

# Transferencia datos con XML

---





# Que es XML?

- XML es un esqueleto para crear lenguajes con delimitadores
- XML significa EXtensible Markup Language
- XML ha sido creado para transportar y almacenar los datos, el HTML para visualizarlos.
  - La sintaxis del XML es muy parecida al los tag del HTML:  
`<element attribute="value">content</element>`
  - Los tags del XML no son pre-definidos. Se pueden definir tag relacionados con la aplicacion
  - Los tags del HTML/XML permiten de definir objetos anidados
- El XML permite representar estructuras de datos complejos en una forma fácil a leer y “auto descriptiva”



# Estructura XML

```
<?xml version="1.0" encoding="UTF-8"?>  <!-- XML promy_text -->
<note private="true">  <!-- root element -->
  <from>Alice Smith (alice@example.com)</from>
  <to>Robert Jones (roberto@example.com)</to>
  <subject>Tomorrow's "Birthday Bash" event!</subject>
  <message language="english">
    Hey Bob, Don't forget to call me this weekend!
  </message>
</note>
```

- Empieza con un tag `<?xml ... ?>` (**promy\_text**)
- Tiene un solo **root element** (en este caso note)
- Los tag, atributos, y comentarios se parecen al HTML pero el HTML solo puede usar (like `<p>`, `<h1>`, etc.).



# Quien usa XML?

- Los datos XML proceden de muchas fuentes del web:
  - Los servidores web almacenan los datos como ficheros XML
  - Las bases de datos a veces devuelven los resultados en XML
  - Los web services usan XML para comunicar
- El XML es el formato universal para intercambiar datos
- Los lenguajes basados en XML se usan para música, matemática, grafica...etc...
- El uso mas popular son los RSS para los news y los podcast



# Tag disponibles

- Cualquier tag puede ser usado ejemplos:
  - Una biblioteca puede usar tag como libro, titulo, autor
  - Una canción puede usar tags como llave, tono, nota
- Cuando se definen los datos XML se tiene que elegir la mejor representación.
  - Las informaciones mas largas y compleja se transforma en tags
  - Los detalles mas pequeños y los meta-data con tipos simples (integer, string, boolean) se transforman en atributos



# Cuando se usa?

- Permite la lectura y el envío de múltiples datos al mismo tiempo.
- Seria muy complicado analizar un string largo que contenga muchos datos concatenados.
- Los datos del XML son
  - separados y identificados,
  - procesados y visualizados.
  - El envio de datos XML se efectua en general usando Ajax (la “x” en Ajax corresponde a XML)
- Los datos XML se pueden tratar con un ***parser*** y separar en una serie de objetos como se hace por el DOM.
  - Existe un DOM del XML, parecido al DOM del HTML



# Otros usos

- **El XML permite hacer dinamico el HTML**
  - Si el contenido de la pagina es dinámico (puede cambiar), se tarda mucho en re-escribir el HTML cada vez.
  - Con un XML, los datos pueden ser almacenados en un fichero XML separado. De esta forma solo se tiene que definir los estilos del CSS y el layout de los párrafos principales.
  - Con pocas líneas de JS se puede leer un fichero externo XML y actualizar la pagina.
- **XML Simplifica la forma de compartir informacion**
  - Las bases de datos tienen que compartir informaciones de formato distinto.
  - Los datos del XML se almacenan en un fichero de texto “plain text” (sin codifica, fácil a abrir y leer).
- **XML Simplifica el transporte de datos**
  - El XML permite el intercambio de datos entre sistemas diferentes (Windows-Linux-Mac... Desktop-Tablet-Telephone) y usando Internet.
  - Los mismos datos pueden ser leídos por diferentes aplicaciones que son en general incompatibles.



# Estructura de un XML

```
<bookstore>
  <book category="CHILDREN">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

En el ejemplo arriba:

- `<bookstore>` and `<book>` se llaman **elementos**, porque contienen otros elementos
- `<book>` tiene un **atributo** (`category="CHILDREN"`).
- `<title>`, `<author>`, `<year>`, and `<price>` son de contenido **texto** porque contienen strings.





# XML Elementos vs. Atributos

1. 

```
<person sex="female">  
  <firstname>Anna</firstname>  
  <lastname>Smith</lastname>  
</person>
```
  2. 

