CS3226

WEB PROGRAMMING AND APPLICATIONS

LECTURE 07 - PHP

DR STEVEN HALIM

STEVENHALIM@GMAIL.COM

For 'not newbies': Try this PHP quiz @ W3Schools first

OUTLINE

WHY SERVER-SIDE PROGRAMMING?

So far, we have seen various *client-side* technologies:

1. Week02: HTML5 & CSS3
2. Week03: JS (jQuery)
3. Week04: A Front-End framework (Bootstrap)

However, client-side programming *alone* is not enough for a full-fledged web application because...

1. Important reason: We may want to serve *dynamic* webpage that is created based on *previous* client inputs (not *just* during the current session) that have been stored in the server's database (or other technology)
2. We do not, and ethically cannot, store our client data in some other client's computer
3. We may not want some of our *non-open-source code* to be accessible in public like in client-side programming
4. Server machines are usually (much) more powerful than the client machine and some computational tasks need (extremely) large processing power which client machine usually unable to handle

PHP: HYPERTEXT PREPROCESSOR

PHP is yet another programming language in this CS3226

It is one of the most popular *server-side scripting* language in the world (and open source a.k.a. free)

This time we will write programs on server-side and you will need a CS3226-1 VM account* to test these programs

Or, you can install things like (free) XAMPP, WAMPServer, etc on your own computer to run PHP locally

HELLO WORLD (FROM PHP)

Let's write our very first PHP code named as `'hello.php'`:

```
<?php // this is the signal of the start of PHP section
echo 'Hello World, this is my first PHP code'; // this is a comment
// the line below signal the end of PHP section
?>
```

For this lecture, Steven uploads the PHP code to his VisuAlgo server

Try accessing http://visualgo.net/cs3226/hello.php; Is it OK? Then check the page source! Is it an HTML(5) document?

CS1010 AGAIN... WITH PHP

Programming Methodology, but using PHP:

1. How to declare variables, that $ symbol...
2. Sequencing commands, the semicolon ; symbol
3. Control flow: Selection, if-else, (cond ? true : false), switch
4. Control flow: Repetition, for, foreach, while, do-while
5. Declaring function: function and recursive function

The live demo will be conducted not-in-browser but directly on (VisuAlgo) server

DEMO.PHP (1)

```
<!DOCTYPE html> <!--We can mix HTML, CSS, JS commands inside PHP file
<html lang="en">
<head><title>PHP demo</title></head>
<body><?php
$w = 1; $x = 2; $y = 3; $z = 4; // variable names starts with a $ sig
define("FOUR", 4); // this is PHP way to declare a constant
eChO "<p>variables demo<br>\n";
ecHo ($w/$x + $y*$z) . "<br>\n"; // echo 12.5; frequently used comman
Echo ($x > $y) . "<br>\n"; // false is not echoed; "." is a concat sy
ECHO ($x < $y && $z == FOUR) . "<br></p>\n"; // echo 1 (true)
// PHP commands are NOT case sensitive but variable names are...
echo "What is this: [$notexistent]?<br>\n"; // nothing is printed
$w = 'changed to string'; // PHP is not a strongly typed language, li
echo $w . "<br>\n";
print ($z%3); // print is similar to echo, % is modulo, prints out 1
```

DEMO.PHP (2)

```
echo "<p>if-else and (cond ? true : false) demo<br>\n";
if ($x > $y)
  echo "x is bigger than y<br>\n";
else
  echo "x is smaller than y<br>\n";
echo "x is " . ($x > $y ? "bigger" : "smaller") . " than y<br></p>\n"

echo "<p>switch demo<br>\n";
switch ($x) {
  case 1: echo "x is 1<br>\n"; break;
  case 2: echo "x is 2<br>\n"; break;
  default: echo "x is not 1 or 2<br>\n"; break;
}
echo "</p>\n";
```

