# PHP: HYPERTEXT PREPROCESSOR

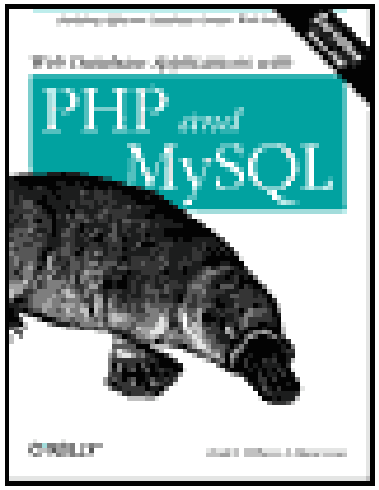
# What we're going to look at:



- Today, we're going to look at :

1. What the PHP is…
2. Why PHP is so bloody great…
3. How to code **properly** in PHP…

# Recommended Reading




**Web Database Application with PHP and MySQL, 2nd Edition**

**By**

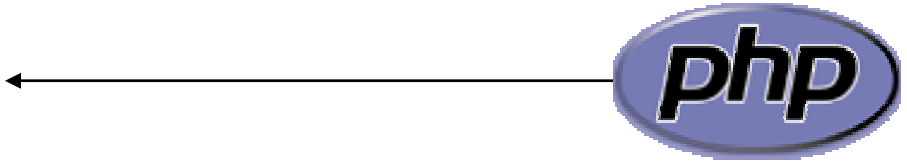
**David Lane,**

**Hugh E. Williams**

# References – Web



- **http://www.php.net**
  - The best PHP site




- **http://www.phpfreaks.com**


- **http://www.phpbuilder.com**

# PHP HISTORY



- PHP is a language for creating interactive web sites.
- It was originally called "**Personal Home Page Tools**" when it was created in **1994** by Rasmus Lerdorf to keep track of who was looking at his online CV.

- **Mid-1997:** students Andi Gutmans and Zeev Suraski redesigned the PHP language engine and wrote some of the most popular PHP modules.

- At that time PHP already had its own site, **php.net,** run by the computer science community, and was powering thousands of Web sites.

# Recursive Names



- PHP originally stood for:

**Personal Home Pages**

- PHP is now a recursive acronym that stands for:

**PHP: Hypertext Preprocessor**

# These Dates are recent…



Version 0.0: Conceived in autumn of 1994
Version 1.0: Personal Home Page Tools in early 1995
Version 2.0: PHP/FI 1995-1997
Version 3.0: PHP 1997-2000
Version 4.0: PHP mid-2000
Version 4.1: 10 Dec 2001
Version 4.2: 22 Apr 2002
...
**Version 5.0: 26 February 2009**

# What is PHP?



- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle etc)
- PHP is open source and free to download
- PHP files many contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have extension of ".php", ".php3" or ".phtml"

# Why PHP?



- PHP runs on different platforms (windows, Linux etc)
- PHP is compatible with almost all servers used today (Apache, IIS etc)
- PHP is FREE to download from official PHP resource: http://www.php.net
- PHP is easy to learn and runs efficiently on the server side

