POLI
[TECH>
NIKA
Czestochowa
University
of Technology

[@> Faculty of Computer Science
and Artificial Intelligence

SCRIPTING LANGUAGES IN WEB APPLICATIONS

L06 – PHP syntax

Web Programming Step by Step

Lecture 7
More PHP; File I/O
Reading: 5.2, 5.4

5.2: PHP Basic Syntax

PHP syntax template

```
HTML content

  <?php
    PHP code
  ?>

HTML content

  <?php
    PHP code
  ?>

HTML content ...
```

- any contents of a `.php` file between `<?php` and `?>` are executed as PHP code
- all other contents are output as pure HTML
- can switch back and forth between HTML and PHP "modes"

Math operations

```php
$a = 3;
$b = 4;
$c = sqrt(pow($a, 2) + pow($b, 2));
```

abs	ceil	cos	floor	log	log10	max
min	pow	rand	round	sin	sqrt	tan

math functions

M_PI	M_E	M_LN2

math constants

- the syntax for method calls, parameters, returns is the same as Java

int and float types

```
$a = 7 / 2;               # float: 3.5
$b = (int) $a;            # int: 3
$c = round($a);           # float: 4.0
$d = "123";               # string: "123"
$e = (int) $d;            # int: 123
```

- `int` for integers and `float` for reals
- division between two `int` values can produce a `float`

`String` type (5.2.6)

```
$favorite_food = "Ethiopian";
print $favorite_food[2];            # h

$favorite_food = $favorite_food . " cuisine";
print $favorite_food;               # Ethiopian cuisine
```

- zero-based indexing using bracket notation
- there is no `char` type; each letter is itself a `String`
- string concatenation operator is `.` (period), not `+`
    - `5 + "2 turtle doves" === 7`
    - `5 . "2 turtle doves" === "52 turtle doves"`
- can be specified with `""` or `''`

String functions

```
# index  0123456789012345
$name = "Stefanie Hatcher";
$length = strlen($name);                 # 16
$cmp = strcmp($name, "Brian Le");        # > 0
$index = strpos($name, "e");             # 2
$first = substr($name, 9, 5);            # "Hatch"
$name = strtoupper($name);               # "STEFANIE HATCHER"
```

PHP

Name	Java Equivalent
strlen	length
strpos	indexOf
substr	substring
strtolower, strtoupper	toLowerCase, toUpperCase
trim	trim
explode, implode	split, join
strcmp	compareTo

bool (Boolean) type (5.2.8)

```php
$feels_like_summer = FALSE;
$php_is_rad = TRUE;

$student_count = 217;
$nonzero = (bool) $student_count;     # TRUE
```

PHP

- the following values are considered to be `FALSE` (all others are `TRUE`):
  - `0` and `0.0`
  - `""`, `"0"`, and `NULL` (includes unset variables)
  - arrays with 0 elements
- can cast to boolean using `(bool)`
- `FALSE` prints as an empty string (no output); `TRUE` prints as a `1`

- `TRUE` and `FALSE` keywords are case insensitive

NULL

```php
$name = "Victoria";
$name = NULL;
if (isset($name)) {
  print "This line isn't going to be reached.\n";
}
```

- a variable is `NULL` if
  - it has not been set to any value (undefined variables)
  - it has been assigned the constant `NULL`
  - it has been deleted using the `unset` function
- can test if a variable is `NULL` using the `isset` function
- `NULL` prints as an empty string (no output)

Arrays (5.4.3)

```
$name = array();                          # create
$name = array(value0, value1, ..., valueN);

$name[index]                              # get element value
$name[index] = value;                     # set element value
$name[] = value;                          # append
```

```
$a = array();      # empty array (length 0)
$a[0] = 23;        # stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!";    # add string to end (at index 5)
```

- to append, use bracket notation without specifying an index
- element type is not specified; can mix types

Array functions

function name(s)	description
`count`	number of elements in the array
`print_r`	print array's contents
`array_pop`, `array_push`, `array_shift`, `array_unshift`	using array as a stack/queue
`in_array`, `array_search`, `array_reverse`, `sort`, `rsort`, `shuffle`	searching and reordering
`array_fill`, `array_merge`, `array_intersect`, `array_diff`, `array_slice`, `range`	creating, filling, filtering
`array_sum`, `array_product`, `array_unique`, `array_filter`, `array_reduce`	processing elements

