Foundations to Get You Started

Beth Tucker Long

How this talk will work:

- You can ask questions at any time.
- Code will build from slide to slide, but due to space constraints, we can't show it all on one slide
- The code being shown is for teaching use only, it is not optimized in any way. It should not be used as is for a live system.

Who am I?

- Beth Tucker Long (@e3betht)
- Editor-in-Chief - php[architect] magazine
- Freelancer under Treeline Design, LLC
- Stay-at-home-mom

- User group organizer – Madison PHP

Get Involved

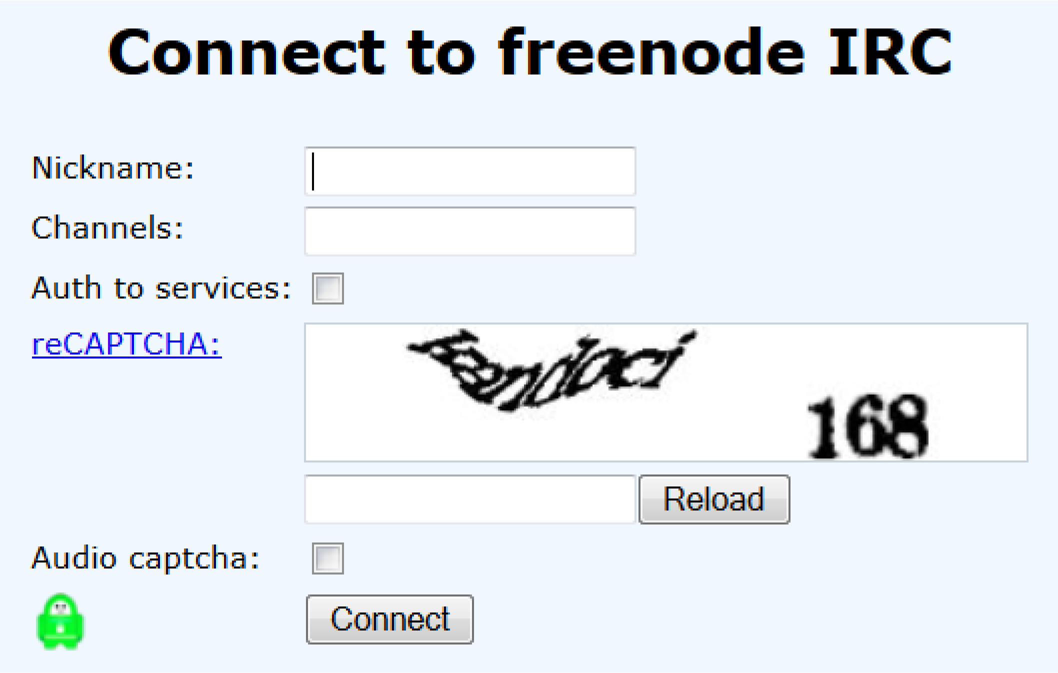
- User Groups
  - PHP.usergroups – http://php.ug
  - Meetup- http://www.meetup.com
  - Nomad PHP - http://nomadphp.com
  - php.net (right sidebar on homepage and http://php.net/cal.php)

Get Involved

- Conferences and Summits
  - Day Camp 4 Developers - http://daycamp4developers.com/
  - php[architect] Summit Series - http://summits.phparch.com/
  - ProTalk - http://protalk.me/
  - Joind.in - http://joind.in/

Get Involved and Find Help

- IRC – Freenode
  - ##php – help channel
  - #phpc – community channel
  - Web chat: https://webchat.freenode.net/



Connect to freenode IRC

Nickname:

Channels:

Auth to services: □

reCAPTCHA:

Audio captcha: □

Get Involved and Find Help

- Twitter
  - https://twitter.com/phpc
- Facebook
  - PHP Community - https://www.facebook.com/groups/4189052132/
  - php[architect] - https://www.facebook.com/phparch
- PHPDeveloper - http://phpdeveloper.org/

