Introduction to Web Programming

Lecture 7: Embedded PHP

7.1: More PHP Syntax

`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

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

- zero-based indexing using bracket notation
- string concatenation operator is `.` (period), not `+`
  - `5 + "2 turtle doves"` produces `7`
  - `5 . "2 turtle doves"` produces `"52 turtle doves"`
- can be specified with `""` or `''`

Interpreted strings

```
$age = 16;
print "You are " . $age . " years old.\n";
print "You are $age years old.\n";    # You are 16 years old.
```

- strings inside " " are **interpreted**
  - variables that appear inside them will have their values inserted into the string
- strings inside ' ' are *not* interpreted:

```
print 'You are $age years old.\n';    # You are $age years old.\n
```

- if necessary to avoid ambiguity, can enclose variable in {}:

```
print "Today is your $ageth birthday.\n";    # $ageth not found
print "Today is your {$age}th birthday.\n";
```

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"
```

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

bool (Boolean) type

```
$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

Math operations

```
$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

NULL

```
$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

```
$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

```
$tas = array("MD", "BH", "KK", "HM", "JP");
for ($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

```
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

7.2: Embedded PHP

- 7.1: More PHP Syntax
- **7.2: Embedded PHP**

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"

Printing HTML tags in PHP = bad style

```
<?php
print "<!DOCTYPE html>\n";
print "<html>\n";
print "  <head>\n";
print "    <title>Geneva's web page</title>\n";
...
for ($i = 1; $i <= 10; $i++) {
  print "<p class=\"count\"> 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

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

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

```
The answer is 42
```

- **PHP expression block**: evaluates and embeds an expression's value into HTML
- `<?= expr ?>` is equivalent to `<?php print expr; ?>`
- not always available (must be enabled in the configuration file `php.ini`)

Expression block example

```
<!DOCTYPE html>
<html>
  <head><title>IWP: 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>
```

- `</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

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