Array function example

```php
$tas = array("MD", "BH", "KK", "HM", "JP");
for ($i = 0; $i < count($tas); $i++) {
  $tas[$i] = strtolower($tas[$i]);
}                                  # ("md", "bh", "kk", "hm", "jp")
$morgan = array_shift($tas);       # ("bh", "kk", "hm", "jp")
array_pop($tas);                   # ("bh", "kk", "hm")
array_push($tas, "ms");            # ("bh", "kk", "hm", "ms")
array_reverse($tas);               # ("ms", "hm", "kk", "bh")
sort($tas);                        # ("bh", "hm", "kk", "ms")
$best = array_slice($tas, 1, 2);   # ("hm", "kk")
```

- the array in PHP replaces many other collections in Java
    - list, stack, queue, set, map, ...

The foreach loop (5.4.4)

```
foreach ($array as $variableName) {
  ...
}
```

```
$stooges = array("Larry", "Moe", "Curly", "Shemp");
for ($i = 0; $i < count($stooges); $i++) {
  print "Moe slaps {$stooges[$i]}\n";
}
foreach ($stooges as $stooge) {
  print "Moe slaps $stooge\n";  # even himself!
}
```

- a convenient way to loop over each element of an array without indexes

5.4: PHP File Input

PHP file I/O functions (5.4.5)

function name(s)	category
file, **file_get_contents**, **file_put_contents**	reading/writing entire files
basename, file_exists, filesize, fileperms, filemtime, is_dir, is_readable, is_writable, disk_free_space	asking for information
copy, rename, unlink, chmod, chgrp, chown, mkdir, rmdir	manipulating files and directories
glob, **scandir**	reading directories

Reading/writing files

contents of `foo.txt`	`file("foo.txt")`	`file_get_contents("foo.txt")`
`Hello`	`array(`	`"Hello\n`
`how r u?`	`  "Hello\n",      # 0`	`how r u?\n       # a single`
	`  "how r u?\n",   # 1`	`\n               # string`
`I'm fine`	`  "\n",           # 2`	`I'm fine\n"`
	`  "I'm fine\n"    # 3`	
	`)`	

- `file` returns lines of a file as an array (`\n` at end of each)
- `file_get_contents` returns entire contents of a file as a single string
  - `file_put_contents` writes a string into a file

The file function

```
# display lines of file as a bulleted list
$lines = file("todolist.txt");
foreach ($lines as $line) {        # for ($i = 0; $i < count($lines); $i++)
  print "<li>$line</li>\n";
}
```
PHP

- `file` returns the lines of a file as an array of strings
- each ends with `\n` ; to strip it, use an optional second parameter:

```
$lines = file("todolist.txt", FILE_IGNORE_NEW_LINES);
```
PHP

- common idiom: `foreach` or `for` loop over lines of file

Unpacking an array: list

```
list($var1, ..., $varN) = array;
```
PHP

contents of input file `personal.txt`

```
Marty Stepp
(206) 685-2181
570-86-7326
```

```
list($name, $phone, $ssn) = file("personal.txt");
...
```
PHP

- the odd `list` function "unpacks" an array into a set of variables you declare
- when you know a file's exact length/format, use `file` and `list` to unpack it

Reading directories

function	description
scandir	returns an array of all file names in a given directory (returns just the file names, such as "myfile.txt")
glob	returns an array of all file names that match a given pattern (returns a file path and name, such as "foo/bar/myfile.txt")

- glob can accept a general path with the * wildcard character

glob example

```
# reverse all poems in the poetry directory
$poems = glob("poetry/poem*.dat");
foreach ($poems as $poemfile) {
  $text = file_get_contents($poemfile);
  file_put_contents($poemfile, strrev($text));
  print "<p>I just reversed " . basename($poemfile) . "</p>\n";
}
```

PHP

- glob can match a "wildcard" path with the * character
  - glob("foo/bar/*.doc") returns all .doc files in the foo/bar subdirectory
  - glob("food*") returns all files whose names begin with "food"
- the basename function strips any leading directory from a file path
  - basename("foo/bar/baz.txt") returns "baz.txt"

scandir example

```
<ul>
  <?php
  $folder = "taxes/old";
  foreach (scandir($folder) as $filename) {
    print "<li>$filename</li>\n"
  }
  ?>
</ul>
```

