UD 8. Tratamiento de documentos JSON

TEORÍA

Lenguaje de marcas y sistemas de gestión de la información

1ro CFGS DAW

2022/2023

Índice de contenido

1. Introducción

JSON (acrónimo de **JavaScript Object Notation**, «notación de objeto de JavaScript») es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera (año 2019) un formato independiente del lenguaje.

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. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval(), algo que (debido a la ubicuidad de JavaScript en casi cualquier navegador web) ha sido fundamental para que haya sido aceptado por parte de la comunidad de desarrolladores AJAX.

En la práctica, los argumentos a favor de la facilidad de desarrollo de analizadores o de sus rendimientos son poco relevantes, debido a las cuestiones de seguridad que plantea el uso de eval() y el auge del procesamiento nativo de XML incorporado en los navegadores modernos. Por esa razón, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo!, Google, Mozilla, etc, que atienden a millones de usuarios) cuando la fuente de datos es explícitamente de fiar y donde no es importante el hecho de no disponer de procesamiento XSLT para manipular los datos en el cliente.

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; así, una aplicación de cliente que integra datos de Google Maps con datos meteorológicos en SOAP necesita hacer uso de ambos formatos.

En diciembre de 2005, Yahoo! comenzó a dar soporte opcional de JSON en algunos de sus servicios web.[1]

2. Reglas de sintaxis JSON

La sintaxis JSON se deriva de la sintaxis de notación de objetos de JavaScript:

- Los datos están en pares de nombre / valor
- Los datos están separados por comas
- Las llaves sostienen objetos
- Los corchetes sostienen matrices

2.1. Datos JSON: un nombre y un valor

Los datos JSON se escriben como pares de nombre / valor.

Un par de nombre / valor consta de un nombre de campo (entre comillas dobles), seguido de dos puntos, seguido de un valor:

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

2.2 Valores JSON

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

- una cuerda
- un número
- un objeto (objeto JSON)
- una matriz
- un booleano
- nulo

En JSON, *los valores de cadena* deben escribirse con comillas dobles.

2.3 Archivos JSON

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

3. JSON frente a XML

JSON y XML se pueden usar para recibir datos de un servidor web.

Los siguientes ejemplos de JSON y XML definen un objeto utilizado, con una matriz de 3 empleados:

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 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

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.

4. Tipos de datos válidos

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*

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

- Una función
- una fecha
- indefinido

4.1. Cadenas en JSON

Las cadenas en JSON deben escribirse entre comillas dobles.

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

4.2. Números JSON

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

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

4.3. Objetos JSON

Los valores en JSON pueden ser objetos.

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

4.4. Matrices JSON

Los valores en JSON pueden ser matrices.

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

4.5. JSON Booleanos

Los valores en JSON pueden ser verdadero / falso.

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

4.6. JSON nulo

Los valores en JSON pueden ser nulos.

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

5. Función PARSE

Un uso común de JSON es intercambiar datos hacia / desde un servidor web.

Al recibir datos de un servidor web, los datos siempre son una cadena.

Analice los datos con JSON.parse()y los datos se convertirán en un objeto JavaScript.

5.1. Ejemplo: análisis de JSON

Imagina que recibimos este texto de un servidor web:

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

La función JavaScript JSON.parse() se usa para convertir texto en un objeto JavaScript:

```
var obj = JSON.parse('{ "nombre":"Pedro", "edad":30, "ciudad":"Valencia"}');
```

Si usamos esto en nuestra página deberíamos hacerlo entre etiquetas <script></script> donde pondremos el código Javascript

```
<!DOCTYPE html>
<html>
<body>

<h2>Crear un objeto desde un texto en JSON</h2>

<p id="demo"></p>

<script>
var txt = '{"nombre":"Pedro", "edad":30, "ciudad":"Valencia"}'
var obj = JSON.parse(txt); //obtenemos un objeto Javascript
document.getElementById("demo").innerHTML = obj.nombre + ", "
```

```
+ obj.edad; //Colocamos en el párrafo con id demo el nombre y la edad
</script>

</body>
</html>
```

Resultado en el navegador:

Crear un objeto desde un texto en JSON

Pedro, 30

5.2. JSON desde el servidor

Puede solicitar JSON desde el servidor mediante una solicitud AJAX

Siempre que la respuesta del servidor esté escrita en formato JSON, puede analizar la cadena en un objeto JavaScript.