Online Training

- php[architect] - http://www.phparch.com/training/
- Code Academy - http://www.codecademy.com/tracks/php

Getting Started

- Start with your standard HTML
- **Opening PHP tag**
- **PHP code**
- **Closing PHP tag**
- Close out your HTML page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
    Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
    transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en" lang="en">
<head><title>Order Some Pizza</title</head>
<body>

<h1>Welcome to Joe's Pizza!</h1>

<?php

    $name = "Beth";
    echo "<p>$name's order:</p>";

?>

</body></html>
```

Start with Standard HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
  Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
  transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="en" lang="en">
<head><title>Order Some Pizza</title</head>
<body>

<h1>Welcome to Joe's Pizza!</h1>
```

Opening PHP tag

```
<?php
```

Short Opening PHP tag

```
<?
```

PHP code

```
$name = "Beth";
```

Variables:

- Preceded by a dollar sign - $
- Names may contain letters, numbers, and underscores, but must start with a letter or underscore
- Loosely-typed
- Assigned a value using the equal sign

PHP code

```
$name = "Beth";

echo "<p>$name's order:</p>";
```

Will send to the browser:

```
<p>Beth's order:</p>
```

Quoting Strings

Double-quoted, parsed and interpreted:

```
echo "<p>$name said, \"Let's Go!\"</p>";

echo 'Enter it at the C:\ prompt';
echo 'Enter it at the C:\\ prompt';
```

Single-quoted, string literal, not parsed:

```
echo '<p>$name said, "Let\'s Go!"</p>';

echo "Enter it at the C:\\ prompt";
```

Quoting Strings

Heredoc, parsed and interpreted:

```
echo <<<MYSTRING
<p>$name said, "Let's Go!"</p>
MYSTRING;
```

Nowdoc, string literal, not parsed, since PHP 5.3.0:

```
echo <<<'MYSTRING'
<p>$name said, "Let's Go!"</p>
MYSTRING;
```

Closing PHP tag - Optional

```
?>
```

Close out your HTML page

```
</body></html>
```

Echo'ing and HTML

```
echo "<p>Welcome, $name!<br />
<em>Thank you</em> for visiting $sitename
  today.</p>";
```

Echo'ing and HTML

```
<p>Welcome, <?php echo "$name"; ?>! Thank you
  for visiting <?php echo "$sitename"; ?>
  today.</p>
```

If you have PHP 5.4.0+ or the short_open_tag configuration enabled:

```
<p>Welcome, <?="$name"?>! Thank you for
  visiting <?="$sitename"?> today.</p>
```

Making Comments

```
// This is a comment

# This is a comment

/* This is also a comment
and one you are very likely
to see in code
*/
```

Good Coding Practices

- Pear Coding Standard: http://pear.php.net/manual/en/standards.php
- Pear2 Coding Standard: http://pear.php.net/manual/en/pear2cs.php
- ZF2 Coding Standard: http://framework.zend.com/wiki/display/ZFDEV2/Coding+Standards
- phpDocumentor: http://www.phpdoc.org/