PHP

```
• .
• ..
• 2007_w2.pdf
• 2006_1099.doc
```

output

- `scandir` sucks; current directory (`"."`) and parent (`".."`) are included in the array
- don't need `basename` with `scandir`; returns file names only without directory

Web Programming Step by Step

Lecture 8
Embedded PHP
Reading: 5.3 - 5.5

Functions; More File I/O

Functions (5.4.1)

```
function name(parameterName, ..., parameterName) {
  statements;
}
```

```
function bmi($weight, $height) {
  $result = 703 * $weight / $height / $height;
  return $result;
}
```

- parameter types and return types are not written
- a function with no `return` statements is implicitly "void"
- can be declared in any PHP block, at start/end/middle of code

Calling functions

```
name(expression, ..., expression);
```

```
$w = 163;  # pounds
$h = 70;   # inches
$my_bmi = bmi($w, $h);
```

- if the wrong number of parameters are passed, it's an error

Variable scope: global and local vars

```php
$school = "UW";                  # global
...

function downgrade() {
  global $school;
  $suffix = "Tacoma";            # local

  $school = "$school $suffix";
  print "$school\n";
}
```

- variables declared in a function are **local** to that function; others are **global**
- if a function wants to use a global variable, it must have a `global` statement
  - but don't abuse this; mostly you should use parameters

Default parameter values

```
function name(parameterName = value, ..., parameterName = value) {
  statements;
}
```

```
function print_separated($str, $separator = ", ") {
  if (strlen($str) > 0) {
    print $str[0];
    for ($i = 1; $i < strlen($str); $i++) {
      print $separator . $str[$i];
    }
  }
}
```

```
print_separated("hello");        # h, e, l, l, o
print_separated("hello", "-");   # h-e-l-l-o
```

- if no value is passed, the default will be used (defaults must come last)

Reading/writing an entire file

```php
# reverse a file
$text = file_get_contents("poem.txt");
$text = strrev($text);
file_put_contents("poem.txt", $text);
```

- `file_get_contents` returns entire contents of a file as a string
  - if the file doesn't exist, you will get a warning and an empty return string
- `file_put_contents` writes a string into a file, replacing its old contents
  - if the file doesn't exist, it will be created

Appending to a file

```php
# add a line to a file
$new_text = "P.S. ILY, GTG TTYL!~";
file_put_contents("poem.txt", $new_text, FILE_APPEND);
```

old contents	new contents
Roses are red, Violets are blue. All my base, Are belong to you.	Roses are red, Violets are blue. All my base, Are belong to you. P.S. ILY, GTG TTYL!~

- `file_put_contents` can be called with an optional third parameter to append (add to the end) rather than overwrite

Splitting/joining strings

```
$array = explode(delimiter, string);
$string = implode(delimiter, array);
```
PHP

```
$s  = "CSE 190 M";
$a  = explode(" ", $s);     # ("CSE", "190", "M")
$s2 = implode("...", $a);   # "CSE...190...M"
```
PHP

- `explode` and `implode` convert between strings and arrays
- for more complex string splitting, you can use **regular expressions** (later)

Example with explode

contents of input file `names.txt`

```
Martin D Stepp
Jessica K Miller
Victoria R Kirst
```

```
foreach (file("names.txt") as $name) {
  list($first, $mid, $last) = explode(" ", $name);
  ?>
  <p> author: <?= $last ?>, <?= $first ?> </p>
  <?php
}
```
PHP

```
author: Stepp, Marty

author: Miller, Jessica

author: Kirst, Victoria
```
output

The `htmlspecialchars` function

`htmlspecialchars`	returns an HTML-escaped version of a string

- text that comes from files / user input might contain <, >, &, etc.
- we could manually write code to strip out these characters
- better idea: allow them, but **escape** them

```
$text = "<p>hi 2 u & me</p>";
$text = htmlspecialchars($text);   # "&lt;p&gt;hi 2 u &amp; me&lt;/p&gt;"
```

PHP

5.3: Embedded PHP

Printing HTML tags in PHP = bad style

```
<?php
print "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.1//EN\"\n";
print " \"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd\">\n";
print "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n";
print "  <head>\n";
print "    <title>Geneva's web page</title>\n";
...
for ($i = 1; $i <= 10; $i++) {
  print "<p> I can count to $i! </p>\n";
}
?>
```

- printing HTML tags with `print` statements is bad style and error-prone:
  - must quote the HTML and escape special characters, e.g. `\"`
- but without `print`, how do we insert dynamic content into the page?