```
<person>  
  <sex>female</sex>  
  <firstname>Anna</firstname>  
  <lastname>Smith</lastname>  
</person>
```
- En el primer ejemplo “sex” es un atributo, mientras que en el segundo es un elemento. En los dos casos proporciona la misma información.
    - los atributos proporcionas información adicional con respecto a los elementos
      - ```
<file type="gif">computer.gif</file>
```



# Reglas para escribir un XML

- **Todos los elementos XML deben tener un tag de fin.**  
MAL `<p>This is a paragraph.  
<br>`  
BIEN! `<p>This is a paragraph.</p>  
<br />`
- **Los tags del XML son Case Sensitive**
  - `<Letter>` es distinto de `<letter>`.
  - El primer y ultimo tag deben ser escrito de la misma forma:  
`<Message>This is incorrect</message>`  
`<message>This is correct</message>`
- **En XML los elementos tienen que ser correctamente anidados.**  
MAL `<b><i>This text is bold and italic</b></i>`  
BIEN `<b><i>This text is bold and italic</i></b>`
- **Los elementos XML deben contener un elemento Root**
  - `<root>  
  <child>  
    <subchild>.....</subchild>  
  </child>  
</root>`
- **Los atributos deben ser “Quoted”**  
MAL `<note date=12/11/2007>`  
BIEN `<note date="12/11/2007">`



# Reglas para escribir un XML

- **Caracteres especiales**

- Si se pone el caracter "<" en un XML el parser lo interpreta como si fuera el principio de un nuevo tag.

MAL `<message>if salary < 1000 then</message>`

BIEN `<message>if salary &lt; 1000 then</message>`

- Usar

|                         |                    |                |
|-------------------------|--------------------|----------------|
| <code>&amp;lt;</code>   | <code>&lt;</code>  | less than      |
| <code>&amp;gt;</code>   | <code>&gt;</code>  | greater than   |
| <code>&amp;amp;</code>  | <code>&amp;</code> | ampersand      |
| <code>&amp;apos;</code> | <code>'</code>     | apostrophe     |
| <code>&amp;quot;</code> | <code>"</code>     | quotation mark |

- **Commentarios**

`<!-- This is a comment -->`

- **Nombres**

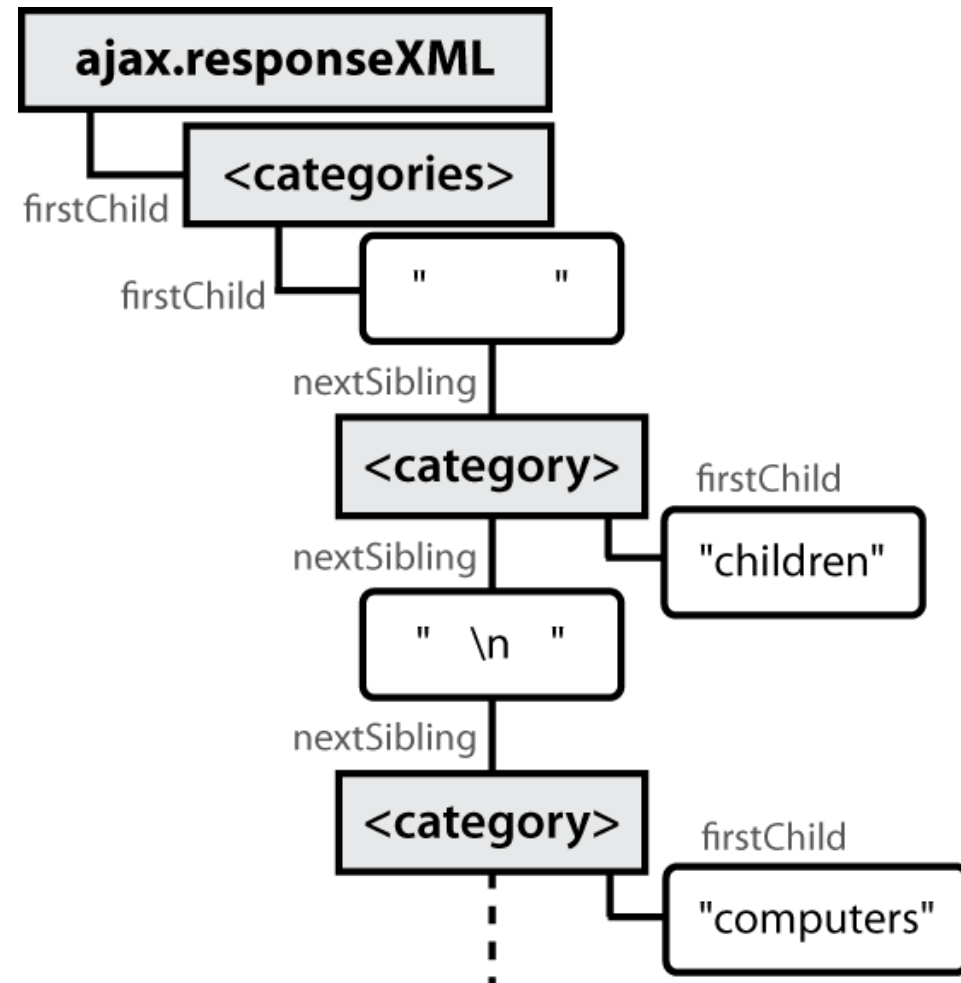
- Los nombres pueden contener letras, numeros y otros caracteres.
- Los nombres no pueden empezar con un numero o un caracter de puntuacion.
- Los nombres no pueden empezar con las letras xml (or XML, or Xml, etc)
- Los nombres no pueden contener espacios