DEMO.PHP (3)

```php
echo "<p>for loop demo<br>\n";
for ($i = $x; $i <= $z; $i++) // $x = 2, $z = 4, loop from 2 to 4
  echo "i is " . $i . "<br>\n";
  // echo "i is $i<br>\n"; // is also possible, notice double quotes

echo "</p>\n<p>while loop demo<br>\n";
$i = 0; // $y = 3
while ($i < $y) // 0, 1, 2 and then stop
  echo "i is " . ($i++) . "<br>\n";

echo "</p>\n<p>do-while loop demo<br>\n";
$i = 0; // $y = 3 (different behavior with above if $y = 0)
do
  echo "i is " . ($i++) . "<br>\n";
while ($i < $y); // 0, 1, 2 and then stop (notice the ;)
echo "</p>\n";
```

DEMO.PHP (4)

```php
echo "<p>more complex control flow demo<br>\n";
$a = array(2, 1, 7, 8, 3); // we will discuss array in more details s
for ($i = $sum = 0; $i < count($a); $i++) // count: to count array le
  if ($a[$i]%2 == 0) continue; // an even number, skip this iteratio
  else if ($sum > 7) break; // stop the associated loop
  else                  $sum += $a[$i];
echo "the sum is $sum<br>\n";

$sum = 0;
foreach ($a as $key => $value) { // foreach loop
  if ($sum > 7) break;
  $sum += $value%2 == 0 ? 0 : $value;
}
echo "the sum is $sum<br></p>\n";
```

DEMO.PHP (5)

```php
echo "<p>function demo<br>\n";
require_once("sum.php"); // this is like C/C++ #include (next slide)

$a = array(2, 1, 7, 8, 3);
echo "sum: " . sum($a) . "<br>\n"; // 21

function fib($n = 7) { // recursive function, default argument $n = 7
  if ($n <= 1) return $n;
  else return fib($n-1) + fib($n-2);
}

echo "fib(6) = " . "<br>\n"; // 8
echo "fib() = " . fib() . "<br></p>\n"; // default argument, fib(7) =
```

SUM.PHP

```php
<?php // an external file that is included in the previous slide
function sum($a) {
  $ans = 0;
  foreach ($a as $val)
    $ans += $val;
  return $ans;
}
?>
```

DEMO.PHP (6)

```php
// PHP 1D array demonstration
$C = array(13, 1, 12, 5, 7);
array_splice($C, 2, 2, array(8, 4, 6)); // replace [12, 5] with [8,
$l = count($C); // length of array is now 6: [13, 1, 8, 4, 6, 7]
echo $C[5] . "<br>\n"; // 7
sort($C); // sort ascending, already compare by value
echo "sorted: ";
foreach ($C as $val) echo $val . " "; // manual
echo "<br>\n";
echo "compare with: " . implode(" ", $C) . "<br>\n"; // use implode
$D = array("N"=>2, "M"=>3); // PHP associative array ~ an ordered ma
echo "Using var_dump:<br>\n";
var_dump($D);
echo "<br>\n";
```

DEMO.PHP (7)

```
// PHP 2D array demonstration
$G = array(
  array(1, 2),
  array(0),
  array(0)
);
for ($i = 0; $i < count($G); $i++) {
  echo "row $i: ";
  for ($j = 0; $j < count($G[$i]); $j++)
    echo $G[$i][$j] . " ";
  echo "<br>\n";
}
echo "Using var_dump:<br>\n";
var_dump($G);
echo "<br>\n";
```

DEMO.PHP (8)

```php
// PHP string manipulation demonstration
$str1 = 'steven';
echo strlen($str1) . "<br>\n"; // 6
echo $str1[1] . $str1[3] . "<br>\n"; // 'tv', 0-based indexing
$str2 = 'seven';
echo "strcmp: " . strcmp($str1, $str2) . "<br>\n"; // positive value
echo "eve: " . strpos($str1, "eve") . "<br>\n"; // index 2
echo "ev3: " . strpos($str1, "ev3") . "<br>\n"; // false/not found
$seven = 7;
echo "$seven<br>\n"; // will output: 7
echo '$seven<br>\n'; // will output: $seven<br>\n, notice the differe
?>
</body>
</html>
```

DEBUGGING PHP CODE (1)

Now this can be painful...