PHP expression blocks (5.3.2)

```
<?= expression ?>
```
PHP

```
<h2> The answer is <?= 6 * 7 ?> </h2>
```
PHP

The answer is 42

output

- **PHP expression block**: evaluates and embeds an expression's value into HTML
- `<?=` *`expr`* `?>` is equivalent to `<?php print` *`expr`*`; ?>`

Expression block example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>CSE 190 M: Embedded PHP</title></head>
  <body>
    <?php
    for ($i = 99; $i >= 1; $i--) {
      ?>
      <p> <?= $i ?> bottles of beer on the wall, <br />
          <?= $i ?> bottles of beer. <br />
          Take one down, pass it around, <br />
          <?= $i - 1 ?> bottles of beer on the wall. </p>
      <?php
    }
    ?>
  </body>
</html>
```

Common errors: unclosed braces, missing = sign

```
  <body>
    <p>Watch how high I can count:
      <?php
      for ($i = 1; $i <= 10; $i++) {
        ?>
        <? $i ?>
    </p>
  </body>
</html>
```

PHP

- `</body>` and `</html>` above are inside the `for` loop, which is never closed
- if you forget to close your braces, you'll see an error about 'unexpected `$end`'
- if you forget `=` in `<?=`, the expression does not produce any output

Complex expression blocks

```php
<body>
  <?php
  for ($i = 1; $i <= 3; $i++) {
    ?>
    <h<?= $i ?>>This is a level <?= $i ?> heading.</h<?= $i ?>>
    <?php
  }
  ?>
</body>
```

PHP

```
This is a level 1 heading.

This is a level 2 heading.

This is a level 3 heading.
```

output

- expression blocks can even go inside HTML tags and attributes

Web Programming Step by Step

Lecture 8
Embedded PHP
Reading: 5.3 - 5.5

Functions; More File I/O

- 5.1: Server-Side Basics
- 5.2: PHP Basic Syntax
- 5.3: Embedded PHP
- **5.4: Advanced PHP Syntax**

Functions (5.4.1)

```
function name(parameterName, ..., parameterName) {
  statements;
}
```

```
function bmi($weight, $height) {
  $result = 703 * $weight / $height / $height;
  return $result;
}
```

- parameter types and return types are not written
- a function with no `return` statements is implicitly "void"
- can be declared in any PHP block, at start/end/middle of code

Calling functions

```
name(expression, ..., expression);
```

```
$w = 163;  # pounds
$h = 70;   # inches
$my_bmi = bmi($w, $h);
```

- if the wrong number of parameters are passed, it's an error

Variable scope: global and local vars

```php
$school = "UW";                  # global
...

function downgrade() {
  global $school;
  $suffix = "Tacoma";            # local

  $school = "$school $suffix";
  print "$school\n";
}
```

- variables declared in a function are **local** to that function; others are **global**
- if a function wants to use a global variable, it must have a `global` statement
  - but don't abuse this; mostly you should use parameters

Default parameter values

```
function name(parameterName = value, ..., parameterName = value) {
  statements;
}
```

```
function print_separated($str, $separator = ", ") {
  if (strlen($str) > 0) {
    print $str[0];
    for ($i = 1; $i < strlen($str); $i++) {
      print $separator . $str[$i];
    }
  }
}
```

```
print_separated("hello");        # h, e, l, l, o
print_separated("hello", "-");   # h-e-l-l-o
```

- if no value is passed, the default will be used (defaults must come last)

Reading/writing an entire file

```php
# reverse a file
$text = file_get_contents("poem.txt");
$text = strrev($text);
file_put_contents("poem.txt", $text);
```

- `file_get_contents` returns entire contents of a file as a string
  - if the file doesn't exist, you will get a warning and an empty return string
- `file_put_contents` writes a string into a file, replacing its old contents
  - if the file doesn't exist, it will be created

Appending to a file

