

12.1 Origins and Uses of PHP

- *Origins*
 - Rasmus Lerdorf - 1994
 - Developed to allow him to track visitors to his Web site
- PHP is an open-source product
- PHP is an acronym for *Personal Home Page*, or *PHP: Hypertext Preprocessor*
- PHP is used for form handling, file processing, and database access

12.2 Overview of PHP

- PHP is a server-side scripting language whose scripts are embedded in HTML documents
 - Similar to JavaScript, but on the server side
- PHP is an alternative to CGI, Active Server Pages (ASP), and Java Server Pages (JSP)
- The PHP processor has two modes: copy (HTML) and interpret (PHP)

12.2 Overview of PHP (continued)

- PHP syntax is similar to that of JavaScript
- PHP is dynamically typed

12.3 General Syntactic Characteristics

- PHP code can be specified in an HTML document internally or externally:

Internally: `<?php`
 ...
 `?>`

Externally: `include ("myScript.inc")`

- the file can have both PHP and HTML
- If the file has PHP, the PHP must be in `<?php .. ?>`, even if the `include` is already in `<?php .. ?>`
- All variable names begin with \$

12.3 General Syntactic Characteristics (continued)

- **Comments** - three different kinds (Java and Perl)

```
// ...  
# ...  
/* ... */
```

- Compound statements are formed with braces
- Compound statements cannot be blocks

12.4 Primitives, Operations, and Expressions

- Variables

- There are no type declarations
- An unassigned (unbound) variable has the value, `NULL`
 - The `unset` function sets a variable to `NULL`
 - The `isset` function is used to determine whether a variable is `NULL`

12.4 Primitives, Operations, and Expressions (continued)

- `error_reporting(15);` - prevents PHP from using unbound variables
- PHP has many predefined variables, including the environment variables of the host operating system
 - You can get a list of the predefined variables by calling `phpinfo()` in a script
- *There are eight primitive types:*
 - Four scalar types: Boolean, integer, double, and string
 - Two compound types: array and object
 - Two special types: resource and `NULL`
- Integer & double are typical
- *Strings*
 - Characters are single bytes
 - String literals use single or double quotes

12.4 Primitives, Operations, and Expressions (continued)

- *Single-quoted string literals*

- Embedded variables are NOT interpolated
- Embedded escape sequences are NOT recognized

- *Double-quoted string literals*

- Embedded variables ARE interpolated
 - If there is a variable name in a double-quoted string but you don't want it interpolated, it must be backslashed

- Embedded escape sequences ARE recognized

- For both single- and double-quoted literal strings, embedded delimiters must be backslashed

- **Boolean** - values are `true` and `false` (case insensitive)

- `0` and `""` and `"0"` are false; others are true

12.4 Primitives, Operations, and Expressions (continued)

- *Arithmetic Operators and Expressions*

- Usual operators
- If the result of integer division is not an integer, a double is returned
- Any integer operation that results in overflow produces a double
- The modulus operator coerces its operands to integer, if necessary
- When a double is rounded to an integer, the rounding is always towards zero

- *Arithmetic functions*

- `floor`, `ceil`, `round`, `abs`, `min`, `max`, `rand`, etc.

- *String Operations and Functions*

- The only operator is period, for catenation
- Indexing - `$str{3}` is the fourth character

12.4 Primitives, Operations, and Expressions (continued)

- *String Operations and Functions* (continued)

- **Functions:**

`strlen`, `strcmp`, `strpos`, `substr`, as in C

`chop` – remove whitespace from the right end

`trim` – remove whitespace from both ends

`ltrim` – remove whitespace from the left end

`strtolower`, `strtoupper`

- *Scalar Type Conversions*

- *String to numeric*

- If the string contains an e or an E, it is converted to double; otherwise to int

- If the string does not begin with a sign or a digit, zero is used

12.4 Primitives, Operations, and Expressions (continued)

- *Scalar Type Conversions* (continued)