# Acceder al DOM-XML

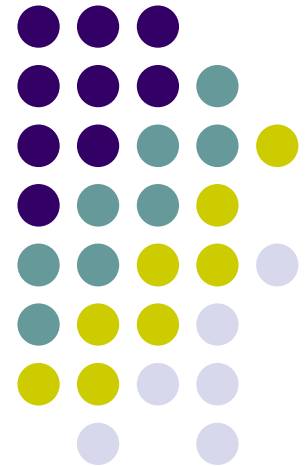
- Los tags XML tienen una estructura de albero como el DOM de una pagina web
- Los nodos del DOM tienen padres, hijos, y hermanos
- Cada nodo del DOM tiene propiedades/ métodos para acceder a los nodos vecinos

```
<?xml version="1.0" encoding="UTF-8"?>
<categories>
  ...
  <category>children</category>
  <category>computers</category>
  ...
</categories>
```



# Servidor

Creación de un XML con PHP





# XML : DOMDocument

- La clase de PHP [DOMDocument](#) representa un documento XML. Contiene los siguientes métodos:
  1. [createElement\(tag\)](#) crea un nuevo elemento que se puede añadir al documento
  2. [createTextNode\(text\)](#) crea un nuevo nodo de texto para que sea anadido al documento
  3. [getElementById\(id\)](#), [getElementsByTagName\(tag\)](#) busca un elemento en el documento
  4. [load\(filename\)](#), [loadXML\(string\)](#) lee los datos XML desde un fichero en el disco o desde un string
  5. [save\(filename\)](#), [saveXML\(\)](#) escribe datos XML en un fichero o lo devuelve como string
  6. [validate\(\)](#) controla si el documento contiene en datos XML validos



# XML : DOMElement

● La clase de PHP [DOMElement](#) representa cada elemento DOM, Contiene los siguientes métodos:

- |                                                                                             |                                                               |
|---------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| 1. <code>tagName, nodeValue</code>                                                          | 1. El nombre de un nodo (tag) y su valor (text)               |
| 2. <code>parentNode, childNodes, firstChild, lastChild, previousSibling, nextSibling</code> | 2. Referencia los nodos vecinos                               |
| 3. <code>appendChild(DOMNode), insertBefore(newNode, oldNode), removeChild(DOMNode)</code>  | 3. Modifica la lista de nodos hijos                           |
| 4. <code>getElementsByTagName(tag)</code>                                                   | 4. busca elementos descendiente del elemento indicado         |
| 5. <code>getAttribute(name), setAttribute(name, value), removeAttribute(name)</code>        | 5. Recupera/define el valor de un atributo a partir de su tag |

# Crear un XML usando DOMDocument



```
$xmldoc = new DOMDocument();
$books_tag = $xmldoc->createElement("books");
$xmlDoc->appendChild($books_tag);
foreach ($books as $book) {
    $book_tag = $xmldoc->createElement("book");
    $book_tag->setAttribute("title", $book["title"]);
    $book_tag->setAttribute("author", $book["author"]);
    $books_tag->appendChild($book_tag);
}
print $xmldoc->saveXML();
```