Math

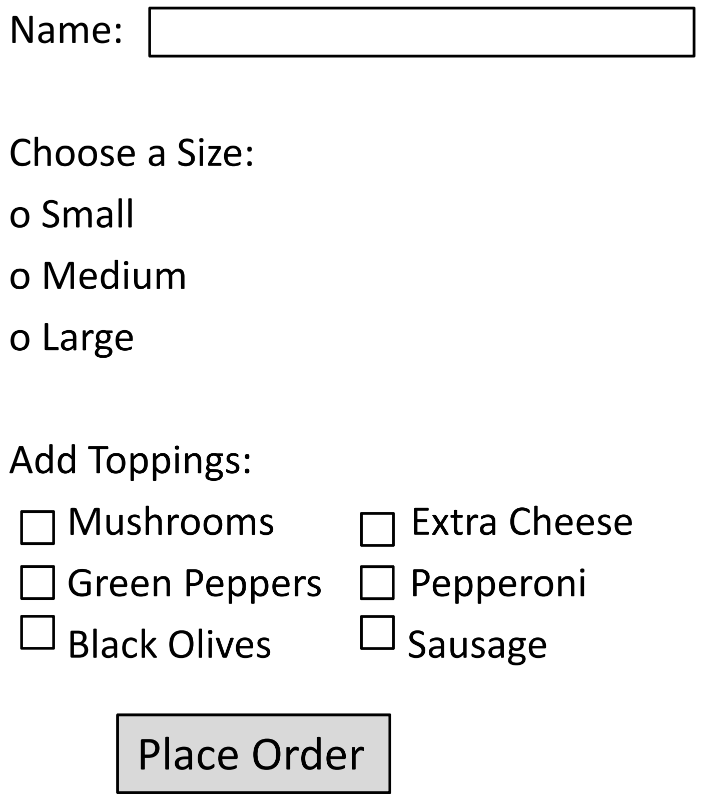
```
$a = $a + 3;
$a += 3;
$a++;

$a = $a – 3;
$a -= 3;
$a--;

$a = $a * 2;
$a *= 2;

$a = $a / 2;
$a /= 2;
```

Let's Make Something

- Order form to order a pizza
  - Customer name
  - Size of pizza
  - Allow multiple topping choices
  - Printable receipt

The Order Form

- We'll create a form that looks like this:



Name:

Choose a Size:

o Small

o Medium

o Large

Add Toppings:

□ Mushrooms □ Extra Cheese

□ Green Peppers □ Pepperoni

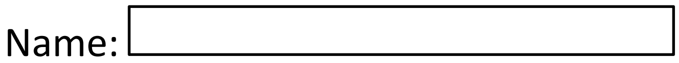
□ Black Olives □ Sausage

Place Order

Starting the Form

```
<form action="./orderPizza.php" method="POST">
```

Text Field



Name:

```
<p>Name: <input type="text" name="custName"
  maxlength="200"/></p>
```

Radio Buttons

Choose a Size:

o Small

o Medium

o Large

```
<p>Choose a Size:<br />
<input type="radio" name="pizzaSize"
  value="Small" /> Small<br />
<input type="radio" name="pizzaSize"
  value="Medium" /> Medium<br />
<input type="radio" name="pizzaSize"
  value="Large" /> Large</p>
```

Checkboxes

Add Toppings:

□ Mushrooms

□ Green Peppers

□ Black Olives

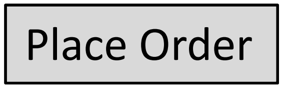
□ Extra Cheese

□ Pepperoni

□ Sausage

```
<input type="checkbox" name="Mushrooms"
  value="Yes" /> Mushrooms<br />
<input type="checkbox" name="GreenPeppers"
  value="Yes" /> Green Peppers<br />
<input type="checkbox" name="BlackOlives"
  value="Yes" /> Black Olives <br />
<input type="checkbox" name="ExtraCheese"
  value="Yes" /> Extra Cheese<br />
<input type="checkbox" name="Pepperoni"
  value="Yes" /> Pepperoni<br />
```

Input Field



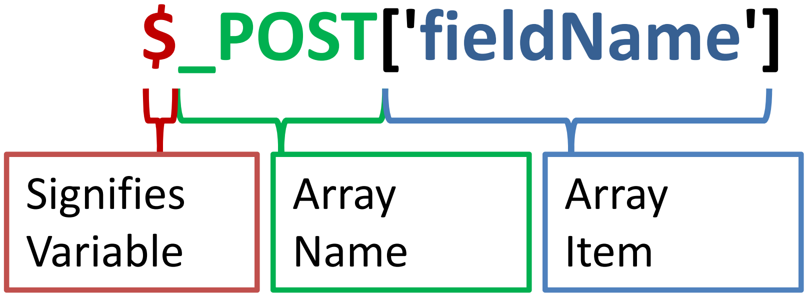
```
<input type="submit" name="pizzaStatus"
  value="Place Order" />
```

