www

http://www.w3schools.com/php/default.asp

PHP

- http://www.w3schools.com/php/default.asp
- http://www.php.net/manual/en/index.php
- http://wazniak.mimuw.edu.pl/index.php?title=AWWW-1st3.6-w06.tresc-1.1-Slajd27

What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML, JavaScript code, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have a default file extension of ". php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can restrict users to access some pages on your website
- PHP can encrypt data

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP has support for a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Basic PHP Syntax

http://www.w3schools.com/php/php_syntax.asp

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

With PHP, there are two basic statements to output text in the browser: echo and print.

Comments in PHP

```
<html>
<body>
<?php
//This is a comment
/*
This is
a comment
block
*/
?>
</body>
</html>
```

PHP Variables

http://www.w3schools.com/php/php_variables.asp

All variables in PHP start with a $ sign symbol.

```
<?php
$txt="Hello World!";
$x=16;
?>
```

In PHP, a variable does not need to be declared before adding a value to it.

PHP is a Loosely Typed Language

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must begin with a letter or the underscore character
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- A variable name should not contain spaces
- Variable names are case sensitive ($y and $Y are two different variables)

PHP Variable Scopes

PHP has four different variable scopes:

- local
- global
- static
- parameter

Local Scope

A variable declared within a PHP function is local and can only be accessed within that function:

```
<?php
$x=5; // global scope

function myTest()
{
echo $x; // local scope // wydrukuje pusty napis
}

myTest();
?>
```

Local variables are deleted as soon as the function is completed.

Global Scope

A variable that is defined outside of any function, has a global scope.

To access a global variable from within a function, use the global keyword ... (następny slajd ...)

```php
<?php
$x=5; // global scope
$y=10; // global scope

function myTest()
{
global $x,$y;
$y=$x+$y;
}

myTest();
echo $y; // outputs 15
?>
```

PHP also stores all global variables in an array called $GLOBALS[index].

The index holds the name of the variable.

This array is also accessible from within functions and can be used to update global variables directly.

(następny slajd ...)

```php
<?php
$x=5;
$y=10;

function myTest()
{
$GLOBALS['y']=$GLOBALS['x']+$GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

Static Scope

When a function is completed, all of its variables are normally deleted.

However, sometimes you want a local variable to not be deleted.

To do this, use the static keyword when you first declare the variable:

```php
<?php

function myTest()
{
static $x=0;
echo $x;
$x++;
}

myTest(); // 0
myTest(); // 1
myTest(); // 2

?>
```

Parameter Scope

A parameter is a local variable whose value is passed to the function by the calling code.

```
<?php

function myTest($x)
{
echo $x;
}

myTest(5);

?>
```

PHP String Variables

- The Concatenation Operator

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

PHP String Variables

- The strlen() function

```
<?php
echo strlen("Hello world!");
?>
```

-

PHP String Variables

- The strpos() function

```
<?php
echo strpos("Hello world!","world"); // 6
?>
```

PHP 5 String Functions

http://www.w3schools.com/php/php_ref_string.asp

Arithmetic Operators

http://www.w3schools.com/php/php_operators.asp

- + Addition
- - Subtraction
- * Multiplication
- / Division
- % Modulus (division remainder)
- ++ Increment
- -- Decrement

Assignment Operators

http://www.w3schools.com/php/php_operators.asp

- = x=y x=y
- += x+=y x=x+y
- -= x-=y x=x-y
- *= x*=y x=x*y
- /= x/=y x=x/y
- .= x.=y x=x.y
- %= x%=y x=x%y

PHP Incrementing/Decrementing Operators

++ x	Pre-increment	Increments x by one, then returns x
x ++	Post-increment	Returns x, then increments x by one
-- x	Pre-decrement	Decrements x by one, then returns x
x --	Post-decrement	Returns x, then decrements x by one

Comparison Operators

http://www.w3schools.com/php/php_operators.asp

- == is equal to 5==8 returns false
- != is not equal 5!=8 returns true
- <> is not equal 5<>8 returns true
- > is greater than 5>8 returns false
- < is less than 5<8 returns true
- >= is greater than or equal to 5>=8 returns false
- <= is less than or equal to 5<=8 returns true

http://www.php.net/manual/en/language.operators.comparison.php

- $a === $b $a is equal to $b, and they are of the same type.
- $a !== $b $a is not equal to $b, or they are not of the same type.

Logical Operators

http://www.php.net/manual/en/language.operators.logical.php

- $a and $b
- $a or $b
- $a xor $b
- ! $a Not
- $a && $b And
- $a || $b Or

The reason for the two different variations of "and" and "or" operators is that they operate at different precedences.

PHP Array Operators

x + y Union Union of x and y
The + operator returns the right-hand array appended to the left-hand array;
for keys that exist in both arrays, the elements from the left-hand array will be used, and the matching elements from the right-hand array will be ignored.

x == y Equality True if x and y have the same key/value pairs
x === y Identity True if x and y have the same key/value pairs in the same order and are of the same type
x != y Inequality True if x is not equal to y
x <> y Inequality True if x is not equal to y
x !== y Non-identity True if x is not identical to y

PHP If...Else Statements

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

Note:

Note that elseif and else if will only be considered exactly the same when using curly brackets.

When using a colon to define your if/elseif conditions, you must not separate else if into two words, or PHP will fail with a parse error.

PHP Switch Statement ...

