

PHP: Functions, regular expressions

Exercise handout
on half wall

INFO/CS 2300:
Intermediate Web Design and
Programming

Enrollment

Student Center issues are fixed

Course is full – about 50 trying to get in
Janeen sending update emails today.

Please be patient

Assignments

HW0 graded – check CMS to be sure

Project 1 due Tuesday 5 PM

No Frameworks

Mini Crash Courses

localhost (setup server on your computer)

- Wed 2/3 7PM
- Fri 2/5 4PM

CSS

- Mon 2/8 6PM
- Wed 2/10 7PM

(See Piazza for more details)

Office Hours

Today's are posted on Piazza.

The rest will be posted as soon as possible

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 );
```

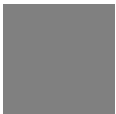
Some other useful functions

`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" ] ) ) {
    //We'll start doing this more securely next week
    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 5.6.5.



Jan. 2015

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;  
}
```



This is a wrapper.
It adds nothing to
++

```
$y = 0;
```

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

```
$z = 1
```

Variable Scope

```
function create_username($netid) {  
    //scope is from outside this function  
    global $course_id;  
  
    //Local variable scope is only inside this function  
    $user_name = $netid . $course_id;  
}
```

Why create your own?

- Simplify your code
- Keep from repeating the same code – simplifies updates

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]");
```



Array('1', '28', '2015')



Month is 1,
Day is 28,
Year is 2015

```
$newdate = implode('-', $myarray);  
print( $newdate );
```



1-28-2015

Click In

Click In

`explode("a", "blah blah blah")`

- A. `array("bl", "h bl", "h")`
- B. `"bl"`
- C. `array("bla", "h bla", "h")`
- D. None of the above

Click In

`explode("a", "blah blah blah")`

A. `array("bl", "h bl", "h")`

B. `"bl"`

C. `array("bla", "h bla", "h")`

D. None of the above

Click In

```
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

Click In

```
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

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.

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

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

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

Start and end

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

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

<code>preg_match('/^b/', 'SpongeBob')</code>	false
<code>preg_match('/b\$/', 'SpongeBob')</code>	true
<code>preg_match('/(eb)\$/', 'SpongeBob')</code>	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



<http://www.phpliveregex.com/>

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

Associative array
pattern is the key and
replacement is the value

```
$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 );  
}
```

/ delimiters
\b any word boundary
i case insensitive

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

Click In

If an exercise like this is on the server I

- A. Would still like a paper copy
- B. Don't need a paper copy
- C. Usually won't need a paper copy

What about server usernames?

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

Tests	result	Comment
sm68sp16	Yes	
smohlke	No	no number, no sp16

Tests

sm68sp15

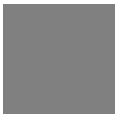
smohlke

sm68SP15

smohlkesp15

68sp15

smsp15



Tests	result	Comment
sm68sp16	Yes	
smohlke	No	no number, no sp16
sm68SP16	No	Capital SP
smohlkesp16	No	No # in netid, text too long
68sp16	No	No letters in netid
smsp16	No	No # in netid
asasas12121212sp15	No	Too long netid
as12121212sp15	No	Too many digits netID
sm68sp15	No	last year
SM68sp16	No	Capital Netid
asas12sp16	No	too many letters netid
sm68sp15a	No	Letter after sp16

$^{\wedge}[a-z]\{2,3\}[0-9]\{1,5\}sp16\$$

Review

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