- Es mas facil leer/modificar XML complejos
- `saveXML` automaticamente anade el prolog XML

```
# <?xml version="1.0"?>
```

```
# <books>
```

```
$book_tag = $xmldoc->createElement("book"); # <book
```

```
$book_tag->setAttribute("title", $book["title"]); # title="Harry Potter" />
```

```
$book_tag->setAttribute("author", $book["author"]); # author="J.K. Rowling" />
```

```
$books_tag->appendChild($book_tag);
```

```
# </books>
```

```
<?xml version="1.0"?>
```

```
<books>
```

```
<book title="Harry Potter" />
```

```
<book author="J.K. Rowling" />
```

```
</books>
```

Si se usan caracteres españoles:

```
$xmldoc = new DOMDocument('1.0', 'UTF-8');
```



# Cliente

Cómo interactuar con los nodos  
XML en JS?



# Cómo interactuar con los nodos XML en JS

- Para recuperar un array de nodos

```
var elms = node.getElementsByTagName("tag");
```

```
var elms = node.querySelectorAll("selector"); // all elements
```

```
var elm = node.querySelector("selector"); // first element
```

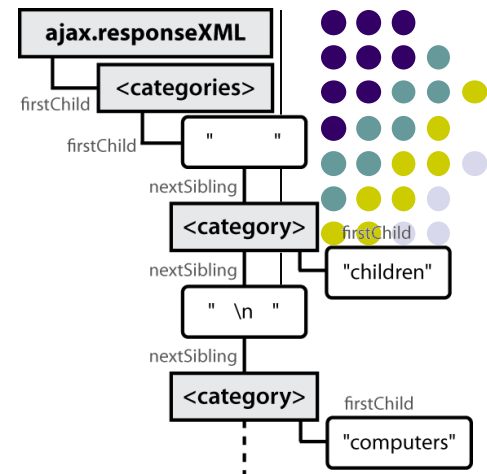
- Para recuperar el texto dentro un nodo

```
var text = node.textContent; // or,
```

```
var text = node.firstChild.nodeValue;
```

- Para recuperar una valor de un atributo de un nodo

```
var attrValue = node.getAttribute("name");
```



# Lista de propiedades de XML DOM



- Propiedades:
  - nodeName, nodeType, nodeValue, attributes
  - firstChild, lastChild, childNodes, nextSibling, previousSibling, parentNode
- metodos:
  - getElementByTagName, getAttribute, hasAttribute[s], hasChildNodes
  - appendChild, insertBefore, removeChild, replaceChild
- No se pueden usar los metodos de Prototype/jQuery como
  - up, down, ancestors, childElements, or siblings

# Diferencias en la sintaxis de JS cuando se manipula un XML

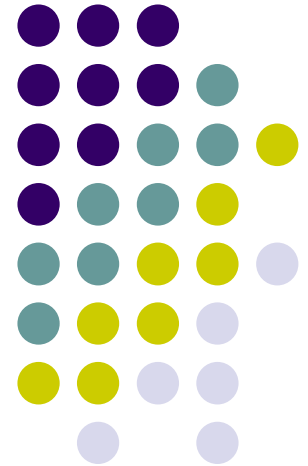


- No es posible recuperar los id de los nodos usando `$`
  - NO `var elms = $("id");`
- No se puede recuperar el mismo texto dentro un nodo como cuando se usa `innerHTML`
  - NO `var text = $("foo").innerHTML;`
- No se puede recuperar un valor de un atributo usando `.attributeName`
  - NO `var text = $("foo").innerHTML;`