```php
# add a line to a file
$new_text = "P.S. ILY, GTG TTYL!~";
file_put_contents("poem.txt", $new_text, FILE_APPEND);
```

old contents	new contents
Roses are red, Violets are blue. All my base, Are belong to you.	Roses are red, Violets are blue. All my base, Are belong to you. P.S. ILY, GTG TTYL!~

- `file_put_contents` can be called with an optional third parameter to append (add to the end) rather than overwrite

Splitting/joining strings

```
$array = explode(delimiter, string);
$string = implode(delimiter, array);
```
PHP

```
$s  = "CSE 190 M";
$a  = explode(" ", $s);      # ("CSE", "190", "M")
$s2 = implode("...", $a);    # "CSE...190...M"
```
PHP

- `explode` and `implode` convert between strings and arrays
- for more complex string splitting, you can use **regular expressions** (later)

Example with explode

contents of input file `names.txt`

```
Martin D Stepp
Jessica K Miller
Victoria R Kirst
```

```
foreach (file("names.txt") as $name) {
  list($first, $mid, $last) = explode(" ", $name);
  ?>
  <p> author: <?= $last ?>, <?= $first ?> </p>
  <?php
}
```
PHP

```
author: Stepp, Marty

author: Miller, Jessica

author: Kirst, Victoria
```
output

The `htmlspecialchars` function

`htmlspecialchars`	returns an HTML-escaped version of a string

- text that comes from files / user input might contain <, >, &, etc.
- we could manually write code to strip out these characters
- better idea: allow them, but **escape** them

```php
$text = "<p>hi 2 u & me</p>";
$text = htmlspecialchars($text);   # "&lt;p&gt;hi 2 u &amp; me&lt;/p&gt;"
```

PHP

5.3: Embedded PHP

Printing HTML tags in PHP = bad style

```
<?php
print "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.1//EN\"\n";
print " \"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd\">\n";
print "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n";
print "  <head>\n";
print "    <title>Geneva's web page</title>\n";
...
for ($i = 1; $i <= 10; $i++) {
  print "<p> I can count to $i! </p>\n";
}
?>
```

- printing HTML tags with `print` statements is bad style and error-prone:
    - must quote the HTML and escape special characters, e.g. `\"`
- but without `print`, how do we insert dynamic content into the page?

PHP expression blocks (5.3.2)

```
<?= expression ?>
```
PHP

```
<h2> The answer is <?= 6 * 7 ?> </h2>
```
PHP

The answer is 42

output

- **PHP expression block**: evaluates and embeds an expression's value into HTML
- `<?=` *`expr`* `?>` is equivalent to `<?php print` *`expr`*`; ?>`

Expression block example

```php
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>CSE 190 M: Embedded PHP</title></head>
  <body>
    <?php
    for ($i = 99; $i >= 1; $i--) {
      ?>
      <p> <?= $i ?> bottles of beer on the wall, <br />
          <?= $i ?> bottles of beer. <br />
          Take one down, pass it around, <br />
          <?= $i - 1 ?> bottles of beer on the wall. </p>
      <?php
    }
    ?>
  </body>
</html>
```

PHP

Common errors: unclosed braces, missing = sign

```
  <body>
    <p>Watch how high I can count:
      <?php
      for ($i = 1; $i <= 10; $i++) {
        ?>
        <? $i ?>
    </p>
  </body>
</html>
```

PHP

- `</body>` and `</html>` above are inside the `for` loop, which is never closed
- if you forget to close your braces, you'll see an error about 'unexpected `$end`'
- if you forget `=` in `<?=`, the expression does not produce any output

Complex expression blocks

```php
<body>
  <?php
  for ($i = 1; $i <= 3; $i++) {
    ?>
    <h<?= $i ?>>This is a level <?= $i ?> heading.</h<?= $i ?>>
    <?php
  }
  ?>
</body>
```

PHP

This is a level 1 heading.

This is a level 2 heading.

This is a level 3 heading.

output

- expression blocks can even go inside HTML tags and attributes

POLI
[TECH>
NIKA

Czestochowa
University
of Technology

[@>

Faculty of Computer Science
and Artificial Intelligence