Closing the form

```
</form>
```

What We Get

- A superglobal
- POST or GET
- Everything is a string
- An array



$_POST['fieldName']

Signifies Variable

Array Name

Array Item

Two Types of Arrays

- Enumerated: $arrayName[]
  - $arrayName[0]
  - $arrayName[1]

- Associative: $arrayName[stringName]
  - $custInfo['firstName']
  - $custInfo['lastName']

Built-in Functions

```
functionName($parameter1, $parameter2)
```

Creating Arrays

- Enumerated:

```
$arrayName = array("firstValue",
"secondValue");

$arrayName[] = "thirdValue";

echo $arrayName[1];
// secondValue
```

Creating Arrays

- Associative:

```
$custInfo= array ("firstName" => "Beth",
"lastName" => "Tucker Long");

$custInfo["twitterHandle"] = "e3betht";

echo $custInfo["lastName"] ;
//  Tucker Long
```

Quoting Arrays

```
echo "First item: $_POST[0]";

{$_POST['custName']}
.$_POST['pizzaSize'].

echo "<p>This order is for
{$_POST['custName']}</p>
<p>Size: ".$_POST['pizzaSize']."</p>";
```

Nested Arrays

A nested array:

```
$_POST = array("custInfo" =>
array("firstName" => "Beth", "lastName" =>
"Tucker Long");

echo $_POST['custInfo']['lastName'];
// Tucker Long
```

What We Are Currently Getting

- $_POST['custName']
- $_POST['pizzaSize']
- $_POST['pizzaStatus']

- $_POST['Mushrooms']
- $_POST['ExtraCheese']
- $_POST['GreenPeppers']
- $_POST['Pepperoni']
- $_POST['Black Olives']
- $_POST['Sausage']

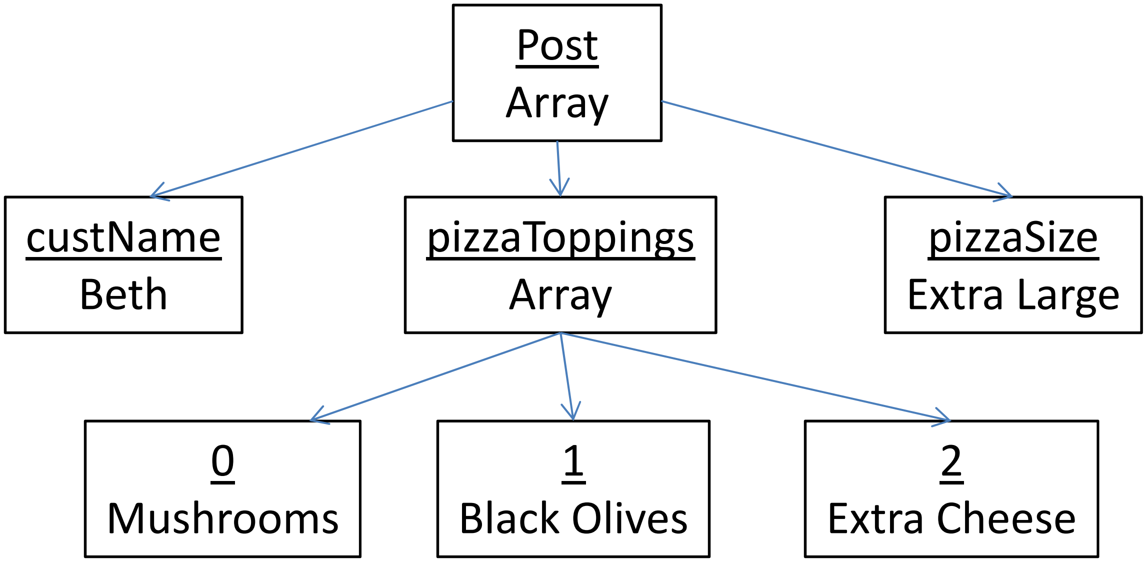
Revised: Adding an Input Array

```
<input type="checkbox" name="pizzaToppings[]"
  value="Mushrooms" /> Mushrooms<br />

<input type="checkbox" name="pizzaToppings[]"
  value="Green Peppers" /> Green Peppers<br />

<input type="checkbox" name="pizzaToppings[]"
  value="Black Olives" /> Black Olives<br />

<input type="checkbox" name="pizzaToppings[]"
  value="Extra Cheese" /> Extra Cheese<br />

<input type="checkbox" name="pizzaToppings[]"
  value="Pepperoni" /> Pepperoni<br />

<input type="checkbox" name="pizzaToppings[]"
  value="Sausage" /> Sausage</p>
```