# Parsear un XML para crear elementos de una pagina web

---

Usando prototype



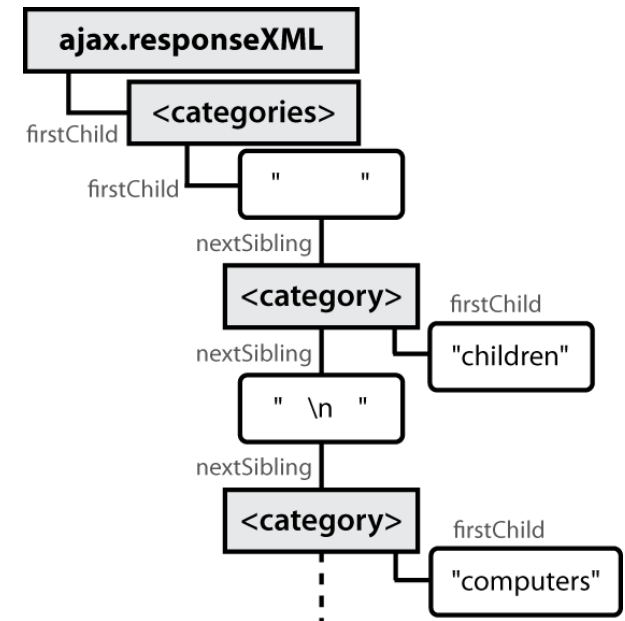
# Como parsear el contenido de un XML



- Imaginamos de recibir un XML por AJAX. Usamos prototype como librería JS

```
new Ajax.Request("url",
{
    method: "get",
    onSuccess: functionName
});
...
```

```
function functionName ajax {
    do something with ajax.responseXML;
}
```



- ajax.responseXML es el objeto DOM que contiene el arbol XML

# Como parsear el contenido de un XML (ejemplo 1)



```
<?xml version="1.0" encoding="UTF-8"?>
<employees>
  <lawyer money="99999.00" />
  <employee name="Ed">
    <job_position model="director" />  </employee>
  <employee name="Bill">
    <job_position model="programmer" />
  </employee>
</employees>
```

//cuanto gana un "lawyer"?

```
var lawyer = ajax.responseXML.getElementsByTagName("lawyer")[0];
var salary = lawyer.getAttribute("money"); // "99999.00"
```

// array de 2 employees

```
var employees = ajax.responseXML.getElementsByTagName("employee");
var employee_position =
  employees[0].getElementsByTagName("job_position")[0].getAttribute("model");
// "director"
```

# Como parsear el contenido de un XML (ejemplo 2)



- Ejemplo de fichero XML mas complejo:

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year><price>30.00</price>
  </book>
  <book category="computers">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <year>2003</year><price>49.99</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year><price>29.99</price>
  </book>
  <book category="computers">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year><price>39.95</price>
  </book>
</bookstore>
```



# Como parsear el contenido de un XML(ejemplo 2)



```
// creamos un párrafo para cada libro de la categoría "computers"
```

```
var books = ajax.responseXML.getElementsByTagName("book");
```

```
for (var i = 0; i < books.length; i++) {
```

```
    var category = books[i].getAttribute("category");
```

```
    if (category == "computers") {
```

```
// extraemos los datos desde el XML
```

```
    var title =
```

```
    books[i].getElementsByTagName("title")[0].firstChild.nodeValue;
```

```
    var author =
```

```
    books[i].getElementsByTagName("author")[0].firstChild.nodeValue;
```

```
// el fichero XML se transforma en código HTML
```

```
// modificamos el DOM del HTML añadiendo tags <p> que contengan el XML
```

```
    var p = document.createElement("p");
```

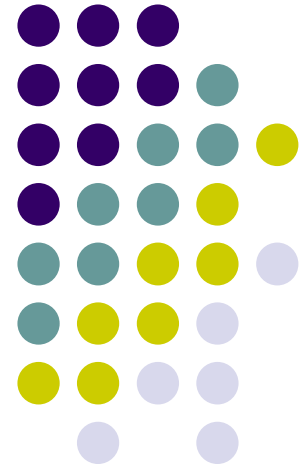
```
    p.innerHTML = title + ", by " + author;
```

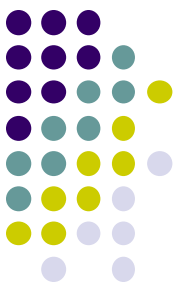
```
    document.body.appendChild(p);
```

# Parsear un XML para crear elementos de una pagina web

---

Usando jQuery





# jQuery – Ajax (ejemplo 1)

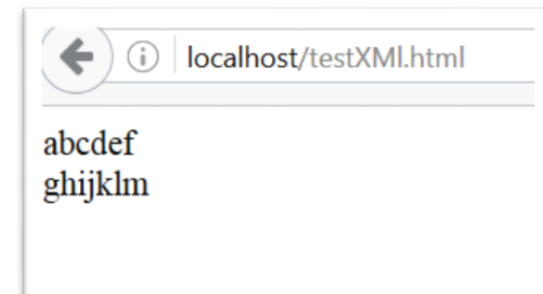
```
<!DOCTYPE html>
<html>
<head>
<script
  src="http://ajax.googleapis.com
  /ajax/libs/jquery/1.4.2/jquery.mi
  n.js"></script>
</head>
<body>

