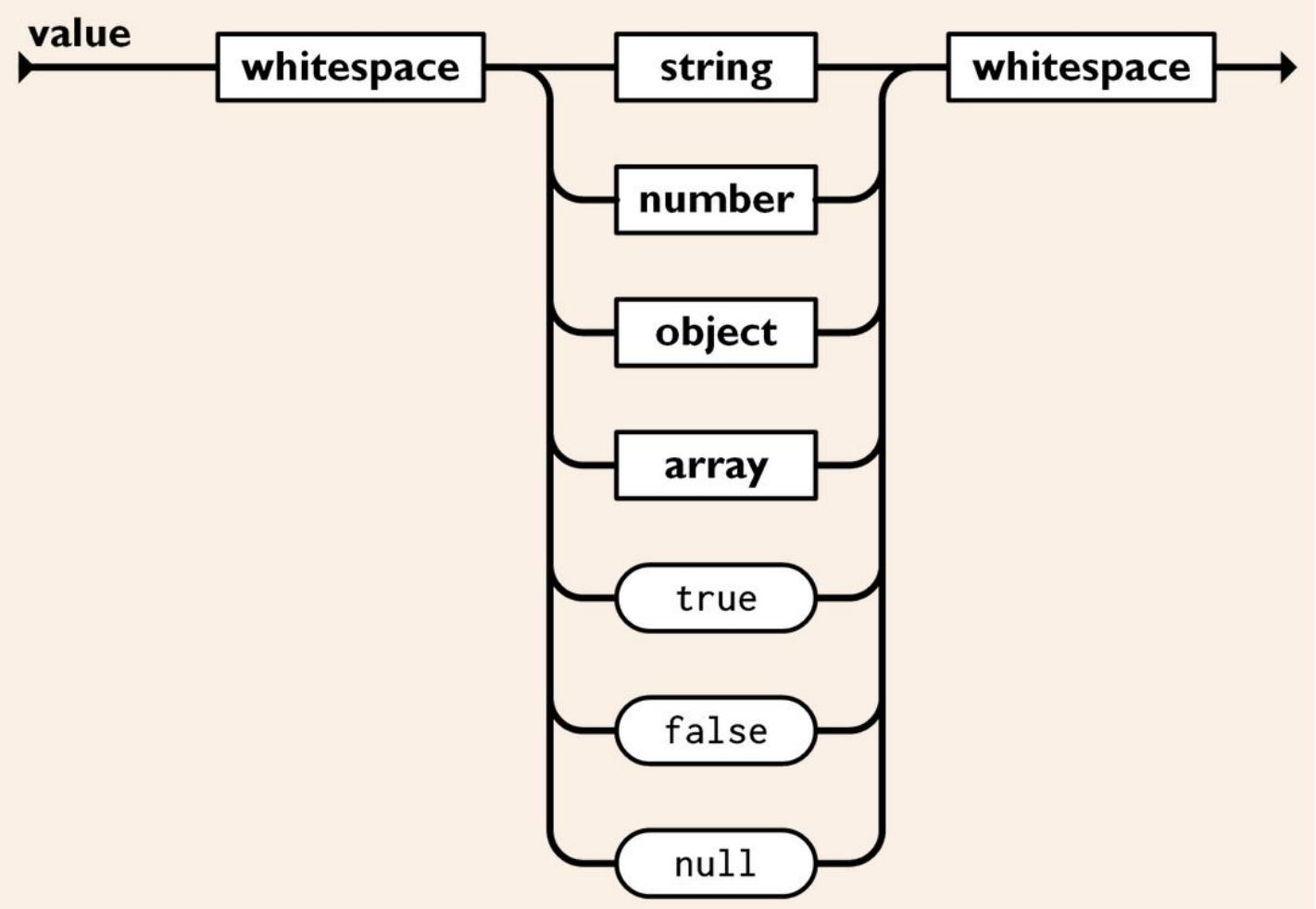
Reglas de sintaxis JSON

Un *array* es una colección de valores. Una array comienza con [corchete izquierdo y termina con]corchete derecho. Los valores se separan por ,coma.



