

SimpleXML is good for parsing existing XML documents, but you can't use it to create a new one from scratch.

The easiest way to generate an XML document is to build a PHP array whose structure mirrors that of the XML document and then to iterate through the array, printing each element with appropriate formatting.

Example

Try out following example:

```
<?php

$channel = array('title' => "What's For Dinner",
                'link' => 'http://menu.example.com/',
                'description' => 'Choose what to eat tonight.');
```

```
print "<channel>\n";
foreach ($channel as $element => $content) {
    print " <$element>";
    print htmlentities($content);
    print "</$element>\n";
}
print "</channel>";
?>
```

It will produce following result:

```
<channel>
<title>What's For Dinner</title>
<link>http://menu.example.com/</link>
<description>Choose what to eat tonight.</description>
</channel></html>
```

DOM

A DOM (Document Object Model) defines a standard way for accessing and manipulating documents.

The XML DOM defines a standard way for accessing and manipulating XML documents.

The XML DOM views an XML document as a tree-structure.

All elements can be accessed through the DOM tree. Their content (text and attributes) can be modified or deleted, and new elements can be created. The elements, their text, and their attributes are all known as nodes.

You can learn more about the XML DOM in our [XML DOM tutorial](#).

The HTML DOM

The HTML DOM defines a standard way for accessing and manipulating HTML documents.

All HTML elements can be accessed through the HTML DOM.

You can learn more about the HTML DOM in our [JavaScript tutorial](#).

Load an XML File - Cross-browser Example

The following example parses an XML document ("[note.xml](#)") into an XML DOM object and then extracts some info from it with a JavaScript:

Example

```
<html>
<body>
<h1>W3Schools Internal Note</h1>
<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>
if (window.XMLHttpRequest)
  { // code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  { // code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
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>
```

Load an XML String - Cross-browser Example

The following example parses an XML string into an XML DOM object and then extracts some info from it with a JavaScript:

Example

```
<html>  
<body>  
<h1>W3Schools Internal Note</h1>  
<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>  
txt="<note>";  
txt=txt+"<to>Tove</to>";  
txt=txt+"<from>Jani</from>";  
txt=txt+"<heading>Reminder</heading>";  
txt=txt+"<body>Don't forget me this weekend!</body>";  
txt=txt+"</note>";  
  
if (window.DOMParser)  
{  
  parser=new DOMParser();  
  xmlDoc=parser.parseFromString(txt,"text/xml");  
}  
else // Internet Explorer  
{  
  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
```

```
xmlDoc.async=false;
xmlDoc.loadXML(txt);
}
```

```
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>
```

Document Type Definition

An XML document with correct syntax is called "Well Formed".

An XML document validated against a DTD is "Well Formed" and "Valid".

Valid XML Documents

A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The DOCTYPE declaration, in the example above, is a reference to an external DTD file. The content of the file is shown in the paragraph below.

XML DTD

The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements:

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
```

```

<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>

```

The DTD above is interpreted like this:

- !DOCTYPE note defines that the root element of the document is note
- !ELEMENT note defines that the note element must contain four elements: "to, from, heading, body"
- !ELEMENT to defines the to element to be of type "#PCDATA"
- !ELEMENT from defines the from element to be of type "#PCDATA"
- !ELEMENT heading defines the heading element to be of type "#PCDATA"
- !ELEMENT body defines the body element to be of type "#PCDATA"

Using DTD for Entity Declaration

A doctype declaration can also be used to define special characters and character strings, used in the document:

Example

```

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

<!DOCTYPE note [
<!ENTITY nbsp "&#xA0;">
<!ENTITY writer "Writer: Donald Duck.">
<!ENTITY copyright "Copyright: W3Schools.">
]>

<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
<footer>&writer;&nbsp;&copyright;</footer>
</note>

```

Why Use a DTD?

With a DTD, independent groups of people can agree on a standard for interchanging data.

With a DTD, you can verify that the data you receive from the outside world is valid.

Displaying XML with XSLT

XSLT (eXtensible Stylesheet Language Transformations) is the recommended style sheet language for XML.

XSLT is far more sophisticated than CSS. With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.

XSLT uses XPath to find information in an XML document.

XSLT Example

We will use the following XML document:

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

  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
    <calories>650</calories>
  </food>

  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>Light Belgian waffles covered with strawberries and whipped cream</description>
    <calories>900</calories>
  </food>

  <food>
    <name>Berry-Berry Belgian Waffles</name>
    <price>$8.95</price>
    <description>Light Belgian waffles covered with an assortment of fresh berries and whipped cream</description>
    <calories>900</calories>
  </food>

  <food>
    <name>French Toast</name>
    <price>$4.50</price>
    <description>Thick slices made from our homemade sourdough bread</description>
    <calories>600</calories>
  </food>

  <food>
```

```

<name>Homestyle Breakfast</name>
<price>$6.95</price>
<description>Two eggs, bacon or sausage, toast, and our ever-popular hash browns</description>
<calories>950</calories>
</food>

</breakfast_menu>

```

Use XSLT to transform XML into HTML, before it is displayed in a browser:

Example XSLT Stylesheet:

```

<?xml version="1.0" encoding="UTF-8"?>
<html xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<body style="font-family:Arial;font-size:12pt;background-color:#EEEEEE">
<xsl:for-each select="breakfast_menu/food">
  <div style="background-color:teal;color:white;padding:4px">
    <span style="font-weight:bold"><xsl:value-of select="name"/> - </span>
    <xsl:value-of select="price"/>
  </div>
  <div style="margin-left:20px;margin-bottom:1em;font-size:10pt">
    <p>
      <xsl:value-of select="description"/>
      <span style="font-style:italic"> (<xsl:value-of select="calories"/> calories per serving)</span>
    </p>
  </div>
</xsl:for-each>
</body>
</html>

```