APRENDERAPROGRAMAR.COM

¿QUÉ ES Y PARA QUÉ SIRVE JSON? ESPECIFICACIÓN OFICIAL DE JAVASCRIPT OBJECT NOTATION (CU01213F)

Sección: Cursos
Categoría: Tutorial básico del programador web: Ajax desde cero

Fecha revisión: 2031

Resumen: Entrega nº13 del Tutorial básico "Ajax desde cero".

Autor: Alex Rodríguez

EL FORMATO JSON

Ya sabemos que un aspecto principal de la tecnología Ajax es el intercambio de datos. Sabemos igualmente que podemos recuperar datos en formatos no estandarizados (por ejemplo datos contenidos en un fichero php) pero que resulta más ventajoso el uso de formatos estándar como XML. JSON es otro formato estándar para datos que destaca por ser ligero y rápido.

DE XML A JSON: PRIMERA APROXIMACIÓN

Al igual que XML define un formato para el intercambio de datos, JSON define otro formato. El formato JSON se basa en pares atributo-valor.

Un ejemplo de dato en formato JSON y su equivalente en XML puede ser el siguiente:

Formato JSON	Formato XML
{ "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 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>

En este ejemplo vemos algunas de las características de JSON. Un elemento de datos (unidad básica de intercambio de información u objeto JSON) queda delimitado por llaves { ... }. En este ejemplo las llaves externas delimitarían los datos de una persona.

Un elemento de datos JSON contiene pares de elementos "nombre": "valor" con la peculiaridad de que valor puede ser tanto una cadena de texto entrecomillada como otras cosas como explicaremos a continuación.

JSON: HISTORIA Y ESPECIFICACIÓN OFICIAL

A finales de los años 90 XML era el formato para intercambio de datos con mayor implantación. No obstante, presentaba problemas sobre todo cuando se trataba de trabajar con ficheros con gran volumen de datos donde el procesamiento se volvía lento.

Surgieron entonces intentos para definir formatos que fueran más ligeros y rápidos para el intercambio de información. Uno de ellos fue JSON, promovido y popularizado por Douglas Crockford y sus colaboradores a principios de los años 2000.

JSON se caracteriza por reducir el tamaño de los archivos y el volumen de datos que es necesario transmitir frente a otros estándares como XML. Por ello JSON fue adquiriendo popularidad hasta convertirse en un estándar. Esto no significa que XML haya dejado de utilizarse. En la actualidad se utiliza tanto XML como JSON para el intercambio de datos. Utilizar uno u otro depende de las circunstancias y de las preferencias que en cada momento se determinen.

Hoy día JSON cuenta con un sitio web oficial (http://json.org) y una especificación oficial producida por ECMA International, el mismo organismo encargado de la especificación oficial de JavaScript. La especificación oficial de JSON, publicada por primera vez en 2013 con el nombre de "Standard ECMA-404 The JSON Data Interchange Format" puede consultarse en la web de Ecma International (http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf).

Aunque en sus orígenes JSON estuvo ligado a JavaScript, y de hecho su notación puede decirse que está inspirada en la notación de objetos JavaScript, con el tiempo se ha convertido en un estándar independiente de datos, no ligado a ningún lenguaje en concreto, de la misma forma que xml no está ligado a ningún lenguaje en concreto.

Al igual que un fichero xml es un fichero de texto, un fichero json también lo es.

JSON: LENGUAJES CON SOPORTE

Con el tiempo prácticamente todos los lenguajes de programación han introducido soporte para el intercambio de datos basado en JSON. En este listado podemos ver algunos de los lenguajes o entornos con soporte para JSON y los nombres de sus librerías, extensiones o utilidades dedicadas a JSON:

ABAP: EPO Connector.
ActionScript: ActionScript3 , JSONConnector.
Ada: GNATCOLL , JSON.
AdvPL: JSON-ADVPL.
ASP: JSON for ASP , JSON ASP utility class.
AWK: JSON , awk , rhawk.
Bash: Jshon , JSON , sh.
BlitzMax: bmx-rjson.
C: JSON_checker , YAJL , js0n , LibU , json-c , json-parser , jsonsl , WJElement , M's JSON parser , cJSON , Jansson , jsmn , cson , parson , ujson4c , nxjson , frozen , microjson.
C++: JSONKit , jsonme-- , ThorsSerializer , JsonBox , jvar , rapidjson , jsoncons , json , JSON Support in Qt , QJson , qmjson , jsoncpp , zoolib , JOST , CAJUN , libjson , nosjob , JSON++ , SuperEasyJSON , Casablanca , JSON library for IoT , minijson.
C#: fastJSON , JSON_checker , Jayrock , Json , NET - LINQ to JSON , LitJSON , JSON for , NET , JsonFx , JSON@CodeTitans , How do I write my own parser? , JSONSharp , JsonExSerializer , fluent-json , Manatee Json , FastJsonParser
Ciao: Ciao JSON encoder and decoder.

Clojure: data , json.
Cobol: XML Thunder.
ColdFusion: SerializeJSON , toJSON.
D: Cashew , Libdjson
Dart: json library.
Delphi: Delphi Web Utils , JSON Delphi Library , tiny-json.
E: JSON in TermL.
Erlang: mochijson2.
Fantom: Json.
FileMaker: JSON.
Fortran: json-fortran , YAJL-Fort.
Go: package json.
Groovy: groovy-io
Haskell: RJson package , json package
Java: JSON-java , JSONUtil , Jackson JSON Processor , jsonp , Json-lib , JSON Tools , Stringtree , SOJO , Jettison , json-taglib , XStream , Flexjson , JON tools , Argo , jsonij , fastjson , mjson , jjson , json-simple , json-io , JsonMarshaller , google-gson , Json-smart , FOSS Nova JSON , Corn CONVERTER , Apache johnzon
JavaScript: JSON , json2 , js , clarinet , Oboe , js.
LabVIEW: i3-JSON , LAVA JSON ,
Lisp: Common Lisp JSON , Yason , Emacs Lisp.
LiveCode: mergJSON ,
LotusScript: JSON LS.
LPC: Grimoire: LPC JSON.
Lua: JSON Modules.
M: DataBallet.
Matlab: JSONlab , 20565 , 23393.
Net.Data: netdata-json.
Objective C: NSJSONSerialization , json-framework , JSONKit , yajl-objc , TouchJSON , ObjFW.
OCaml: Yojson , jsonm.
OpenLaszlo: JSON.
PascalScript: JsonParser.
Perl: CPAN , perl-JSON-SL.
Photoshop: JSON Photoshop Scripting
PHP: PHP 5 , 2 , json , Services_JSON , Zend_JSON , Comparison of php json libraries.
PicoLisp: picolisp-json.
Pike: Public , Parser , JSON , Public , Parser , JSON2.
PL/SQL: pljson: Librairie-JSON.
PowerShell: PowerShell.
Prolog: SWI-Prolog HTTP support.
Puredata: PuRestJson
Python: The Python Standard Library , simplejson , pyson , Yajl-Py , ultrajson , metamagic , json
R: rjson , jsonlite.
Racket: json-parsing.
Rebol: json , r.
RPG: JSON Utilities
Ruby: json , yajl-ruby , json-stream , yajl-ffi.
Scala: package json.
Scheme: MZScheme , PLT Scheme.
Squeak: Squeak.
Symbian: s60-json-library.
Tcl: JSON.
Visual Basic: VB-JSON , PW , JSON
Visual FoxPro: fwJSON , JSON , vfpjson.

JSON: ESPECIFICACIÓN FORMAL

La especificación formal de JSON es relativamente simple.

Gráficamente se define con las siguientes figuras.