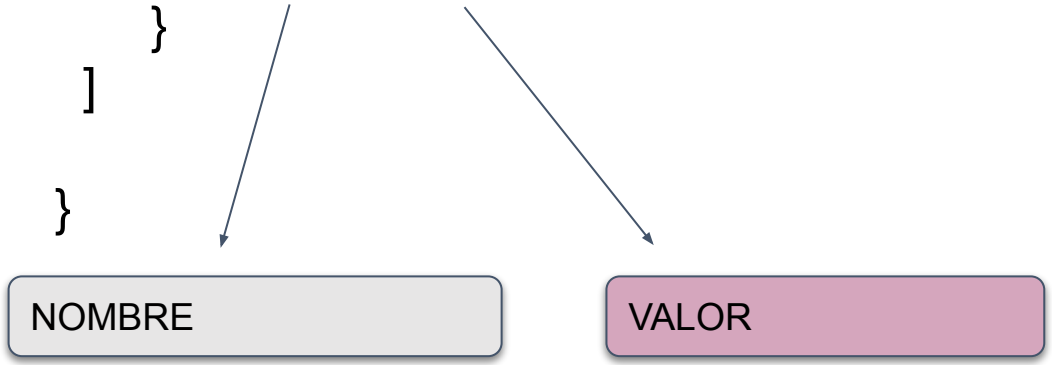
Reglas de sintaxis JSON

Un *valor* puede ser una *cadena de caracteres* con **comillas dobles,** o un *número*, o true o false o null, o un *objeto* o un *arreglo*. Estas estructuras pueden anidarse.



Reglas de sintaxis JSON

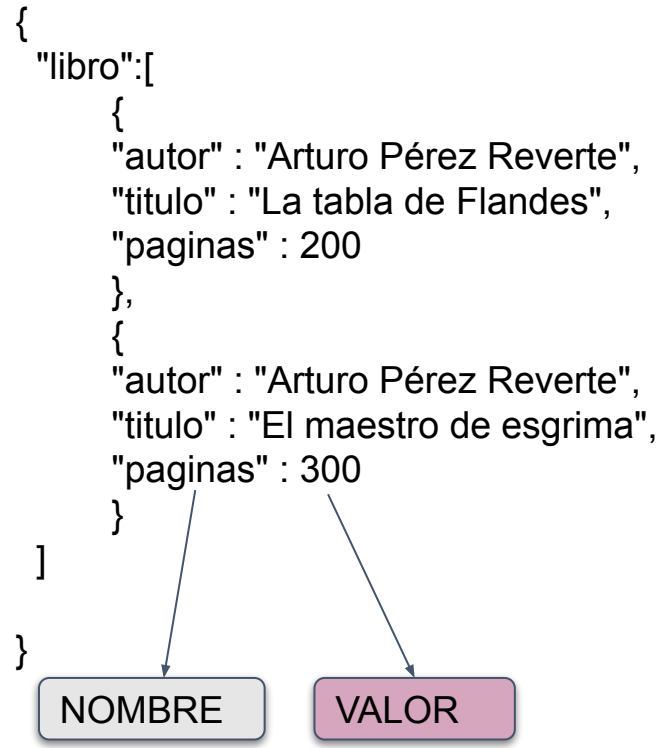
```
{
  "libro":[
      {
      "autor" : "Arturo Pérez Reverte",
      "titulo" : "La tabla de Flandes",
      "paginas" : 200
      },
      {
      "autor" : "Arturo Pérez Reverte",
      "titulo" : "El maestro de esgrima",
      "paginas" : 300
      }
  ]

}
```



Reglas de sintaxis JSON

Documentos JSON

- El tipo de archivo de los archivos JSON es ".json"
- El tipo MIME para el texto JSON es "application / json"



Tipos de datos válidos

VALORES PERMITIDOS

En JSON, los valores deben ser uno de los siguientes tipos de datos:

- una cadena
- un número
- un objeto (objeto JSON)
- una matriz
- un booleano
- *nulo*