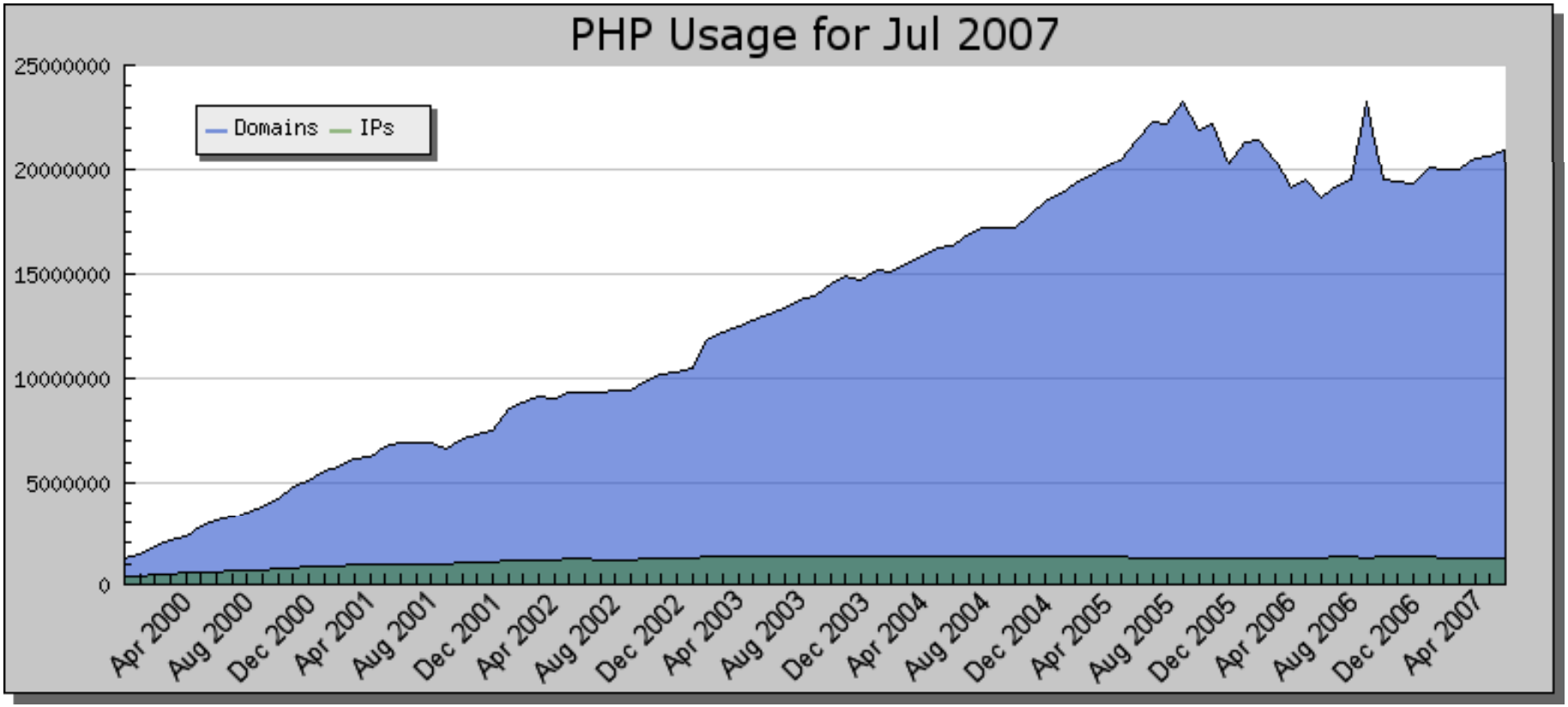
# What you need for PHP?



- Install an Apache Server (web server) on a windows or Linux machine
- Install PHP (server side scripting technlogy) on a windows or Linux machine
- Install MySQL (database server) on a windows or Linux machine
- And lots of Configuration work!!! ☹
- Alternatively, Just download WAMP Server and install
  - It will not only installs Apache, MySQL and PHP on windows machine but will also configure these softwares.
  - Provides you an easy to access interface to run and host PHP files.

# PHP Usage – Jul 2007





# Hello World in PHP



```
<?php

echo 'Hello World'

?>
```

You can also use “print” command instead of “echo”.

# An Embedded Script



- You could view PHP as an embedded language.

- However it is very much still a Server Side Language – this is **not** the same sort of thing as Javascript

# PHP Tag Styles – The bad…



- **Script Style:**

```
<SCRIPT LANGUAGE='php'>
        print "this is script style";
</SCRIPT>
```

- **ASP Style:**

```
<%
        print "this is ASP style";
%>
```

**Both these styles of opening code is perfectly valid – but they are based on other languages.**

# PHP Tag Styles – The good…



- **XML Style:**

```
<?php
        print "this is XML style";
?>
```

- **Short Style:**

```
<?
        print "this is ASP style";
?>
```

To use Short style, the PHP you are using must have "short tags" enabled in its config file… this is almost **always** the case.

# Switching Modes



```
<?
   if($flag)
   {
?>
   <b>You are using Internet Explorer</b>
<?
   }
   else
   {
?>
   <b>You are not using Internet Explorer</b>
<?
   }
?>
```

Using the arrow-question mark tag we can quickly switch modes

Here we are in PHP-mode

Here we have flicked back to HTML again

# PHP is C++ Style



- PHP is very similar to C++.
- This is because C++ is top banana.
- As a consequence if you know java (also a c++ clone), C++ or indeed almost any other computer science language you pretty much already know PHP.
- However more than anything PHP is based on Perl.

# Creating a PHP program



- You can use any text editor. (we will use Dreamweaver)
- Different Web server setups work in different ways. (we will use WAMP Server that bundles Apache, MySQL, PHP)
- Create a new file in Dreamweaver and save it with .php extension.
- Use <?php To open your code and ?> to close it.
- Anything not in those tags is rendered as HTML.
- Type your code and load it up in a web browser.
- Any webpage that use even a single line of php and all other contents are in HTML must be saved with .php extension. Otherwise php code will not run.

# Comments



- Why do we go on about comments so much?
- You can any of the following comment style in php.

```
<?
   // C style comment

   # Perl style comment

   /*
        C++ multi line comment
   */
?>
```

# Hello World ... Once again!...



- Mixing HTML and PHP

```
<HTML>
<body>
Hello World in HTML
<?php
        echo "Is it PHP your looking fooooor…\n";
        // This is comment
?>
This is HTML again
</body>
</HTML>
```

# PHP is for Lazy People



- Yup, you know that means you!