- **Explicit conversions – casts**

- e.g., `(int)$total` or `intval($total)` or `settype($total, "integer")`

- **The type of a variable can be determined with `gettype` or `is_type`**

`gettype($total)` - it may return "unknown"

`is_integer($total)` – a predicate function

12.5 Output

- Output from a PHP script is HTML that is sent to the browser

- HTML is sent to the browser through standard output

12.5 Output (continued)

- There are three ways to produce output: `echo`, `print`, and `printf`

- `echo` and `print` take a string, but will coerce other values to strings

```
echo "whatever";    # Only one parameter
```

```
echo("first <br />", $sum) # More than one
```

```
print "Welcome to my site!"; # Only one
```

- PHP code is placed in the body of an HTML document

- An Example:

```
<html>
<head><title> Trivial php example </title>
</head>
<body>
<?php
    print "Welcome to my Web site!";
?>
</body>
</html>
```

12.6 Control Statements

- Control Expressions

- Relational operators - same as JavaScript, (including `===` and `!==`)

- Boolean operators - same as Perl (two sets, `&&` and `and`, etc.)

- Selection statements

- `if`, `if-else`, `elseif`

- `switch` - as in C

- The `switch` expression type must be integer, double, or string

- `while` - just like C

- `do-while` - just like C

- `for` - just like C

- `foreach` - discussed later

12.6 Control Statements (continued)

- **break** - in any `for`, `foreach`, `while`, `do-while`, or `switch`
- **continue** - in any loop
- **Alternative compound delimiters** – more readability

```
if(...):  
    ...  
endif;
```

→ **SHOW** `powers.html`

- **HTML can be intermingled with PHP script**

```
<?php  
    $a = 7;  
    $b = 7;  
    if ($a == $b) {  
        $a = 3 * $a;  
    }  
?>  
<br /> At this point, $a and $b are  
equal <br />  
So, we change $a to three times $a  
<?php  
}  
?>
```

12.7 Arrays

- Not like the arrays of any other programming language
- A PHP array is a generalization of the arrays of other languages
 - A PHP array is really a mapping of keys to values, where the keys can be numbers (to get a traditional array) or strings (to get a hash)
- Array creation
 - Use the `array()` construct, which takes one or more `key => value` pairs as parameters and returns an array of them
 - The keys are non-negative integer literals or string literals
 - The values can be anything
 - e.g., `$list = array(0 => "apples", 1 => "oranges", 2 => "grapes")`
 - This is a “regular” array of strings

12.7 Arrays (continued)

- If a key is omitted and there have been integer keys, the default key will be the largest current key + 1
- If a key is omitted and there have been no integer keys, 0 is the default key
- If a key appears that has already appeared, the new value will overwrite the old one

- Arrays can have mixed kinds of elements

- e.g.,

```
$list = array("make" => "Cessna",  
             "model" => "C210",  
             "year" => 1960,  
             3 => "sold");
```

```
$list = array(1, 3, 5, 7, 9);
```

```
$list = array(5, 3 => 7, 5 => 10,  
             "month" => "May");
```

```
$colors = array('red', 'blue', 'green',  
               'yellow');
```

12.7 Arrays (continued)

- *Accessing array elements* – use brackets

```
$list[4] = 7;  
$list["day"] = "Tuesday";  
$list[] = 17;
```

- If an element with the specified key does not exist, it is created
- If the array does not exist, the array is created

- The keys or values can be extracted from an array

```
$highs = array("Mon" => 74, "Tue" => 70,  
              "Wed" => 67, "Thu" => 62,  
              "Fri" => 65);  
$days = array_keys($highs);  
$temps = array_values($highs);
```

- *Dealing with Arrays*

- An array can be deleted with `unset`

```
unset($list);  
unset($list[4]); # No index 4 element now
```

12.7 Arrays (continued)

- *Dealing with Arrays* (continued)