```
<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
default:
  echo "No number between 1 and 2";
}
?>
```

```php
<?php
$favcolor="red";
switch ($favcolor)
{
case "red":
  echo "Your favorite color is red!";
  break;
case "blue":
  echo "Your favorite color is blue!";
  break;
case "green":
  echo "Your favorite color is green!";
  break;
default:
  echo "Your favorite color is neither red, blue, or green!";
}
?>
```

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Arrays

http://www.w3schools.com/php/php_arrays.asp

A numeric array stores each array element with a numeric index.

```
$cars=array("Saab","Volvo","BMW","Toyota");
$cars[0]="Saab";
```

An associative array, each ID key is associated with a value.

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
$ages['Peter'] = "32";
```

In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.

PHP Arrays

In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.

```
$families = array
  (
  "Griffin"=>array ("Peter","Lois","Megan"),
  "Quagmire"=>array ("Glenn"),
  "Brown"=>array("Cleveland","Loretta","Junior")
  );

echo "Is " . $families['Griffin'][2] . " a part of the Griffin family?";
```

Get The Length of an Array

```php
<?php
$cars=array("Volvo","BMW","Toyota");
echo count($cars);
?>
```

Loop Through an Indexed Array

```
<?php
$cars=array("Volvo","BMW","Toyota");
$arrlength=count($cars);

for($x=0;$x<$arrlength;$x++)
  {
  echo $cars[$x];
  echo "<br>";
  }
?>
```

Loop Through an Associative Array

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");

foreach($age as $x=>$x_value)
  {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
  }
?>
```

PHP - Sort Functions For Arrays

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort() - sort associative arrays in descending order, according to the value
- krsort() - sort associative arrays in descending order, according to the key

```php
<?php
$numbers=array(4,6,2,22,11);
sort($numbers);

$arrlength=count($numbers);
for($x=0;$x<$arrlength;$x++)
  {
  echo $numbers[$x];
  echo "<br>";
  }
?>
```

PHP Looping - While Loops

```
<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i++;
  }
?>
```

PHP Looping - While Loops

```
<?php
$i=1;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<=5);
?>
```

PHP Looping - For Loops

```
<?php
for ($i=1; $i<=5; $i++)
  {
  echo "The number is " . $i . "<br />";
  }
?>
```

PHP Looping - For Loops

```
<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>
```

PHP Functions

http://www.w3schools.com/php/php_functions.asp

```
<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>
```

PHP Functions - Adding parameters

```
<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
?>
```

PHP Functions - Return values

```
<?php
function add($x,$y)
{
$total=$x+$y;
return $total;
}

echo "1 + 16 = " . add(1,16);
?>
```

PHP Forms and User Input

http://www.w3schools.com/php/php_forms.asp

The PHP $_GET and $_POST variables are used to retrieve information from forms, like user input.

PHP Forms and User Input

The example below contains an HTML form with two input fields and a submit button:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>

</body>
</html>
```

PHP Forms and User Input

"welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["fname"]; ?>!<br />
You are <?php echo $_POST["age"]; ?> years old.

</body>
</html>
```

PHP $_GET

```
<form action="welcome.php" method="get">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

the URL sent to the server could look something like this:

http://www.w3schools.com/welcome.php?fname=Peter&age=37

The "welcome.php" file can now use the $_GET function

```
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
```

PHP $_GET

This method should not be used when sending passwords or other sensitive information!
However, because the variables are displayed in the URL, it is possible to bookmark the page.

