What is DOM?

The W3C DOM provides a standard set of objects for HTML and XML documents, and a standard interface for accessing and manipulating them.

The W3C DOM is separated into different parts (Core, XML, and HTML) and different levels (DOM Level 1/2/3):

* Core DOM - defines a standard set of objects for any structured document
* XML DOM - defines a standard set of objects for XML documents
* HTML DOM - defines a standard set of objects for HTML documents

If you want to learn more about the XML DOM, please visit our XML DOM tutorial.

XML Parsing

To read and update - create and manipulate - an XML document, you will need an XML parser.

There are two basic types of XML parsers:

- Tree-based parser: This parser transforms an XML document into a tree structure. It analyzes the whole document, and provides access to the tree elements
- Event-based parser: Views an XML document as a series of events. When a specific event occurs, it calls a function to handle it

The DOM parser is an tree-based parser.

Look at the following XML document fraction:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<from>Jani</from>
```

The XML DOM sees the XML above as a tree structure:

- Level 1: XML Document
- Level 2: Root element: <from>
- Level 3: Text element: "Jani"

An XML File

The XML file below will be used in our example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
```

```
<body>Don't forget me this weekend!</body>
</note>
```

Load and Output XML

We want to initialize the XML parser, load the xml, and output it:

Example

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");

print $xmlDoc->saveXML();
?>
```

The output of the code above will be:

Tove Jani Reminder Don't forget me this weekend!

If you select "View source" in the browser window, you will see the following HTML:

The example above creates a DOMDocument-Object and loads the XML from "note.xml" into it.

Then the saveXML() function puts the internal XML document into a string, so we can output it.

Looping through XML

We want to initialize the XML parser, load the XML, and loop through all elements of the <note> element:

Example

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");

$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item)
  {
  print $item->nodeName . " = " . $item->nodeValue . "<br />";
  }
?>
```

The output of the code above will be:

In the example above you see that there are empty text nodes between each element.

When XML generates, it often contains white-spaces between the nodes. The XML DOM parser treats these as ordinary elements, and if you are not aware of them, they sometimes cause problems.

What is SimpleXML?

SimpleXML is new in PHP 5. It is an easy way of getting an element's attributes and text, if you know the XML document's layout.

Compared to DOM or the Expat parser, SimpleXML just takes a few lines of code to read text data from an element.

SimpleXML converts the XML document into an object, like this:

- Elements - Are converted to single attributes of the SimpleXMLElement object. When there's more than one element on one level, they're placed inside an array
- Attributes - Are accessed using associative arrays, where an index corresponds to the attribute name
- Element Data - Text data from elements are converted to strings. If an element has more than one text node, they will be arranged in the order they are found

SimpleXML is fast and easy to use when performing basic tasks like:

- Reading XML files
- Extracting data from XML strings
- Editing text nodes or attributes

However, when dealing with advanced XML, like namespaces, you are better off using the Expat parser or the XML DOM.

Using SimpleXML

Below is an XML file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

We want to output the element names and data from the XML file above.

Here's what to do:

1. Load the XML file
2. Get the name of the first element
3. Create a loop that will trigger on each child node, using the children() function
4. Output the element name and data for each child node.

Example

```
<?php
$xml = simplexml_load_file("test.xml");

echo $xml->getName() . "<br />";

foreach($xml->children() as $child)
  {
  echo $child->getName() . ": " . $child . "<br />";
  }
?>
```

The output of the code above will be:

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

Creating XML document using DOM parser

```
<HTML>

<HEAD>

 <TITLE>New Document</TITLE>

</HEAD>

<BODY>

<?

  //Creates XML string and XML document using the DOM

 $dom = new DomDocument('1.0');



 //add root - <books>

 $books = $dom->appendChild($dom->createElement('books'));
```

```
//add <book> element to <books>
$book = $books->appendChild($dom->createElement('book'));

//add <title> element to <book>
$title = $book->appendChild($dom->createElement('title'));

//add <title> text node element to <title>
$title->appendChild(
        $dom->createTextNode('Great American Novel'));

//generate xml
$dom->formatOutput = true; // set the formatOutput attribute of
                    // domDocument to true
// save XML as string or file
$test1 = $dom->saveXML(); // put string in test1
$dom->save('test1.xml'); // save as file
?>
</BODY>
</HTML>
```