Figura 1: un objeto JSON está formado por uno o varios pares string: value (cadena:valor)

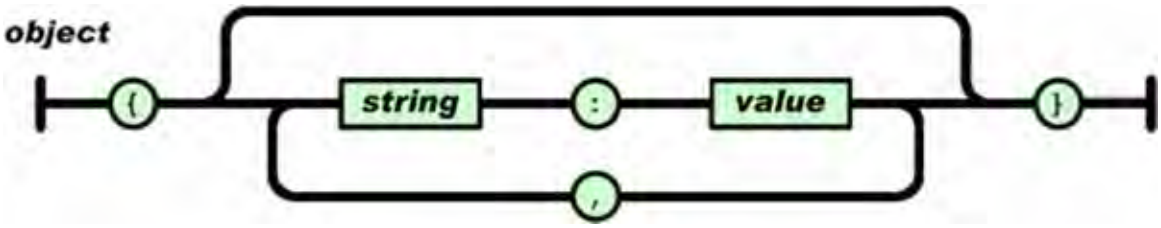


Figura 2: un valor en JSON puede ser un string (cadena), un número, un objeto JSON, un array, el valor true, el valor false o el valor null.

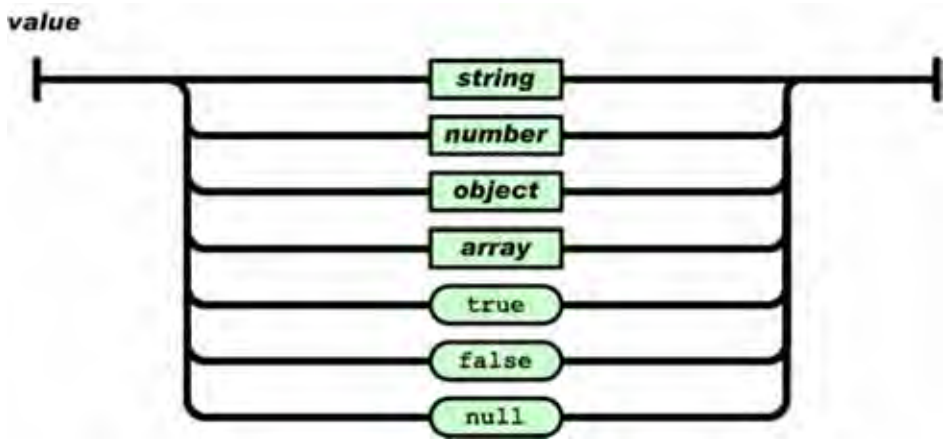


Figura 3: un array o arreglo en JSON está formado por valores delimitados entre corchetes y separados por comas.

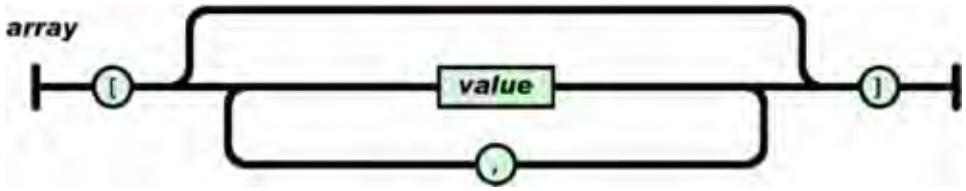


Figura 4: una cadena JSON queda formada de forma análoga a como se forma en muchos lenguajes de programación. Determinados caracteres han de usar una secuencia de escape.