Note: The get method is not suitable for very large variable values. It should not be used with values exceeding 2000 characters.

PHP $_POST

```
<form action="welcome.php" method="post">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

The "welcome.php" file can now use the $_POST function

```
Welcome <?php echo $_POST["fname"]; ?>!<br />
You are <?php echo $_POST["age"]; ?> years old.
```

The PHP $_REQUEST Function

The PHP built-in $_REQUEST function contains the contents of both $_GET, $_POST, and $_COOKIE.

The $_REQUEST function can be used to collect form data sent with both the GET and POST methods.

PHP Date() Function

```
<?php
echo date("Y/m/d") . "<br />";
echo date("Y.m.d") . "<br />";
echo date("Y-m-d");
?>
```

```
<?php
$tomorrow = mktime(0,0,0,date("m"),date("d")+1,date("Y"));
echo "Tomorrow is ".date("Y/m/d", $tomorrow);
?>
```

PHP Include File

You can insert the content of one PHP file into another PHP file before the server executes it

```
<html>
<body>
<?php include("header.php"); ?>
<h1>Welcome to my home page!</h1>
<p>Some text.</p>
</body>
</html>
```

[...] how they handle errors:

- include() generates a warning, but the script will continue execution
- require() generates a fatal error, and the script will stop

PHP File Handling

Reading a File Line by Line

```
<?php
$file = fopen("welcome.txt", "r") or exit("Unable to open file!");
//Output a line of the file until the end is reached
while(!feof($file))
  {
  echo fgets($file). "<br />";
  }
fclose($file);
?>
```

PHP File Handling

Reading a File Character by Character

```
<?php
$file=fopen("welcome.txt","r") or exit("Unable to open file!");
while (!feof($file))
  {
  echo fgetc($file);
  }
fclose($file);
?>
```

PHP File Upload

http://www.w3schools.com/php/php_file_upload.asp
Look at the following HTML form for uploading files:

```
<html>
<body>
<form action="upload_file.php" method="post"
enctype="multipart/form-data">
<label for="file">Filename:</label>
<input type="file" name="file" id="file" />
<br />
<input type="submit" name="submit" value="Submit" />
</form>
</body>
</html>
```

Notice the following about the HTML form above:

- The enctype attribute of the <form> tag specifies which content-type to use when submitting the form. "multipart/form-data" is used when a form requires binary data, like the contents of a file, to be uploaded
- The type="file" attribute of the <input> tag specifies that the input should be processed as a file. For example, when viewed in a browser, there will be a browse-button next to the input field

PHP File Upload

The "upload_file.php" file contains the code for uploading a file:

```
<?php
if ($_FILES["file"]["error"] > 0)
  {
  echo "Error: " . $_FILES["file"]["error"] . "<br />";
  }
else
  {
  echo "Upload: " . $_FILES["file"]["name"] . "<br />";
  echo "Type: " . $_FILES["file"]["type"] . "<br />";
  echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
  echo "Stored in: " . $_FILES["file"]["tmp_name"];
  }
?>
```

Restrictions on Upload

```
<?php
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/pjpeg"))
&& ($_FILES["file"]["size"] < 20000))
  {
    [ . . . ]
  }
else
  {
  echo "Invalid file";
  }
?>
```

Saving the Uploaded File

```
if (file_exists("upload/" . $_FILES["file"]["name"]))
  {
  echo $_FILES["file"]["name"] . " already exists. ";
  }
else
  {
  move_uploaded_file($_FILES["file"]["tmp_name"],
  "upload/" . $_FILES["file"]["name"]);
  echo "Stored in: " . "upload/" . $_FILES["file"]["name"];
  }
```

PHP Cookies

A cookie is a small file that the server embeds on the user's computer.
Each time the same computer requests a page with a browser, it will send the cookie too.

```
<?php
$expire=time()+60*60*24*30;
setcookie("user", "Alex Porter", $expire);
?>

<html>
.....
```

How to Retrieve a Cookie Value?

```php
<?php
// Print a cookie
echo $_COOKIE["user"];

// A way to view all cookies
print_r($_COOKIE);
?>
```

PHP Cookies

In the following example we use the isset() function to find out if a cookie has been set:

```
<html>
<body>
<?php
if (isset($_COOKIE["user"]))
  echo "Welcome " . $_COOKIE["user"] . "!<br />";
else
  echo "Welcome guest!<br />";
?>
</body>
</html>
```

PHP Cookies

How to Delete a Cookie?

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time()-3600);
?>
```

PHP Sessions

Starting a PHP Session
Before you can store user information in your PHP session, you must first start up the session.

