<http://zvon.org/xxl/XMLTutorial/General_spa/contents.html>

Un documento bien formado con un elemento :

|  |
| --- |
| <texto>Éste es un documento XML</texto> |

Un documento bien formado con varios elementos :

|  |
| --- |
| <texto>Éste es un  <doctipo>documento XML</doctipo>  </texto> |

El nombre de un elemento debe ser igual en las etiquetas inicial y final. En los nombres se distinguen mayúsculas y minúsculas.

<lista>  
  
<item>Coche</item>  
  
<ITEM>Avión</ITEM>  
  
<Item>Tren</Item>  
  
</lista>

Si la etiqueta inicial está dentro de un elemento, la etiqueta final también debe estar dentro del mismo elemento. Para decirlo más facil, los elementos, enmarcados por una etiqueta inicial y otra final, deben anidarse adecuadamente unos en otros

<libro>  
  
     <capitulo>  
  
          <titulo>Introducción</titulo>  
  
     </capitulo>  
  
     <capitulo>  
  
          <titulo>Historia</titulo>  
  
          <seccion>  
  
               <titulo>Parte 1</titulo>  
  
          </seccion>  
  
          <seccion>  
  
               <titulo>Parte 2</titulo>  
  
          </seccion>  
  
     </capitulo>  
  
     <capitulo>  
  
          <title>Indice</title>  
  
     </capitulo>  
  
</libro>

Todo elemento cuyo principio se marca con una etiqueta inicial, debe marcarse con una etiqueta final que repita el nombre del elemento, expresado igual que en la etiqueta inicial. El texto contenido entre la etiqueta inicial y la etiqueta final se llama contenido del elemento. Un elemento sin contenido puede tomar una forma especial <nombre/>. La barra situada antes de > sustituye la etiqueta final

<listaDeEtiquetas>  
  
     <AAA></AAA>  
  
     <BBB></BBB>  
  
     <CCC/>  
  
     <DDD/>  
  
</listaDeEtiquetas>

Los nombres de los elementos pueden incluir letras, números, guiones, guiones bajos, comas y puntos. La coma se puede usar sólo en un caso especial, cuando separa lo que llamamos namespace. Los nombres de elemento que comienzan con xml, XML u otras combinaciones de esta clase de cadena están reservadas para el standard.

<nombresPermitidos>  
  
     <nombre/>  
  
     <xsl:copy-of/>  
  
            <Un\_nombre\_de\_elemento\_largo/>  
  
            <Un.nombre.separado.con.puntos/>  
  
     <a123323123-231-231/>  
  
     <\_12/>  
  
</nombresPermitidos>

Ejemplo con datos:

**<?xml version="1.0" encoding="UTF-8"?>**

<root>

<list type="cdrom">

<item title="cd 1">Descripción de cd 1</item>

<item title="cd 2">Descripción de cd 2</item>

</list>

<list type="dvd">

<item title="dvd 1">Descripción de dvd 1</item>

</list>

**YAML**

default language: es

Significa, como es de suponer, la asignación del valor 'es' a la "variable" 'default language'.

other languages: [en, it]

Estaríamos definiendo una "variable" llamada 'other languages' que contendría un "array" con los valores "en" y "it"

default controller:

  web: home

  admin: login

En este caso "default controller" guardaría los indices "web" y "admin" con los valores"home" y "login"respectivamente...

EJEMPLO

<?php  
$addr = array(  
    "given" => "Chris",  
    "family"=> "Dumars",  
    "address"=> array(  
        "lines"=> "458 Walkman Dr.  
        Suite #292",  
        "city"=> "Royal Oak",  
        "state"=> "MI",  
        "postal"=> 48046,  
      ),  
  );  
$invoice = array (  
    "invoice"=> 34843,  
    "date"=> "2001-01-23",  
    "bill-to"=> $addr,  
    "ship-to"=> $addr,  
    "product"=> array(  
        array(  
            "sku"=> "BL394D",  
            "quantity"=> 4,  
            "description"=> "Basketball",  
            "price"=> 450,  
          ),  
        array(  
            "sku"=> "BL4438H",  
            "quantity"=> 1,  
            "description"=> "Super Hoop",  
            "price"=> 2392,  
          ),  
      ),  
    "tax"=> 251.42,  
    "total"=> 4443.52,  
    "comments"=> "Late afternoon is best. Backup contact is Nancy Billsmer @ 338-4338.",  
    );  
  
// genera un representación de la factura en YAML  
$yaml = yaml\_emit($invoice);  
var\_dump($yaml);  
  
// convierte lo anterior en YAML a una variable PHP  
$parsed = yaml\_parse($yaml);  
  
// verifica que la comprobación de ida y vuelta a una extructura equivalente  
var\_dump($parsed == $invoice);  
?>

El resultado del ejemplo sería algo similar a:

string(631) "---

invoice: 34843

date: "2001-01-23"

bill-to:

given: Chris

family: Dumars

address:

lines: |-

458 Walkman Dr.

Suite #292

city: Royal Oak

state: MI

postal: 48046

ship-to:

given: Chris

family: Dumars

address:

lines: |-

458 Walkman Dr.

Suite #292

city: Royal Oak

state: MI

postal: 48046

product:

- sku: BL394D

quantity: 4

description: Basketball

price: 450

- sku: BL4438H

quantity: 1

description: Super Hoop

price: 2392

tax: 251.420000

total: 4443.520000

comments: Late afternoon is best. Backup contact is Nancy Billsmer @ 338-4338.