- `is_array($list)` returns true if `$list` is a function
- `in_array(17, $list)` returns true if 17 is an element of `$list`
- `explode(" ", $str)` creates an array with the values of the words from `$str`, split on a space
- `implode(" ", $list)` creates a string of the elements from `$list`, separated by a space

- *Sequential access to array elements*

- current and next

```
$colors = array("Blue", "red", "green",  
               "yellow");  
$color = current($colors);  
print("$color <br />");  
while ($color = next($colors))  
    print ("$color <br />");
```

12.7 Arrays (continued)

- This does not always work – for example, if the value in the array happens to be `FALSE`

- Alternative: `each`, instead of `next`

```
while ($element = next($colors)) {  
    print ("$element['value'] <br />");  
}
```

- The `prev` function moves current backwards

- `array_push($list, $element)` and
`array_pop($list)`

- Used to implement stacks in arrays

- `foreach (array_name as scalar_name) { ... }`

```
foreach ($colors as $color) {  
    print "Is $color your favorite?<br /> ";  
}
```

```
Is red your favorite color?  
Is blue your favorite color?  
Is green your favorite color?  
Is yellow your favorite color?
```

12.7 Arrays (continued)

- `foreach` can iterate through both keys and values:

```
foreach ($colors as $key => $color) { ... }
```

- Inside the compound statement, both `$key` and `$color` are defined

```
$ages = array("Bob" => 42, "Mary" => 43);  
foreach ($ages as $name => $age)  
    print("$name is $age years old <br />");
```

12.7 Arrays (continued)

- `sort`

- To sort the values of an array, leaving the keys in their present order - *intended for traditional arrays*

e.g., `sort($list);`

- The `sort` function does not return anything

- Works for both strings and numbers, even mixed strings and numbers

```
$list = ('h', 100, 'c', 20, 'a');  
sort($list);  
// Produces (20, 100, 'a', 'c', 'h')
```

- In PHP 4, the `sort` function can take a second parameter, which specifies a particular kind of sort

```
sort($list, SORT_NUMERIC);
```

- `asort`

- To sort the values of an array, but keeping the key/value relationships - *intended for hashes*

12.7 Arrays (continued)

- `rsort`

- To sort the values of an array into reverse order

- `ksort`

- To sort the elements of an array by the keys, maintaining the key/value relationships

e.g.,

```
$list("Fred" => 17, "Mary" => 21,  
      "Bob" => 49, "Jill" => 28);  
ksort($list);  
// $list is now ("Bob" => 49,  
// "Fred" => 17, "Jill" => 28, "Mary" => 21)
```

- `krsort`

- To sort the elements of an array by the keys into reverse order

→ **SHOW** `sorting.php`

12.8 User-Defined Functions

- *Syntactic form:*

```
function function_name (formal_parameters) {  
    ...  
}
```

- *General Characteristics*

- Functions need not be defined before they are called (in PHP 3, they must)
- Function overloading is not supported
- If you try to redefine a function, it is an error
- Functions can have a variable number of parameters
- Default parameter values are supported
- Function definitions can be nested
- Function names are NOT case sensitive
- The `return` function is used to return a value; If there is no `return`, there is no returned value

12.8 User-Defined Functions (continued)

- Parameters

- If the caller sends too many actual parameters, the subprogram ignores the extra ones
- If the caller does not send enough parameters, the unmatched formal parameters are unbound
- The default parameter passing method is pass by value (one-way communication)
- To specify pass-by-reference, prepend an ampersand to the formal parameter

```
function addOne(&$param) {  
    $param++;  
}  
  
$it = 16;  
addOne($it); // $it is now 17
```

- If the function does not specify its parameter to be pass by reference, you can prepend an ampersand to the actual parameter and still get pass-by-reference semantics

```
function subOne($param) { $param--; }  
$it = 16;  
subOne(&$it); // $it is now 15
```