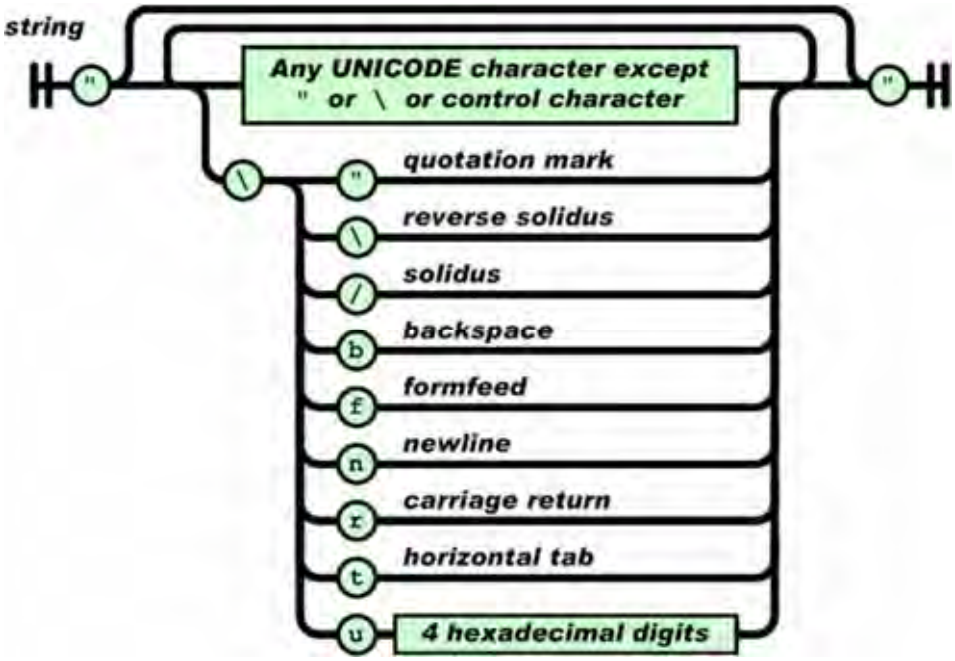
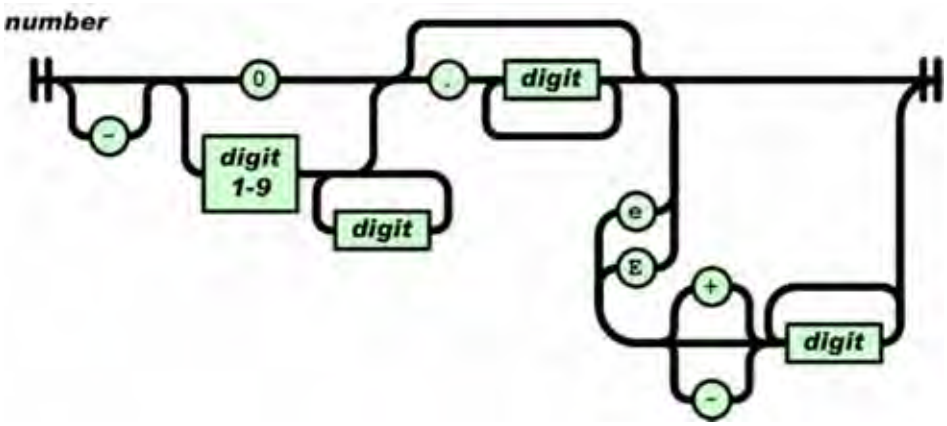


Figura 5: un número JSON queda formado de forma análoga a como se forma en muchos lenguajes de programación.



EJERCICIO

Transforma los datos en formato JSON mostrados a continuación en datos en formato XML y responde:

```
{
  "personas": [
    {
      "nombre": "Juan Pérez Galones",
      "altura": 1.72,
      "peso": 75,
      "pasatiempos": [
        "Comics",
        "Baloncesto",
        "Gaming"
      ],
      "soltero": true,
      "direccion": {
        "calle": "Avenida Soleares",
        "numero": "44",
        "pais": "Chile"
      }
    },
    {
      "nombre": "Pedro Motos Sandez",
      "altura": 1.69,
      "peso": 72,
      "pasatiempos": [
        "Programación",
        "Senderismo"
      ],
      "soltero": true,
      "direccion": {
        "calle": "Torero Pedro Cano",
        "numero": "78",
        "pais": "México"
      }
    }
  ]
}
```

a) Muestra el código XML equivalente.

b) Compara el número de caracteres que forma una codificación y otra. ¿Cuántos caracteres ocupa la codificación JSON? ¿Cuántos caracteres ocupa la codificación XML? (Nota: el número de caracteres se puede contar con un editor de texto).

c) Transforma la notación JSON para que toda la información quede en una sola línea. ¿Crees que el contenido en una sola línea es equivalente al contenido inicial? ¿Qué ventajas e inconvenientes le ves a tener toda la información en una sola línea?

d) Transforma la notación XML para que toda la información quede en una sola línea. ¿Qué línea resulta más larga, la línea con notación JSON o la línea con notación XML?

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01214F

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=83&Itemid=212