Para obtener datos desde el servidor es necesario usar XMLHttpRequest:

```
<!DOCTYPE html>
<html>
<body>

<h2>Utilice XMLHttpRequest para obtener el contenido de un archivo.
</h2>
<p>El contenido está escrito en formato JSON y se puede convertir fácilmente en un objeto JavaScript.

.</p>

<p id="demo"></p>

<script>
```

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
       var miObj = JSON.parse(this.responseText);
       document.getElementById("demo").innerHTML =
miObj.nombre;
  }
};
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
</script>

<p>Echa un vistazo a <a href="json_demo.txt"
target="_blank">json_demo.txt</a></p>

</body>
</html>
```

El fichero json_demo.txt contiene los siguientes datos:

```
{
   "nombre":"Pedro",
   "edad":30,
   "mascotas":[
      { "animal":"perro", "nombre":"Fido" },
      { "animal":"gato", "nombre":"Felix" },
      { "animal":"hamster", "nombre":"Lightning" }
   ]
}
```

6. Función stringify

Un uso común de JSON es intercambiar datos hacia / desde un servidor web.

Al enviar datos a un servidor web, los datos deben ser una cadena.

Convierta un objeto JavaScript en una cadena con JSON.stringify().

6.1 Cadena de un objeto JavaScript

Imagina que tenemos este objeto en JavaScript:

```
var obj = { nombre: "Pedro", edad: 30, ciudad: "Valencia" };
```

En el siguiente código, utilizamos la función de JavaScript JSON.stringify()para convertirlo en una cadena.

```
var miJSON = JSON.stringify(obj);
```

El resultado será una cadena siguiendo la notación JSON

miJSON ahora es una cadena y está lista para enviarse a un servidor:

```
var obj = { nombre: "Pedro", edad: 30, ciudad: "Valencia" };
var miJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = miJSON ;
```

Página HTML:

```
<!DOCTYPE html>
<html>
<body>

<h2>Creación de una cadena JSON desde un objeto Javascript.</h2>

<p id="demo"></p>

<script>
var obj = { nombre: "Pedro", edad: 30, ciudad: "Valencia" };
var miJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = miJSON ;
</script>

</body>
```

```
</html>
```

Resultado desde el navegador:

Creación de una cadena JSON desde un objeto Javascript.

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

6.2 Cadena de una matriz de JavaScript

También es posible secuenciar matrices de JavaScript.

Imagina que tenemos esta matriz en JavaScript:

```
var array = [ "Pedro", "Juan", "Ana", "Luis" ];
```

Vamos a utilizar la función JavaScript JSON.stringify()para convertirlo en una cadena.

```
var miJSON = JSON.stringify(array);
```

El resultado será una cadena siguiendo la notación JSON

miJSON ahora es una cadena y está lista para enviarse a un servidor:

```
var miJSON = JSON.stringify(array);
document.getElementById("demo").innerHTML = miJSON;
```

7. Objetos JSON

Las características de los objetos JSON son:

- Los objetos JSON están rodeados por llaves {}.

- Los objetos JSON se escriben en pares clave / valor.
- Las claves deben ser cadenas y los valores deben ser un tipo de datos JSON válido (cadena, número, objeto, matriz, booleano o nulo).
- Las claves y los valores están separados por dos puntos.
- Cada par clave / valor está separado por una coma.

```
{ "nombre":"Pedro", "edad":30, "coche":null }
```

7.1 Acceder a los valores de los objetos

Puede acceder a los valores del objeto mediante la notación de puntos (.):

```
miObj = { "nombre":"Pedro", "edad":30, "coche":null };
x = miObj.nombre;
```

También puede acceder a los valores del objeto utilizando la notación entre corchetes ([]):

```
miObj = { "nombre":"Pedro", "edad":30, "coche":null };
x = miObj["nombre"];
```

7.2 Bucle de un objeto

Puede recorrer las propiedades del objeto utilizando el bucle for-in:

```
miObj = { "nombre":"Pedro", "edad":30, "coche":null };
for (dato in miObj) {
  document.getElementById("demo").innerHTML += dato;
}
```

Página HTML:

```
<!DOCTYPE html>
```

```
<html>
<body>

<p>Cómo recorrer todas las propiedades JSON con un bucle.</p>

<p id="demo"></p>

<script>
var miObj, dato;
miObj = { "nombre":"Pedro", "edad":30, "coche":null };
for (dato in miObj) {
  document.getElementById("demo").innerHTML += dato;
}