- **Basic data types**
  - **Scalar**
    - numbers (integers and float. Holds 4 bytes of space)
    - strings (Double-quoted "abc“ and single-quoted 'abc' )
    - booleans (true, false )
  - **Compound**
    - Arrays
    - Objects
- **Dynamic typing**
  - Don't have to declare types
  - Automatic conversion done

# Variables in action



- **Every variable must have a $ sign at the beginning of the variable.**

```
<?
   $x = false;                    // boolean
   $x = true;

   $x = 10;                       // decimal
   $x = 1.45;                     // Floating point

   $x = 'Hello World';            //  Hello World
   $x = "mmm\"oo'oo";             //  mmm"oo'oo

   $y = &$x;                      // Reference

   $x[1] = 10;                    // array of decimals
   $x["name"] = "jimbo";          // associative array
   $x[2]["lala"] = "xx";          // a two dimensional array
?>
```

# Variable substitution



- Variable substitution provides a convenient way to embed data held in a variable directly into string literals.
- PHP examines, or parses , double-quoted strings and replaces variable names with the variable's value.
- The following example shows how:

```
<?
$number = 45;
$vehicle = "bus";
$message = "This $vehicle holds $number people";
// prints "This bus holds 45 people"
print $message;
?>
```

# Constants



- Constants associate a name with a scalar value.
- For example, the Boolean values true and false are constants associated with the values 1 and 0, respectively.
- It's also common to declare constants in a script.
- Consider this example constant declaration:

```
<?
define("PI", 3.14159);
// This outputs 3.14159
print PI;
?>
```

# Type conversion



- Conversion between types can be pure (forced) or dirty (automatic)…
- **Forced casting (pure):**

```
<?
        $bool = true;
        print (int)$bool;
?>
```

- **Automatic Type Juggling (dirty):**

```
<?
        $x = "100";
        $x++; // $x is now 101
?>
```

- **Check Point:**

  If $x = "12" and $y = "13"

  What will be the output for $x . $y and $x + $y?

- For more on type conversion, must read Topic 2.5-Working with Types in Chapter 2.

# Variable Scope



- The 3 basic types of scope in PHP is:
  - Global variables declared in a script are visible throughout that script, but not inside functions
    - Declared as: global $x;
  - Variables used inside functions are local (limited) to the function
    - By default a variable inside a function is local.
    - You can also define a local variable as: local $x;
  - Variables used inside functions that are declared as **global** refer to the global variable of the same name

# Numerical functions – as per normal



- Addition

```
$a = 1 + 1;          // sets $a to 2
$a += 4;                  // adds 4 to $a
$a++;                // adds 1 to $a
```

Subtraction

```
$a = 10 - 5;              # sets $a to 5
$a -= 6;             # subtracts 6 from $a
$a--;                # subtracts 1 from $a
```

# Final Numerical functions



- Multiplication

```
$a = 2 * 3;          // sets $a to 6
$a *= 10;            // multiplies $a by 10
```

- Division

```
$a = 10 / 3;         // sets $a to 3.3333
$a /= 2;             // halves $a
```

- Modulus

```
$a = 10 % 3;         // sets $a to 1
$a %= 2;             // sets $a to modulus 2
                     // of itself
```

# PHP Operators



| | |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulus |
| & | And (00001101 & 00000111 → 00000101) |
| \| | Or (00001101 \| 00000111 → 00001111) |
| ^ | Xor (00001101 ^ 00000111 → 00001010) |
| . | add string (concatenation) |
| << | Shift left (00001101 → 00011010) |
| >> | Shift right(00001101 → 00000110) |

# Simple String functions



| | |
|---|---|
| ***chr -*** | *Return a specific character* |
| ***ord -*** | *Return ASCII value of character* |
| ***strlen -*** | *Get string length* |
| ***strpos -*** | *Find position of first occurrence of a string* |
| ***strrev -*** | *Reverse a string* |
| ***strtolower -*** | *Make a string lowercase* |
| ***strtoupper -*** | *Make a string uppercase* |
| ***str_replace -*** | *Replace all occurrences of the search string with the replacement string* |

# Deja Vu



- Just for the record what's the same as C++/Java?
  - For Loops
  - While Loops
  - If Statements
  - Break, Continue, Exit, Switch, etc.
- The main concepts which differ in syntax are:
  - Functions
  - Classes

# More déjà vu – If statements



- **`if`** statements

```
if (some test)
{
   // code here
}
else if (some other test)
{
   // code here
}
else
{
   // if neither test met, do this
}
```

# Conditional tests



| | |
|---|---|
| == | Equals |
| > | Greater than |
| < | Less than |
| >= | Greater or equal to |
| <= | Less or equal to |
| != | Not equal |

This works exactly the same way for strings as it does for integers and floats.