12.8 User-Defined Functions (continued)

- Return Values

- Any type may be returned, including objects and arrays, using the `return`
- If a function returns a reference, the name of the function must have a prepended ampersand

```
function &newArray($x) { ... }
```

- The Scope of Variables

- An undeclared variable in a function has the scope of the function
- To access a nonlocal variable, it must be declared to be global, as in

```
global sum;
```

- The Lifetime of Variables

- Normally, the lifetime of a variable in a function is from its first appearance to the end of the function's execution

```
static sum = 0; # sum is a static variable
```

12.9 Pattern Matching

- PHP has two kinds:

- POSIX
- Perl-compatible

```
preg_match(regex, str [,array])
```

- The optional array is where to put the matches

12.10 Form Handling

- Simpler with PHP than either CGI or servlets

- Forms could be handled by the same document that creates the form, but that may be confusing

- PHP particulars:

- It does not matter whether GET or POST method is used to transmit the form data

- PHP builds a variable for each form element with the same name as the element

→ SHOW popcorn3.html

→ SHOW popcorn3.php

12.11 Files

- PHP can:

- Deal with any files on the server

- Deal with any files on the Internet, using either http or ftp

- Instead of filehandles, PHP associates a variable with a file, called the *file variable* (for program reference)

- A file has a file pointer (where to read or write)

```
$fptr = fopen(filename, use_indicator)
```

Use indicators:

- r read only, from the beginning
- r+ read and write, from the beginning
- w write only, from the beginning (also creates the file, if necessary)
- w+ read and write, from the beginning (also creates the file, if necessary)
- a write only, at the end, if it exists (creates the file, if necessary)
- a+ read and write, read at the beginning, write at the end

- Because `fopen` could fail, use it with `die`

12.11 Files (continued)

- Use `file_exists(filename)` to determine whether file exists before trying to open it

- Use `fclose(file_var)` to close a file

- Reading files

1. Read all or part of the file into a string variable

```
$str = fread(file_var, #bytes)
```

- To read the whole file, use `filesize(file_name)` as the second parameter

2. Read the lines of the file into an array

```
$file_lines = file(file_name)
```

- Need not open or close the file

3. Read one line from the file

```
$line = fgets(file_var, #bytes)
```

- Reads characters until `eoln`, `eof`, or `#bytes` characters have been read

12.11 Files (continued)

- Reading files (continued)

4. Read one character at a time

```
$ch = fgetc(file_var)
```

- Control reading lines or characters with eof detection using `feof` (`TRUE` for eof; `FALSE` otherwise)

```
while(feof($file_var)) {  
    $ch = fgetc($file_var);  
}
```

- Writing to files

```
$bytes_written = fwrite(file_var, string)
```

- `fwrite` returns the number of bytes it wrote

- Files can be locked (to avoid interference from concurrent accesses) with `flock` (just like Perl)

12.12 Cookies

- Create a cookie with `setcookie`

```
setcookie(cookie_name, cookie_value, lifetime)
```

e.g.,

```
setcookie("voted", "true", time() + 86400);
```

- Cookies must be created before any other HTML is created by the script
- Cookies are obtained in a script the same way form values are gotten
- There could be conflicts between GET, POST, and cookie variables
 - PHP puts all POST form variables in their own array (hash) , `HTTP_POST_VARS`
 - Ditto for GET form variables (`HTTP_GET_VARS`)

12.13 Session Tracking

- For session tracking, PHP creates and maintains a session tracking id
- Create the id with a call to `session_start` with no parameters
- Subsequent calls to `session_start` retrieves any session variables that were previously registered in the session
- To create a session variable, use `session_register`
 - The only parameter is a string literal of the name of the session variable (without the dollar sign)
 - Example: count number of pages visited
 - Put the following code in all documents

```
session_start();  
if (!isset($page_number))  
    $page_number = 1;  
print("You have now visited $page_number");  
print(" of pages <br />");  
$page_number++;  
$session_register("page_number");
```