</script>

</body>
</html>
```

Resultado en el navegador:

Cómo recorrer todas las propiedades JSON con un bucle.

nombre

edad

coche

En un bucle for-in, use la notación de corchetes para acceder a los *valores de* propiedad :

```
miObj = { "nombre":"Pedro", "edad":30, "coche":null };
for (dato in miObj) {
  document.getElementById("demo").innerHTML += miObj[dato];
}
```

Página HTML:

```
<!DOCTYPE html>
<html>
<body>

<p>Cómo recorrer todas los valores JSON con un bucle.</p>

<p id="demo"></p>

<script>
var miObj, dato;
miObj = { "nombre":"Pedro", "edad":30, "coche":null };
for (dato in miObj) {
  document.getElementById("demo").innerHTML += miObj[dato];
}

</script>

</body>
</html>
```

Resultado en el navegador:

Cómo recorrer todos los valores JSON con un bucle

Pedro
30
null

7.3 Objetos JSON anidados

Los valores de un objeto JSON pueden ser otro objeto JSON.

```
miObj = {
  "nombre":"Pedro",
  "edad":30,
  "coches": {
    "coche1":"Ford",
    "coche2":"BMW",
```

```
  "coche3":"Fiat"
 }
}
```

Puede acceder a los objetos JSON anidados mediante la notación de puntos o la notación de corchetes:

```
x = miObj.coches.coche2;
// o:
x = miObj.coches["coche2"];
```

7.4 Modificar valores

Puede usar la notación de puntos para modificar cualquier valor en un objeto JSON:

```
miObj.coches.coche2 = "Mercedes";
```

También puede usar la notación entre corchetes para modificar un valor en un objeto JSON:

```
miObj.coches["coches2"] = "Mercedes";
```

7.5 Eliminar propiedades de objeto

Utilice la palabra **delete** clave para eliminar propiedades de un objeto JSON:

```
delete miObj.coches.coche2;
```

8. Matrices como objetos JSON

Las matrices en JSON son casi las mismas que las matrices en JavaScript.

En JSON, los valores de la matriz deben ser de tipo cadena, número, objeto, matriz, booleano o *nulo* .

En JavaScript, los valores de matriz pueden ser todos los anteriores, más cualquier otra expresión de JavaScript válida, incluidas funciones, fechas e *indefinidas.*

```
[ "Ford", "BMW", "Fiat" ]
```

8.1 Matrices en objetos JSON

Las matrices pueden ser valores de una propiedad de objeto:

```
{
"nombre":"Pedro",
"edad":30,
"coches":[ "Ford", "BMW", "Fiat" ]
}
```

8.2 Acceder a los valores de matriz

Puede acceder a los valores de la matriz utilizando el número de índice:

```
x = miObj.coches[0];
```

8.3 Bucle a través de una matriz

Puede acceder a los valores de la matriz utilizando un bucle **for-in** :

```
for (i in miObj.coches) {
  x += myObj.coches[i];
}
```

O puede usar un bucle for:

```
for (i = 0; i < miObj.coches.length; i++) {
```

```
  x += miObj.coches[i];
}
```

8.4 Matrices anidadas en objetos JSON

Los valores de una matriz también pueden ser otra matriz o incluso otro objeto JSON:

```
miObj = {
  "nombre":"Pedro",
  "edad":30,
  "coches": [
    { "nombre":"Ford", "modelos":[ "Fiesta", "Focus", "Mustang" ] },
    { "nombre":"BMW", "modelos":[ "320", "X3", "X5" ] },
    { "nombre":"Fiat", "modelos":[ "500", "Panda" ] }
  ]
 }
```

Para acceder a las matrices dentro de las matrices, use un bucle for-in para cada matriz:

```
for (i in miObj.coches) {
  x += "<h1>" + miObj.coches[i].nombre + "</h1>";
  for (j in miObj.coches[i].modelos) {
    x += miObj.coches[i].modelos[j];
  }
}
```

8.5 Modificar valores de matriz

Utilice el número de índice para modificar una matriz:

```
miObj.coches[1] = "Mercedes";
```

8.6 Eliminar elementos de matriz

Use la palabra **delete** para eliminar elementos de una matriz:

```
delete miObj.coches[1];
```

9. Bibliografía

XMLHttpRequest - Web APIs
W3 Schools JSON Introduction