VALORES NO PERMITIDOS

Los valores JSON **no pueden** ser uno de los siguientes tipos de datos:

- Una función
- una fecha
- *indefinido*

Tipos de datos válidos

CADENAS EN JSON

Las cadenas en JSON deben escribirse entre comillas dobles.

```
{ "nombre":"Pedro" }
```

NÚMEROS JSON

Los números en JSON deben ser un número entero o un punto flotante.

```
{ "edad":30 }
```

MATRICES JSON

Los valores en JSON pueden ser matrices.

```
{
"empleados":[ "Pedro", "Luis", "Ana" ]
}
```

OBJETOS JSON

Los valores en JSON pueden ser objetos.

```
{
"empleado":{ "nombre":"Pedro", "edad":30, "ciudad":"Valencia" }
}
```

BOOLEANOS EN JSON

Los valores en JSON pueden ser verdadero / falso.

```
{ "vendido":true }
```

JSON NULO

Los valores en JSON pueden ser nulos.

```
{ "software":null }
```

Ejercicio

Convierte este archivo XML a Json utilizando las estructuras permitidas.

EJEMPLO XML

```xml
<empleados>
  <empleado>
    <nombre>Pedro</nombre>
    <apellido>Sanchis</apellido>
  </empleado>
  <empleado>
    <nombre>Ana</nombre>
    <apellido>Lopez</apellido>
  </empleado>
  <empleado>
    <nombre>Juan</nombre>
    <apellido>Vidal</apellido>
  </empleado>
</empleados>
```

JSON frente a XML

EJEMPLO JSON

```
{"empleados":[
  { "nombre":"Pedro", "apellido":"Sanchis" },
  { "nombre":"Ana", "apellido":"Lopez" },
  { "nombre":"Juan", "apellido":"Vidal" }
]}
```

EJEMPLO XML

```
<empleados>
  <empleado>
    <nombre>Pedro</nombre>
    <apellido>Sanchis</apellido>
  </empleado>
  <empleado>
    <nombre>Ana</nombre>
    <apellido>Lopez</apellido>
  </empleado>
  <empleado>
    <nombre>Juan</nombre>
    <apellido>Vidal</apellido>
  </empleado>
</empleados>
```

JSON frente a XML

SEMEJANZAS

JSON es como XML porque:

- Tanto JSON como XML son "autodescriptivos" (legibles por humanos)
- Tanto JSON como XML son jerárquicos (valores dentro de valores)
- Tanto JSON como XML pueden ser analizados y utilizados por muchos lenguajes de programación.
- Tanto JSON como XML se pueden recuperar con XMLHttpRequest

DIFERENCIAS

JSON es diferente a XML porque:

- JSON no usa etiqueta de cierre
- JSON es más corto
- JSON es más rápido de leer y escribir
- JSON puede usar matrices

La mayor diferencia es:

XML debe analizarse con un analizador XML. JSON se puede analizar mediante una función estándar de JavaScript (función eval()).

¿Por qué el archivo JSON cada vez es más utilizado?

La simplicidad del formato JSON es una de las principales razones por las que es bastante utilizado.

Eso porque las peticiones AJAX, que permiten la actualización de la página sin la necesidad de recargarla completamente, deben ser ejecutadas con mucha rapidez para que esas actualizaciones sean transparentes para el usuario.

Por ser liviano y compacto, el formato JSON atiende a esa necesidad. Por lo tanto, los datos pueden transferirse de forma rápida e interpretarse con facilidad por la aplicación.

Cabe mencionar que el formato XML también se puede utilizar en peticiones AJAX. Sin embargo, es un archivo más grande, por contener más información en lo que se refiere al gran número de tags de apertura y cierre. Algo que torna su transferencia y procesamiento más lentos que el modelo JSON.

Enlaces de interés

1. XMLHttpRequest - Web APIs
2. W3 Schools JSON Introduction

Bibliografía

Prácticamente todo contenido del documento ha sido extraido de https://www.w3schools.com/js/js_json_intro.asp