<div id="output"></div>
</script>
----->
</script>
</body>
</html>
```

```
<script type="text/javascript">
$(document).ready(function){
  $.ajax({
    type: "GET",
    dataType: "xml",
    url: "example.xml",
    success: function(xml){
      $(xml).find("book").each(function(){
        $("#output").append($(this).attr("code") + "<br />");
      }); //end (XML).FIND
    } //end FUNCTION(XML)
  }); //end AJAX
}; //END DOC.READY
</script>
```

example.xml

```
<?xml version="1.0"?>
<books title="A list of books">
  <book code="abcdef" />
  <book code="ghijklm">
    Some text contents
  </book>
</books>
```



- Inicialmente solo hay 1 elemento en el DOM (output)
- En caso que la comunicación AJAX tenga éxito (on success)
  - Añado al elemento output nuevos nodos ("#output").append
  - El nodo que añado es el texto (xml).find("book")

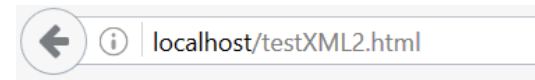


# jQuery – Ajax (ejemplo 2)

```
$(document).ready(function(){
    $("#dvContent").append("<ul></ul>");
    $.ajax({
        type: "GET",
        url: "BookList.xml",
        dataType: "xml",
        success: function(xml)
        {
            $(xml).find('Book').each(function(){
                var sTitle = $(this).find('Title').text();
                var sPublisher = $(this).find('Publisher').text();
                $("<li></li>").html(sTitle + ", "
                +sPublisher).appendTo("#dvContent ul");
            });
        }
    });
});
```

## Booklist.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<BookList>
  <Book>
    <Title>jQuery: Novice to Ninja</Title>
    <Publisher>Site point</Publisher>
  </Book>
  <Book>
    <Title>Learning jQuery</Title>
    <Publisher>PACKT</Publisher>
  </Book>
</BookList>
```



This is the context of the XML file:

- jQuery: Novice to Ninja, Site point
- Learning jQuery, PACKT
- Head First jQuery, O'Reilly
- jQuery UI 1.8, PACKT

# jQuery – Ajax (ejemplo 2)



## El código completo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
```

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
```

```
<script type="text/javascript" language="javascript">
```

```
$(document).ready(function() {  
    $("#divContent").append("<ul></ul>");  
    $.ajax({  
        type: "GET",  
        url: "BookList.xml",  
        dataType: "xml",  
        success: function(xml) {  
            $(xml).find('Book').each(function() {  
                var sTitle = $(this).find('Title').text();  
                var sPublisher = $(this).find('Publisher').text();  
                $("#<li></li>").html(sTitle + ", " + sPublisher).appendTo("#divContent ul");  
            });  
        },  
        error: function() {  
            alert("An error occurred while processing XML file.");  
        }  
    });  
});
```

```
</script>
```

```
<style type="text/css">
```

```
body  
{  
    font-family : Arial;  
    font-size : 10pt;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<form id="form1" runat="server">
```

```
<div id="divContent">This is the context of the XML file:
```

```
</div>
```

```
</form>
```

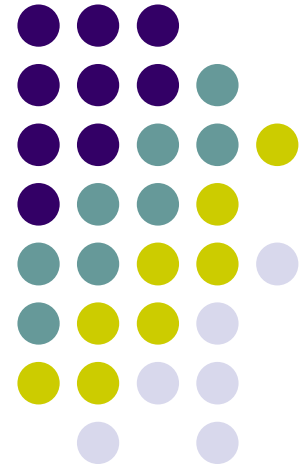
<http://stackoverflow.com/questions/30637288/how-to-read-parse-and-display-an-xml-using-jquery-and-ajax>

Copyright © 2016 by Balboia Simoes. All rights reserved.

