Lecture 7
PHP Syntax
Reading: 5.2 - 5.4

Except where otherwise noted, the contents of this presentation are Copyright 2009 Marty Stepp and Jessica Miller.

5.2: PHP Basic Syntax

- 5.1: Server-Side Basics
- **5.2: PHP Basic Syntax**
- 5.3: Embedded PHP
- 5.4: Advanced PHP Syntax
- 6.1: Parameterized Pages
PHP syntax template

**HTML content**

```php
<?php
    PHP code
?>
```

**HTML content**

```php
<?php
    PHP code
?>
```

**HTML content**

```php
<?php
    PHP code
?>
```

• 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));
```

<table>
<thead>
<tr>
<th>abs</th>
<th>ceil</th>
<th>cos</th>
<th>floor</th>
<th>log</th>
<th>log10</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>pow</td>
<td>rand</td>
<td>round</td>
<td>sin</td>
<td>sqrt</td>
<td>tan</td>
</tr>
</tbody>
</table>

math functions

<table>
<thead>
<tr>
<th>M_PI</th>
<th>M_E</th>
<th>M_LN2</th>
</tr>
</thead>
</table>

math constants

• the syntax for method calls, parameters, returns is the same as Java
**int and float types**

```php
$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)**

```php
$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

```php
# 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"
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Java Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>strlen</td>
<td>length</td>
</tr>
<tr>
<td>strpos</td>
<td>indexOf</td>
</tr>
<tr>
<td>substr</td>
<td>substring</td>
</tr>
<tr>
<td>strToLower, strToUpper</td>
<td>toLowerCase, toUpperCase</td>
</tr>
<tr>
<td>trim</td>
<td>trim</td>
</tr>
<tr>
<td>explode, implode</td>
<td>split, join</td>
</tr>
<tr>
<td>strcmp</td>
<td>compareTo</td>
</tr>
</tbody>
</table>

if/else statement

```php
if (condition) {
    statements;
} elseif (condition) {
    statements;
} else {
    statements;
}
```

- NOTE: although elseif keyword is much more common, else if is also supported
while loop (same as Java)

```
while (condition) {
    statements;
}
```

```
do {
    statements;
} while (condition);
```

- break and continue keywords also behave as in Java

bool (Boolean) type (5.2.8)

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

- 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
$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)**

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

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

```php
$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**

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

**Array function example**

```
$tas = array("MD", "BH", "KK", "HM", "JP");
for ($i = 0; $i < count($tas); $i++) {
    $tas[$i] = strtolower($tas[$i]);
}
$morgan = array_shift($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);
$best = array_slice($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)

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

```php
$stooges = array("Larry", "Moe", "Curly", "Shemp");
for ($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

5.3: Embedded PHP

- 5.1: Server-Side Basics
- 5.2: PHP Basic Syntax
- **5.3: Embedded PHP**
- 5.4: Advanced PHP Syntax
- 6.1: Parameterized Pages
printing HTML tags in PHP = bad style

- printing HTML tags with print statements is bad style and error-prone:
  - must quote the HTML and escape special characters, e.g. "
  - best PHP style is to minimize print/echo statements in embedded PHP code
- but without print, how do we insert dynamic content into the page?

PHP expression blocks (5.3.2)

- **PHP expression block**: a small piece of PHP that evaluates and embeds an expression's value into HTML
  - <?= expression ?> is equivalent to:
    - <?php print expression; ?>
  - useful for embedding a small amount of PHP (a variable's or expression's value) in a large block of HTML without having to switch to "PHP-mode"
Expression block example

```html
<!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

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

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

This is a level 1 heading.

This is a level 2 heading.

This is a level 3 heading.

- expression blocks can even go inside HTML tags and attributes