Revised: What we get

The variables from our form:

- $_POST['custName']
- $_POST['pizzaSize']
- $_POST['pizzaToppings']
- $_POST['pizzaStatus']



Displaying Arrays

```
print_r($arrayName);
Array
(
     [0] => firstValue
     [1] => secondValue
     [2] => thirdValue
)
```

Displaying Arrays

```
var_dump($custInfo);

array(3) {
     ["firstName"]=> string(4) "Beth"
     ["lastName"]=> string(11) "Tucker Long"
     ["twitterHandle"]=> string(7) "e3betht"
}
```

while Loop

```
// count($arrayToCount);

$n = 0;
while($n < count($_POST['pizzaToppings']))  {
     echo
     "<li>{$_POST['pizzaToppings'][$n]}</li>";
     $n++;
}
```

do Loop

```
$n = 0;
do  {
     echo
     "<li>$_POST['pizzaToppings'][$n]</li>";
     $n++;
} while($n < count($_POST['pizzaToppings']));
```

for Loop

```
for($n = 0; $n <
count($_POST['pizzaToppings']); $n++;)  {
     echo
     "<li>$_POST['pizzaToppings'][$n]</li>";
}
```

foreach Loop

```
foreach($_POST['pizzaToppings'] as $topping)
{

     echo "<li>$topping</li>";

}
```

Associative foreach Loop

```
foreach($_POST['custInfo'] as $label =>
$value)  {

     echo "$label: $value<br />";

}
```

Displaying the Pizza Choices

```
echo "<p>This order is for
{$_POST['custName']}</p>
<p>Size: ".$_POST['pizzaSize']."</p>
<p>Toppings:</p><ul>";
foreach($_POST['pizzaToppings'] as $topping)
{
     echo "<li>$topping</li>";
}
echo "</ul>";
```

Reordering the Toppings

```
sort($_POST['pizzaToppings'];
echo "<p>Toppings:</p><ul>";
foreach($_POST['pizzaToppings'] as $topping)
{
     echo "<li>$topping</li>";
}
echo "</ul>";
```

Notes on Sorting

```
$pictures = array("img1", "img20", "img5",
"img10", "img3");

sort($pictures);
print_r($pictures);
```

```
Array (
        [0] => img1
        [1] => img10
        [2] => img20
        [3] => img3
        [4] => img5
)
```

Sorting Naturally

With PHP 5.4.0+:

```
$pictures = array("img1", "img20", "img5",
"img10", "img3");

sort($pictures, SORT_NATURAL);
print_r($pictures);
```

```
Array (
        [0] => img1
        [1] => img3
        [2] => img5
        [3] => img10
        [4] => img20
)
```

Sorting Naturally

With PHP 5.4.0+:

```
sort($place);
print_r($place);
```

```
Array (
        [0] => Greece
        [1] => Malaysia
        [2] => US
        [3] => Uganda
)
```

```
sort($place,
SORT_NATURAL);
print_r($place);
```

```
Array (
        [0] => Greece
        [1] => Malaysia
        [2] => Uganda
        [3] => US
)
```

Sorting with Keys

```
$winners = array("first" => "blue", "second"
=> "green", "third" => "purple");

sort($winners);
print_r($winners);
```

```
Array (
        [0] => blue
        [1] => green
        [2] => purple
)
```

