PHP: Functions, regular expressions

INFO/CS 2300:
Intermediate Web Design and Programming

Logistics

Office hour calendar is now in good shape
http://info230.cs.cornell.edu/office_hours/

HW0 graded – check CMS to be sure

Project 1 due Tuesday 5 PM

No Frameworks

Running your PHP code

- Upload to info2300 server by SFTP then browse
  - http://info2300.cs.cornell.edu/users/username/www/filename
- Installing a webserver and PHP on your computer
  - A one-click install: XAMPP; see http://www.apachefriends.org/index.html
  - Note: the server is running PHP 5.3 but planning an update during the semester

Mini Crash Courses

CSS - this Friday afternoon

XAMPP setup – next week

Debugging – after Feb. break

– details will be announced on Piazza

Functions

Function basics

PHP has lots of functions that do lots of things. The basic form:

```
function(arg1, arg2, … )
```

We've already seen some functions:

```
print( $name );
$count = count( $arrayname );
```

Some other useful functions

isset($variable): returns true if the variable has been assigned a value, false otherwise

empty($variable): broader than isset. equivalent to
! isset($variable) || $variable == false

form.php

```
<?php
    if ( ! empty( $_POST[ "username" ] ) ) {
        print( "<p>Welcome, " . $_POST['username'] . "! </p>" );
    } else {
        ?>
            <form method="post" action="form.php">
                <p>What is your name?
                    <input type="text" name="username">
                </p>
                <input type="submit" value="Click to submit">
            </form>

        <?php
    }
?>
```

Example functions

mail($to, $subject, $message)

sends email to email address $to with subject $subject and body $message

…and many, many more

Search for what you want such as:

PHP function send mail

Note: the INFO 2300 server is running PHP 5.3.

Hoping to update during the semester.

Defining your own functions

Example

Define your own functions for reuse and legibility.

Function name Argument(s)

```
function makeSubmitButton($name) {
    print( "<input type='submit' value='$name'>" );
}

makeSubmitButton(“Send message”);
```

Returning values

Your functions can return values as well.

```
function increment($x) {
   $x++;
   return $x;
}

$y = 0;
$z = increment($y);                $z = 1
```

Functions on strings

trim

trim($string) – returns a string with whitespaces removed from beginning and end

E.g.

```
$name = '  Spongebob Squarepants  ';
$newname = trim( $name );
print(“$newname$newname”);
```

```
      Spongebob SquarepantsSpongebob Squarepants
```

Strings and arrays

explode($separator, $string) – returns an array containing parts of $string that were joined by $separator.

implode($glue, $array) – returns a string containing parts of $array joined by $glue.

Example: explode / implode

```
$date = “1/28/2015”;
$myarray = explode(‘/’, $date);
print(“Month is $myarray[0],
        Day is $myarray[1],
        Year is $myarray[2]”);

$newdate = implode(‘-’,$myarray);

print( $newdate );
```

Array(‘1’, ‘28’, ‘2015’)

Month is 1,
Day is 28,
Year is 2015

1-28-2015

Pattern matching

preg_match($pattern, $string) – returns true if the $pattern appears in the $string

$pattern needs to have ‘delimiters’, usually ‘/’.

preg_match('/geb/', 'Spongebob')

returns true

Pattern replacing

preg_replace($pattern, $replacement, $subject)

returns a string in which all occurrences of $pattern in $subject are replaced by $replacement

E.g.

preg_replace(“/o/”, “aw”, “Spongebob”)

returns Spawngebawb

What is the result?

explode(“a”, “blah blah blah”)

A. array(“bl”, “h bl”, “h”)

B. “bl”

C. array(“bla”, “h bla”, “h”)

D. None of the above

D. array(“bl”, “h bl”, “h bl”, “h”)

What is the result?

```
preg_replace('/ah /', 'ow ', 'blah blah blah')
```

A. array(“blah”, “blah”, “blah”)
B. “blow blow blow”
C. “blow blow blah”
D. “blow blah blah”
E. None of the above

Answer: C – no space after the last blah

Regular expressions

http://www.phpro.org/tutorials/Introduction-to-PHP-Regex.html

Regular Expressions

With preg_match, preg_replace, and preg_split, can actually look for more complicated patterns via *regular expressions.*

Regular expressions are patterns expressed via special symbols.

Repeating and grouping

* -- means zero or more of the preceding "character"

+ -- means one or more of the preceding "character"

() – treat a group of characters as a unit

Examples

preg_match('/a*/', 'SpongeBob')	true
preg_match('/ab*/', 'SpongeBob')	false
preg_match('/(ab)+/', 'Krusty Krab')	true
preg_match('/(ab)*/', 'The Chum Bucket')	true

Start and end

^ -- matches when the following "character" starts the string

$ -- matches when the preceding "character" ends the string

```
preg_match( '/^b/', 'SpongeBob' )          false
preg_match( '/b$/', 'SpongeBob' )          true
preg_match( '/(eb)$/', 'SpongeBob' )       false
```

Or

| -- matches if either the preceding or the following "character" matches

```
preg_match('/(on)|(an)/', 'SpongeBob')
```

true

Any and character classes

. – matches any single character

[ ] – matches any single character inside the brackets (a *character class*)

```
preg_match( 'B.b', 'Bob' )                 true
preg_match( '^[Sp]', 'SpongeBob' )         true
preg_match( '^[Sp]$', 'SpongeBob' )        false
```

Character class ranges

Character classes are often given by *ranges*

[0-9] is shorthand for [0123456789]

[A-Z] matches any uppercase letter

Exercise: Netlingo translator

'brb' => 'be right back'

'cul8r' => 'see you later'

'imho' => 'in my humble opinion'

imho im aatk in 2300

```
$input = $_POST[ 'input' ];
print( "<p>Input: $input</p>" );
$result = $input;

$lingo_terms = array(
    'brb' => 'be right back',
    'cul8r' => 'see you later',
    'imho' => 'in my humble opinion',
    'im' => "I'm",
    'aak' => 'asleep at keyboard',
    'aatk' => 'always at the keyboard',
);

foreach( $lingo_terms as $index => $value ) {

    $search = "/\b$index\b/i";
    $result = preg_replace( $search, $value, $result );
}

print( "<p>Translation: $result </p><br>" );
```

Associative array
pattern is the key and
replacement is the value

/ delimiters
\b any word boundary
i case insensitive

What about server usernames?

What would we need to test usernames for the following pattern?
netidsp15

<u>Tests</u>

sm68sp15

smohlke

sm68SP15

smohlkesp15

68sp15

smsp15

Review

- PHP has many useful functions; search "PHP what I want to do"
- Define your own functions.
- Regular Expressions give you powerful pattern matching