When the client requests for a PHP file, the server will execute the PH(Pre-processor) code (pre-process it) before returning an HTML file (although we still see .php in the address bar of our client browser)

If there is a *warning* in the PHP script, PHP will continue but *logs* the warning message in the web server `error.log` file (it is in folder `/var/log/httpd` on our CS3226-1 VM, your account has read access but no write access)

DEBUGGING PHP CODE (2)

For example, the script below will add a warning message at the tail of `error.log`

```php
<?php $x = 1/0; // division by zero
?>
```

```
$ tail -1 /var/log/httpd/error.log # show the last line in error.log
[Mon Feb 01 16:07:36.676355 2016] [:error] [pid ?????] [client ???.???.???.
???:?????] PHP Warning:  Division by zero in /home/steven/public_html/error
.php on line 1
```

Occassionally check the log file of your web server for such (silent) warning like this because if your web server is not configured properly, you may quickly run out of disk space due to accummulated warning messages

DEBUGGING PHP CODE (3)

If there is an *unrecoverable error*, the server cannot execute PHP code further, sends HTTP status 500 (Internal Server Error), (usually) shows a blank page (or partially rendered page), and also *logs* the error message in the error.log file

Such error usually cannot be handled even with this set_error_handler function

DEBUGGING PHP CODE (4)

For example, the script below will add an error message at the tail of `error.log`

```
<?php echoes "Hello World"; // unknown PHP command 'echoes'
?>
```

```
$ tail -1 /var/log/httpd/error.log # change -1 to -n to get last n lines
[Mon Feb 01 16:11:26.018589 2016] [:error] [pid ?????] [client ???.???.???.
???:?????] PHP Parse error:  syntax error, unexpected '"Hello World"' (T_CO
NSTANT_ENCAPSED_STRING) in /home/steven/public_html/error.php on line 1
```

This time, the web programmer usually reacts faster as his/her PHP code does not work

Debugging server-side PHP code is indeed (much) harder than debugging client-side JavaScript code

LINKING
CLIENT-AND-SERVER

PHP SUPERGLOBALS VARIABLES (1)

So far, what we have learned in this PHP lecture is similar with the client-side technology: JavaScript

We can use PHP to make the server reacts to *client's inputs* rather than just serving the same static webpage

PHP can process the information sent by client to server via HTTP GET* or HTTP POST* requests

To do this in PHP is simple, we access its superglobal arrays $_GET (useful in some cases, e.g. in http://nusmods.com) or $_POST (recommended method in most cases)

PHP SUPERGLOBALS VARIABLES (2)

There are a few other PHP superglobal arrays, e.g. `$_REQUEST` (combination of `$_GET`, `$_POST`, and `$_COOKIE`), `$_SESSION` (will be discussed soon), and some optional ones: `$_SERVER`, `$_FILES`, `$_ENV`, `$_COOKIE`, `$GLOBALS`

During the HTML5 lecture, we briefly discussed about the HTML5 forms, mainly about the HTML5 tags to display the form elements, but we *have not* discuss on how to actually process the data collected from user and we will do so now

CLOSING THE LOOP

A simple form.html and the corresponding form.php

```
<--client side, a simple form ("form.html")-->
<form method="post" action="form.php">
<p>What is 7 + 3?<br></p>
<input type="radio" name="mcq" value="9">A. 9
<input type="radio" name="mcq" value="10">B. 10
<input type="radio" name="mcq" value="11">C. 11
<input type="submit" name="submit" value="Answer">
</form>
```

```
<--server side, form.php-->
<p>You selected: <?php $ans = $_POST["mcq"]; echo $ans;?><br>
Correct: <?php echo ($ans == 10) ? "Y" : "N"; ?><br></p>
```

Not shown in the simple example above: Always validate/sanitize form input on server-side too even if you have done so in client-side using either HTML5 form elements' settings or JavaScript (details in security lecture)

PHP SESSION

In JavaScript lecture, we have seen how to add "state" to the stateless HTTP protocol in client-side: using HTML5 `localStorage` (we abandoned HTTP cookies)

We can also add "state" to the stateless HTTP protocol in server-side, using PHP session control

COUNT.PHP AND RESET.PHP

```php
<?php session_start(); must be the first line of your PHP script
$elapsed = 0;
if (empty($_SESSION['count'])) { /* or use !isset */
  $_SESSION['count'] = 1;
  $_SESSION['visittime'] = time();
}
else {
  $_SESSION['count']++;
  $elapsed = time()-$_SESSION['visittime'];
  $_SESSION['visittime'] = time();
}
echo "You have visited this page " . $_SESSION['count'] . " times<br>
echo session_id() . ", your last visit was " . $elapsed . "s ago<br>\
/* this is count.php */ ?>
```

```php
<?php session_start(); // must still use this first
session_destroy();
echo "Try visiting count.php again<br>\n";
/* this is reset.php */ ?>
```