Keeping Keys

```
$winners = array("first" => "blue", "second"
=> "green", "third" => "purple");

asort($winners);
print_r($winners);
```

```
Array (
        [third] => blue
        [first] => green
        [second] => purple
)
```

Sorting Keys

```
$winners = array("first" => "green",
"second" => "purple", "third" => "blue");

ksort($winners);
print_r($winners);
```

```
Array (
        [first] => green
        [second] => purple
        [third] => blue
)
```

Making Decisions

- Only display form when ordering pizza
- Afterwards, display only the receipt

Comparison Operators

==	Checks if the value of the two is the same
===	Checks if the value and data type of the two is the same
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Logical Operators

&& both operands are true (AND)

|| at least one operand is true (OR)

XOR exactly one operand is true

if

```
if ($_POST['pizzaStatus'] == "Place Order")
  {
  // Code to be executed
}
```

if-else

```
if ($_POST['pizzaStatus'] == "Place Order")
  {
  // Code to be executed
} else {
  // Code to be executed
}
```

if-elseif-else

```
if ($_POST['pizzaStatus'] == "Place Order")
  {
  // Code to be executed
} elseif ($_POST['pizzaStatus'] ==
  "Continue") {
  // Code to be executed
} else {
  // Code to be executed
}
```

Decision Code

```
<?php
if ($_POST['pizzaStatus'] == "Place Order") {
    echo "<p>This order is for {$_POST['custName']}</p>
    <p>Size: ".$_POST['pizzaSize']."</p>
    <p>Toppings:</p><ul>";
    foreach($_POST['pizzaToppings'] as $topping)  {
          echo "<li>$topping</li>";
    }
    echo "</ul>";
}
else {
?>
    <form action="./orderPizza.php" method="POST">
    <p>Name: <input type="text" name="custName" maxlength="200" /></p>
    <p>Choose a Size:<br />
    <input type="radio" name="pizzaSize" value="Small" /> Small<br />
    <input type="radio" name="pizzaSize" value="Medium" /> Medium<br />
    <input type="radio" name="pizzaSize" value="Large" /> Large</p>
    <p>Add Additional Toppings:<br />
    <input type="checkbox" name="pizzaToppings[]" value="Mushrooms" /> Mushrooms<br />
    …
    <input type="submit" name="pizzaStatus" value="Place Order" />
    </form>
<?php
}
?>
```

Loop Uh-oh

Warning: Invalid argument supplied for foreach() in **/your/dir/path/file.php** on line **6**

```
foreach($_POST['pizzaToppings'] as $topping)
{
     echo "<li>$topping</li>";
}
```

Corrected Loop

```
if(is_array($_POST['pizzaToppings']) {
     foreach($_POST['pizzaToppings'] as
          $topping)  {
          echo "<li>$topping</li>";
     }
}

if(count($_POST['pizzaToppings']) > 0) {
```

Corrected Decision Code

```
<?php
if ($_POST['pizzaStatus'] == "Place Order") {
   echo "<p>This order is for {$_POST['custName']}</p>
   <p>Size: ".$_POST['pizzaSize']."</p>
   <p>Toppings:</p><ul>";
   if(is_array($_POST['pizzaToppings']) {
         foreach($_POST['pizzaToppings'] as $topping)  {
                 echo "<li>$topping</li>";
         }
   }
   echo "</ul>";
}
else {
?>
   <form action="./orderPizza.php" method="POST">
   <p>Name: <input type="text" name="custName" maxlength="200" /></p>
   …
   <input type="submit" name="pizzaStatus" value="Place Order" />
   </form>
<?php
}
?>
```

Validation

```
if (strlen($_POST['custName']) < 1) {
  $errorMessages[] = "Please enter your Name.";
}

if(count($_POST['pizzaToppings'] < 1) {
  $errorMessages[] = "Please choose at least
  one topping.";
}
```

Validation

Besides incomplete submissions, we also always want to avoid malicious submissions.

```
$_POST['custName'] =
  htmlentities($_POST['custName']);



if(!ctype_alpha($_POST['pizzaSize'])) {
   $errorMessages[] = "Please choose a Size.";
}
```

Validation

```
if (is_array($_POST['pizzaToppings'])) {
  $checkToppings = implode("a",$_POST['pizzaToppings']);

  $checkToppings = str_replace(" ","a", $checkToppings);

  if(!ctype_alpha($checkToppings)) {
      $errorMessages[] = "Please choose some
  Toppings.";
  }
}

if(!ctype_alpha(str_replace(" ","a",
  implode("a",$_POST['pizzaToppings'])))) {
```

Basic Security

Validate input; Escape out.