# Parsear un XML para crear elementos de una pagina web

---

Sin librerias





# AJAX con XML (ejemplo 2)

## CLIENTE

```
<!DOCTYPE html> <html><body>
<div>
<b>To:</b> <span id="to"></span><br>
<b>From:</b> <span id="from"></span><br>
<b>Message:</b> <span id="message"></span>
</div>
```

```
<script>
xmlhttp=new XMLHttpRequest();
xmlhttp.open("GET","note.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;
```

```
document.getElementById("to").innerHTML=
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML=
xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
</script>
</body></html>
```

### note.xml

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

## W3Schools Internal Note

**To:** Tove  
**From:** Jani  
**Message:** Don't forget me this weekend!

# AJAX con XML (ejemplo 2)



## CLIENTE

```
<html>
  <body>
    <script type="text/javascript">
      xmlhttp=new XMLHttpRequest();
      xmlhttp.open("GET","cd_catalog.xml",true);
      xmlhttp.send();
      xmlDoc=xmlhttp.responseXML;

      document.write("<table border='1'>");
      var x=xmlDoc.getElementsByTagName("CD");
      for (i=0;i<x.length;i++)
      {
        document.write("<tr><td>");
        document.write(x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue);
        document.write("</td><td>");
        document.write(x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue);
        document.write("</td></tr>");
      }
      document.write("</table>");
    </script>
  </body>
</html>
```

## "cd\_catamy\_text.xml"

```
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE> <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD> ...
</CATALOG>
```

Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros
Bee Gees	One night only
Dr.Hook	Sylvias Mother
Rod Stewart	Maggie May
Andrea Bocelli	Romanza
Percy Sledge	When a man loves a
Savage Rose	Black angel
Mary	1999 Century Ne



# AJAX con XML

---

Cómo enviar con PHP  
y recibir con JS





# AJAX con XML (ejemplo 1)

- [http://www.w3schools.com/php/php\\_ajax\\_xml.asp](http://www.w3schools.com/php/php_ajax_xml.asp)

Bob Dylan ▼

Select a CD:

Bob DylanBurlesque

Bee GeesDylan

Cat StevensA

**COMPANY:** Columbia

**PRICE:** 10.90

**YEAR:** 1985

# AJAX con XML (ejemplo 1)



cd\_catalog.xml

```
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
  <CD>
    <TITLE>Greatest Hits</TITLE>
    <ARTIST>Dolly Parton</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>RCA</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1982</YEAR>
  </CD>
```

# Cliente



```
<html><head>
<script>
function showCD(str)
{
xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 &&
xmlhttp.status==200)
{
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET","getcd.php?q="+str,true);
xmlhttp.send();
}
</script>
```

```
</head><body>

<form>
Select a CD:
<select name="cds"
onchange="showCD(this.value)">
<option value="">Select a CD:</option>
<option value="Bob Dylan">Bob
Dylan</option>
<option value="Bonnie Tyler">Bonnie
Tyler</option>
<option value="Dolly Parton">Dolly
Parton</option>
</select>
</form>
<div id="txtHint"><b>CD info will be listed
here...</b></div>

</body></html>
```

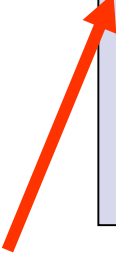
# Servidor

```
<?php
$q=$_GET["q"];

$xmlDoc = new DOMDocument();
$xmlDoc->load("cd_catalog.xml");

$x=$xmlDoc-
    >getElementsByTagName('ARTIST');

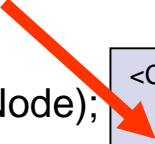
for ($i=0; $i<=$x->length-1; $i++)
{
    //Process only element nodes
    if ($x->item($i)->nodeType==1)
    {
        if ($x->item($i)->childNodes->item(0)-
            >nodeValue == $q)
        {
            $y=($x->item($i)->parentNode);
        } } }
```



```
<CD>
<TITLE>Empire Burlesque</TITLE>
<ARTIST>Bob Dylan</ARTIST>
<COUNTRY>USA</COUNTRY>
<COMPANY>Columbia</COMPANY>
<PRICE>10.90</PRICE>
<YEAR>1985</YEAR>
</CD>
```

```
$cd=($y->childNodes);

for ($i=0;$i<$cd->length;$i++)
{
    //Process only element nodes
    if ($cd->item($i)->nodeType==1)
    {
        echo("<b>" . $cd->item($i)->nodeName .
            ":",</b> ");
        echo($cd->item($i)->childNodes->item(0)-
            >nodeValue);
        echo("<br>");
    }
}
?>
```



```
<CD>
<TITLE>Empire Burlesque</TITLE>
<ARTIST>Bob Dylan</ARTIST>
<COUNTRY>USA</COUNTRY>
<COMPANY>Columbia</COMPANY>
<PRICE>10.90</PRICE>
<YEAR>1985</YEAR>
</CD>
```

