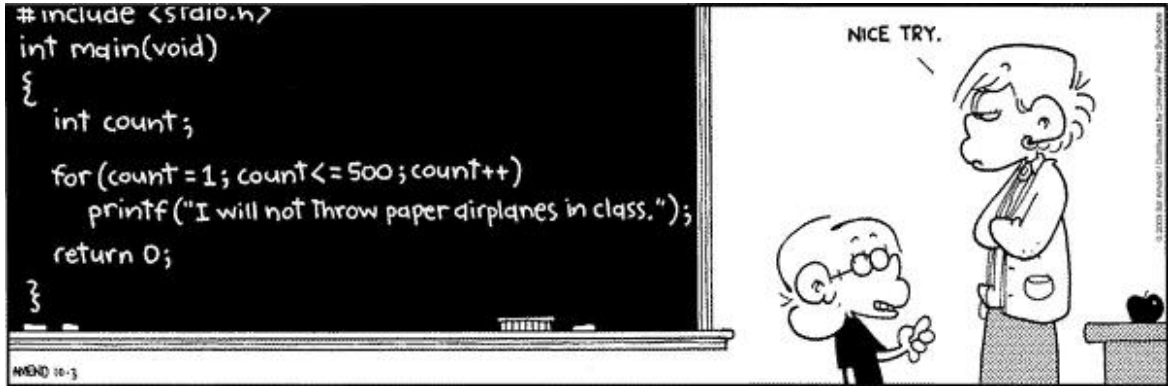
CSE 154



LECTURE 15: *EMBEDDED PHP*

PHP syntax template

```
HTML content

  <?php
  PHP code
  ?>

HTML content

  <?php
  PHP code
  ?>

HTML content ...      PHP
```

- any contents of a .php file between <?php and ?> are executed as PHP code
- all other contents are output as pure HTML

Arrays

```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

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) {
   ...
}
```
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

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

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)

Printing HTML tags in PHP = bad style

```php
<?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";
}
?>
```

PHP

- 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 ?>
```
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>
<html>
  <head><title>CSE 154: 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

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

PHP

output

- expression blocks can even go inside HTML tags and attributes

Functions

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

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

- 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);
```
PHP

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

- 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 = "(Wisconsin)";         # 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;
}
```
PHP

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

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

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