...

"

bool(true)

Ejemplo:

cdrom:

item:

title: cd 1

description: Descripción de cd 1

item:

title: cd 2

description: Descripción de cd 2

dvd:

title: dvd 1

description: Descripción de dvd 1

**JSON:**

{"menu": {

"id": "file",

"value": "File",

"popup": {

"menuitem": [

{"value": "New", "onclick": "CreateNewDoc()"},

{"value": "Open", "onclick": "OpenDoc()"},

{"value": "Close", "onclick": "CloseDoc()"}

]

}

}

}

XML:

**<menu** id="file" value="File"**>**

**<popup>**

**<menuitem** value="New" onclick="CreateNewDoc()" **/>**

**<menuitem** value="Open" onclick="OpenDoc()" **/>**

**<menuitem** value="Close" onclick="CloseDoc()" **/>**

**</popup>**

**</menu>**

Por ejemplo, una API pública de servicio Web devuelve el siguiente objeto JSON, que describe información sobre un usuario del servicio.

{"personal": {"name": "Paul", "age": 23, "height": 1.7, "isSingle": true, "luckyNumbers": [5,17,21]}, "favoriteBands": ["Band ABC", "Band XYZ"]}

{"widget": {

"debug": "on",

"window": {

"title": "Sample Konfabulator Widget",  
 "name": "main\_window",  
 "width": 500,  
 "height": 500

},  
 "image": {

"src": "Images/Sun.png",

"name": "sun1",  
 "hOffset": 250,  
 "vOffset": 250,  
 "alignment": "center"

},  
 "text": {

"data": "Click Here",

"size": 36,

"style": "bold",  
 "name": "text1",  
 "hOffset": 250,  
 "vOffset": 100,  
 "alignment": "center",

"onMouseUp": "sun1.opacity = (sun1.opacity / 100) \* 90;"

}

}}

The same text expressed as [XML](http://www.konfabulator.com/workshop/Konfabulator_Ref_1.8.pdf):

<widget>  
 <debug>on</debug>  
 <window title="Sample Konfabulator Widget">  
 <name>main\_window</name>  
 <width>500</width>  
 <height>500</height>  
 </window>  
 <image src="Images/Sun.png" name="sun1">  
 <hOffset>250</hOffset>  
 <vOffset>250</vOffset>  
 <alignment>center</alignment>  
 </image>  
 <text data="Click Here" size="36" style="bold">  
 <name>text1</name>  
 <hOffset>250</hOffset>  
 <vOffset>100</vOffset>  
 <alignment>center</alignment>  
 <onMouseUp>  
 sun1.opacity = (sun1.opacity / 100) \* 90;  
 </onMouseUp>  
 </text>  
</widget>

Anatomía de un objeto JSON

La anatomía de un marcado JSON es casi idéntica a la de un objeto Javascript. Veamos un ejemplo:

|  |
| --- |
| {  "id" : "0001",  "type" : "donut",  "name" : "Cake",  "image" : {  "url" : "images/0001.jpg",  "width" : 200,  "height" : 200  },  "thumbnail" : {  "url" : "images/thumbnails/0001.jpg",  "width" : 32,  "height" : 32  },  "dateEntry" : "2010-12-05"  } |

Como podemos observar, este ejemplo es prácticamente idéntico a la definición de un objeto en Javascript: todo el marcado queda enmarcado entre dos llaves dentro de las cuales, se definen parámetros siguiendo el esquema de pares *‘nombre-valor’*. Cada uno de los términos va invariablemente delimitado por comillas, algo que en Javascript no es siempre necesario pero si recomendable.

Particularidades de JSON sobre Javascript

Algunas de las particularidades o reglas del formato JSON a tener en cuenta son:

* Los pares nombre-valor van siempre **delimitados por comillas**, independientemente de si se tratan de nombres válidos en Javascript que podrían aparecer sin ellas.
* JSON puede representar seis tipos de valores: **objetos, arrays, numeros, cadenas, booleanos y null.**
* Las fechas no son reconocidas como un tipo de objeto propio.
* Los numeros en JSON no pueden ir predecidos de ceros salvo en el caso de notación decimal *( Ejm: 0.001 )*.