Note: The session_start() function must appear BEFORE the <html> tag:

```
<?php session_start(); ?>

<html>
<body>

</body>
</html>
```

PHP Sessions

http://www.w3schools.com/php/php_sessions.asp

In the example below, we create a simple page-views counter.

```
<?php
session_start();

if(isset($_SESSION['views']))
$_SESSION['views']=$_SESSION['views']+1;
else
$_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

Destroying a Session

The unset() function is used to free the specified session variable:

```
<?php
session_start();
if(isset($_SESSION['views']))
  unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the session_destroy() function:

```
<?php
session_destroy();
?>
```

The PHP mail() Function

http://www.w3schools.com/php/php_mail.asp

PHP Error Handling

http://www.w3schools.com/php/php_error.asp

PHP Filter

http://www.w3schools.com/php/php_filter.asp

PHP Exception Handling

http://www.w3schools.com/php/php_exception.asp

PHP Filter

http://www.w3schools.com/php/php_filter.asp

Validating filters:

- Are used to validate user input
- Strict format rules (like URL or E-Mail validating)
- Returns the expected type on success or FALSE on failure

Sanitizing filters:

- Are used to allow or disallow specified characters in a string
- No data format rules
- Always return the string

In the example below, we validate an integer using the filter_var() function:

```
<?php
$int = 123;

if(!filter_var($int, FILTER_VALIDATE_INT))
  {
  echo("Integer is not valid");
  }
else
  {
  echo("Integer is valid");
  }
?>
```

To filter a variable, use one of the following filter functions:

- filter_var() - Filters a single variable with a specified filter
- filter_var_array() - Filter several variables with the same or different filters
- filter_input - Get one input variable and filter it
- filter_input_array - Get several input variables and filter them with the same or different filters

PHP MySQL

http://www.w3schools.com/php/php_mysql_intro.asp

Select Data From a Database Table

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("my_db", $con);
$result = mysql_query("SELECT * FROM Persons");
while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }
mysql_close($con);
?>
```

PHP XML Expat Parser

http://www.w3schools.com/php/php_xml_parser_expat.asp

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. e.g. the Document Object Model (DOM)
- 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 Expat parser is an event-based parser.

PHP XML Expat Parser

Look at the following XML fraction:

<from>Jani</from>

An event-based parser reports the XML above as a series of three events:

1. Start element: from
2. Start CDATA section, value: Jani
3. Close element: from

PHP XML Expat Parser

```
<?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>
```

Initializing the XML Parser

```
<?php
//Initialize the XML parser
$parser=xml_parser_create();
```

Initializing the XML Parser

```
//Function to use at the start of an element
function start($parser,$element_name,$element_attrs)
  {
  switch($element_name)
    {
    case "NOTE":  echo "-- Note --<br />"; break;
    case "TO": echo "To: ";  break;
    case "FROM":  echo "From: "; break;
    case "HEADING": echo "Heading: ";  break;
    case "BODY":  echo "Message: ";
    }
  }
```

Initializing the XML Parser

```
//Function to use at the end of an element
function stop($parser,$element_name)
  {
  echo "<br />";
  }

//Function to use when finding character data
function char($parser,$data)
  {
  echo $data;
  }
```

Initializing the XML Parser

```
//Specify element handler
xml_set_element_handler($parser,"start","stop");

//Specify data handler
xml_set_character_data_handler($parser,"char");