# If – exists



- *You often* need to check whether a variable exists in PHP.

- There are two ways to do this…

```
if ($a)
{
   print "\$a exists";
}
```

```
if (!empty($a))
{
   print "\$a exists";
}
```

# Foreach loops – Useful things

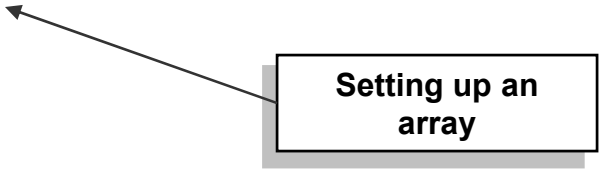


```
<?
   //values
   $a = array (1, 2, 3, 17);

   foreach ($a as $v)
   {
      print "Current value of \$v: $v.<br>";
   }

   // values and keys
   $a = array (1, 2, 3, 17);
   foreach($a as $k => $v)
   {
       print "\$a[$k] is $v<br>";
   }
?>
```

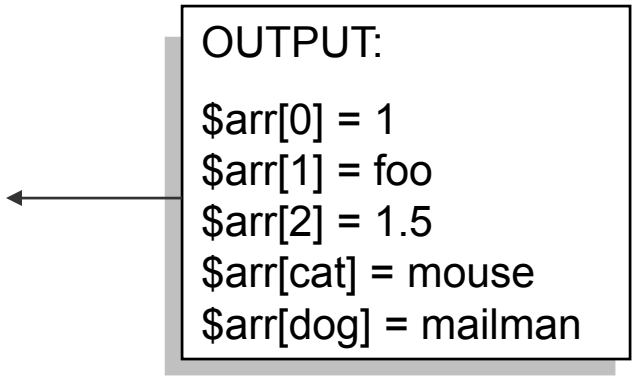
# Mixing up types in Arrays



```
<?
   $arr =
   array(1,'foo',1.57,'cat'=>'mouse','dog'=>'mailman');
?>
```




```
<?
   foreach( $arr as $k=>$v )
   {
       print "\$arr[$k] = $v<br>\n";
   }
?>
```



# PHP Functions



- Functions in PHP are not case sensitive.
- Be careful of this because variable naming is case sensitive

```
function my_function()
{
  print "My function was called";
}
```

- Functions can be created anywhere in your PHP code
- However good style demands they should always be at the top of your code

# Do what is neatest…



```
<?
   Function my_function()
   {
        print "My function was called";
   }
?>
```

IS IDENTICAL TO…

```
<?
   Function my_function()
   {
?>
        My function was called
<?
   }
?>
```

# Passing Parameters



- As normal you don't have to specify the types of your parameters...
- Be careful of this because variable naming is case sensitive

```
<?
function display_table($data)
{
   print "<TABLE border=1>";
   print "<TR><TH>Key</TH><TH>Value</TH></TR>";
   foreach($data as $key=>$value)
        print "<TR><TD>$key</TD><TD>$value</TD></TR>";
   print "</TABLE>";
}

$arr = array(1,'foo',1.57,'cat'=>'mouse','dog'=>'mailman', "abc");
display_table($arr);
?>
```

# Defaults & Passing by Reference



- As with other languages you can set defaults. PHP requires that you do specify the correct number of parameters in a call.

- Equally you don't have to pass by value – an ampersand will mean the variable is passed by reference and so any changes to it are global:

```
function increment(&$value, $amount=1)
{
  $value = $value + $amount;
}
```

# Returning Stuff



- The keyword return stops the execution of a function.
- Your program can contain more than one return statements, the first one encountered in the program flow will end the function.
- You don't need to specify a return type in the function declaration…

```
function larger($x, $y)
{
   if ($x > $y)
         return $x;

   if ($x < $y)
         return $y;

   if ($x == $y)
         return "x and y have the same value";
}
```

# Object Orientation



- Much better in PHP 4 and 5 than its previous version
- However still rather simplified:

```
class classname
{
    var $attribute1;
    var $attribute2;
    function method1(){ }
}
```

- No real data encapsulation.
- **Don't worry about it.** Almost everything you need can be done using functions.

# Conclusion



- PHP is easy. It does most things for you.
- From now on we will be looking at why it is **Good** as opposed to *simple*.
- Its tricks, features and functions are excellent.

- **Reading Book:**
  - Chapter 1 and 2 are mandatory (Might help you in OHT-2 preparation)
  - Reading PHP basics from www.w3schools.com
  - OHT-2 = JavaScript + Above 2 PHP chapters