More about session control in the second half of the class

AJAX, REVISITED

Several ways to execute PHP script @ web-server:

1. Specify the direct URL of the PHP script
2. Use HTML5 form submit feature
3. Use AJAX (with jQuery): Call a PHP script with some input/data, asynchronously receive the output/processed data, and update the client-side with this new data

We will likely pass (complex) data between client and server as JSON, so please study json_encode (convert PHP associative array into JSON to be read by client-side JS) and json_decode (take in JSON from client-side JS and decode it into PHP object that can be cast as associative array)

DATABASE ACCESS

Usually we want to store user's (form) input in a *persistent* storage and a (relational) database is a good solution

Good News: PHP has built-in support to run (My)SQL queries

In the next Database lecture, we will learn on how to use database in our web application

Things will get much more exciting after this

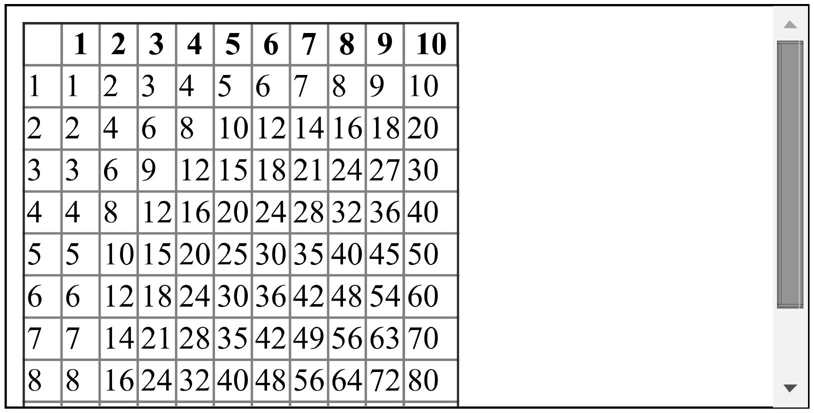
CHALLENGES

CHALLENGE 1: MULTTABLE.PHP

Same as with the challenge in JS lecture but now in PHP

Can I see the php code?

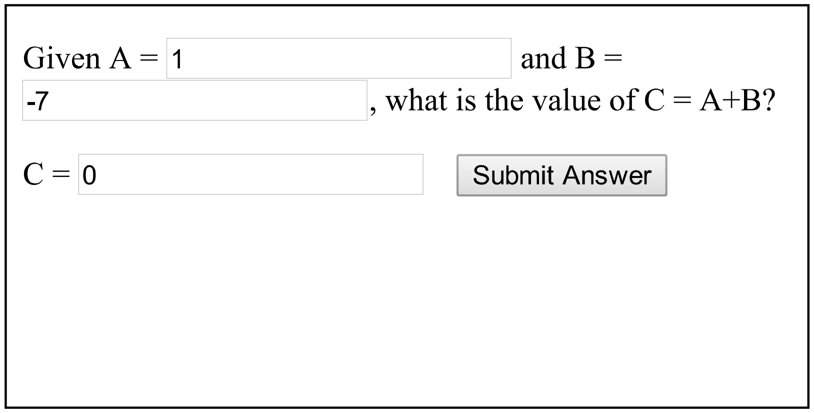
Answer



	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80

CHALLENGE 2: MINIVISUALGO.PHP*

Same as with the challenge in JS lecture but now in PHP



Given A = 1 and B = -7, what is the value of C = A+B?

C = 0

Can I see the php code?

THE END

WHAT HAVE NOT BEEN DISCUSSED?

- Input sanitizing functions (deferred to security lecture)
- PHP `header` function (deferred to security lecture)
- PHP `exit` function (defered to SQL lecture)
- PHP `mail` function (defered to Web Server lecture)
- PHP `$_FILES` superglobals
- PHP framework, e.g. Symfony
- Object-Oriented Programming (OOP) in PHP