//Open XML file
$fp=fopen("test.xml","r");
```

Initializing the XML Parser

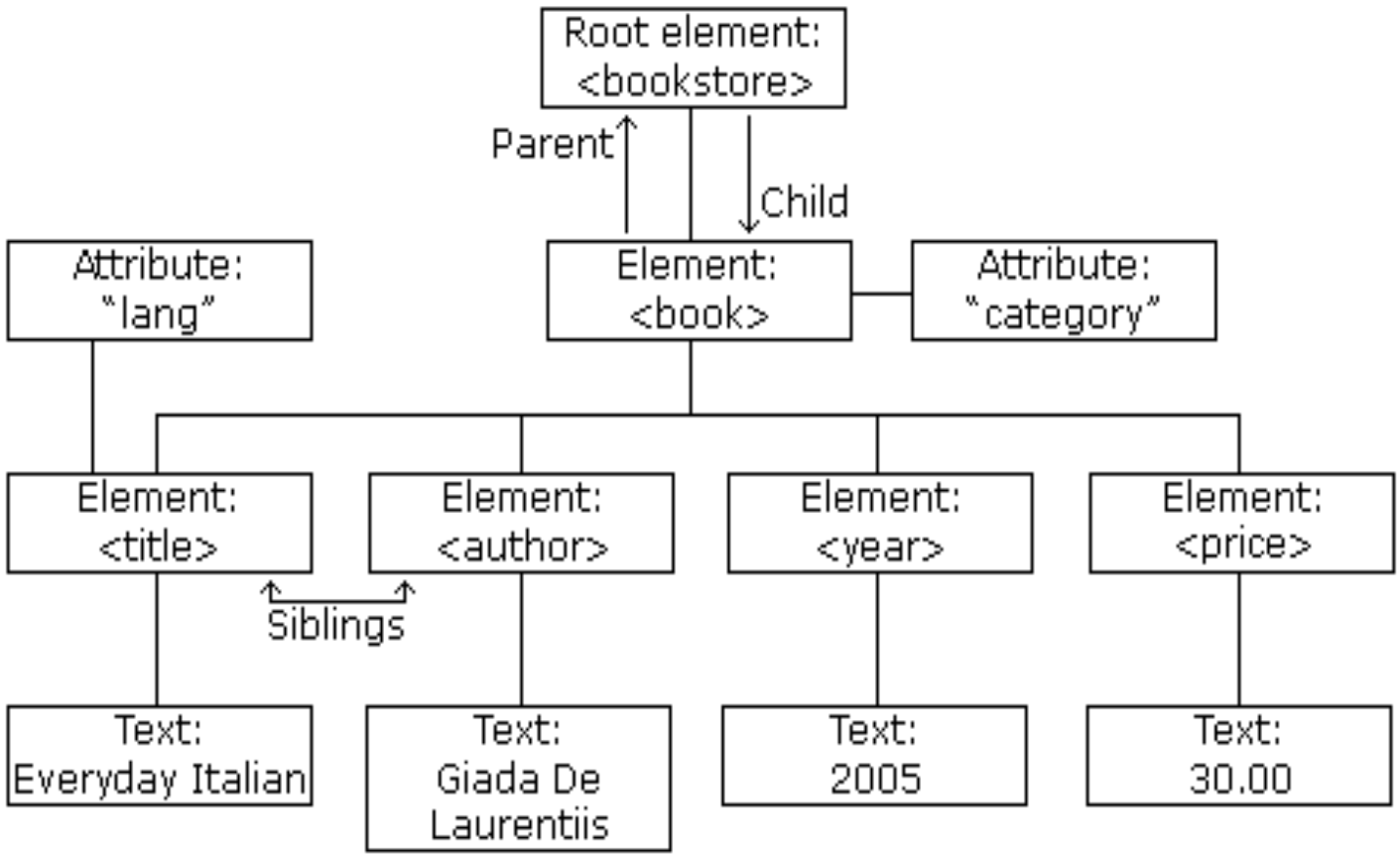
```
//Read data
while ($data=fread($fp,4096))
  {
  xml_parse($parser,$data,feof($fp)) or
  die (sprintf("XML Error: %s at line %d",
  xml_error_string(xml_get_error_code($parser)),
  xml_get_current_line_number($parser)));
  }

//Free the XML parser
xml_parser_free($parser);
?>
```

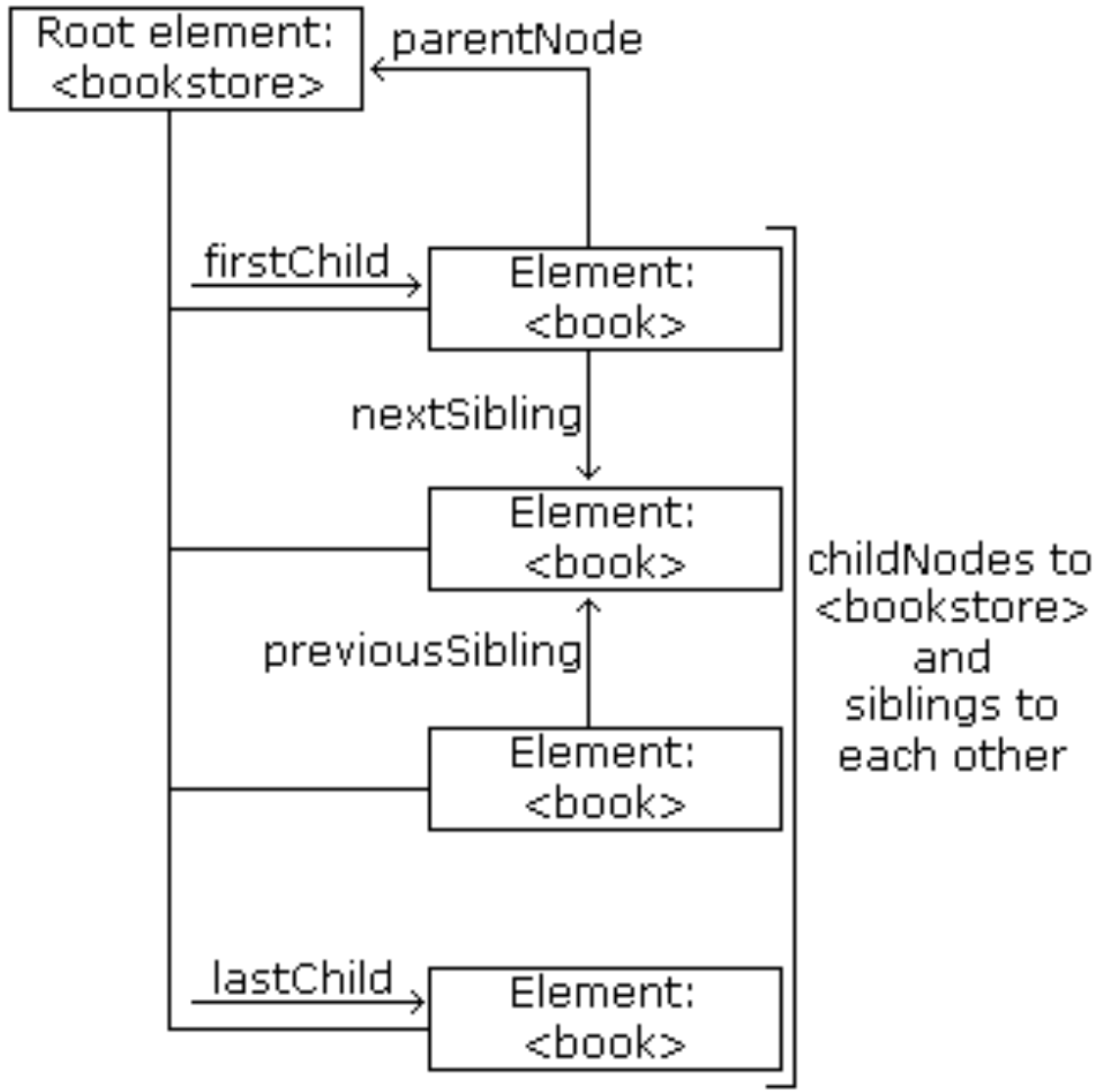
PHP XML DOM

XML DOM

http://www.w3schools.com/dom/default.asp



Node Parents, Children, and Siblings



Load and Output XML

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

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

```
<?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>
```

Looping through XML

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

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

Looping through XML

The output of the code above will be:
#text =
to = Tove
#text =
from = Jani
#text =
heading = Reminder
#text =
body = Don't forget me this weekend!
#text =
In the example above you see that there are empty text nodes between each element.

PHP SimpleXML

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

Output keys and elements of the $xml variable (which is a SimpleXMLElement object):

```
<?php
$xml=simplexml_load_file("note.xml");
print_r($xml);
?>
```

The output of the code above will be:

```
SimpleXMLElement Object ( [to] => Tove [from] => Jani [heading] => Reminder [body] => Don't forget me this weekend! )
```

Output the data from each element in the XML file:

```
<?php
$xml=simplexml_load_file("note.xml");
echo $xml->to . "<br>";
echo $xml->from . "<br>";
echo $xml->heading . "<br>";
echo $xml->body;
?>
```

The output of the code above will be:

Tove
Jani
Reminder
Don't forget me this weekend!

PHP SimpleXML

Output the element's name and data for each child node:

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

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

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

PHP SimpleXML

The output of the code above will be:

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