Failure Happens

When a validation test fails, make it easy for your user to fix it (Check for malicious submissions, but always treat your users as though it were an accident).

```
if ($_POST['pizzaStatus'] == "Place Order")
   {
   //All our validation tests here
   if(is_array($errorMessages)) {
       echo "<ul>";
       foreach($errorMessages as $message)
   {
              echo "<li>$message</li>";
       }
       echo "</ul>";
       //Form Code Goes Here
   }
   else {
       //Confirmation Code Goes Here
   }
}
else {
   //Form Code Goes Here
}
```

Returning the Form with Data

```
<p>Name: <input type=\"text\" name=\"custName\"
  maxlength=\"200\" value=\"$custName\" /></p>
```

Returning the Form with Data

```
<p>Choose a Size:<br />
<input type=\"radio\" name=\"pizzaSize\"
  value=\"Small\" ";
if($pizzaSize == "Small") { echo "checked "; }
echo "/> Small<br />";
```

Returning the Form with Data

```
<p>Add Additional Toppings:<br />
<input type="\"checkbox\""
  name=\"pizzaToppings[]\" value=\"Mushrooms\"
  ";
if(in_array("Mushrooms",$pizzaToppings)) { echo
  "checked "; }
echo "/> Mushrooms<br />";
```

Aside

```
echo "<textarea>$data</textarea>";
```

```
echo "<select name="choice">
<option value=\"Yes\"";
if($_POST['choice'] == "Yes") {
  echo " selected";
}
echo "> Yes</option>";
```

Highlight Fields

```
if (strlen($_POST['custName']) < 1) {
  $errorFields[] = "custName";
  $errorMessages['custName'] = "Please enter your
  Name.";
}
if(in_array("custName", $errorFields) {
  echo "<p
  class=\"error\">{$errorMessages['custName']}<br
  />";
} else {
  echo "<p>";
}
echo "Name: <input type=\"text\" name=\"custName\"
  maxlength=\"200\" value=\"$custName\" /></p>";
```

Highlight Fields

```
if (strlen($_POST['custName']) < 1) {
  $errorMessages['custName'] = "Please enter your
  Name.";
}
if(array_key_exists("custName", $errorMessages) {
  echo "<p
  class=\"error\">{$errorMessages['custName']}<br
  />";
} else {
  echo "<p>";
}
echo "Name: <input type=\"text\" name=\"custName\"
  maxlength=\"200\" value=\"$custName\" /></p>";
```

Lots of Redundancy

```
if ($_POST['pizzaStatus'] == "Place Order") {
  //All our validation tests here
  if(is_array($errorMessages)) {
      echo "<ul>";
      foreach($errorMessages as $message) {
            echo "<li>$message</li>";
      }
      echo "</ul>";
      //Form Code Goes Here
  }
  else {
      //Confirmation Code Goes Here
  }
}
else {
  //Form Code Goes Here
}
```

Reducing Redundancy

```
function checkIfBad($fieldName) {
    if (strlen($_POST[$fieldName] > 0)) {
        if(strpos($_POST[$fieldName],"=") === false) {
                return true;
        }
        else {
                return false;
        }
    }
    else {
        return false;
    }
}
if (checkIfBad("custName")) {  $errorMessage[] = "custName message"; }
if (checkIfBad("pizzaSize")) {  $errorMessage[] = "pizzaSize message"; }
```

Ternary Operator

```
function checkIfBad($fieldName) {
    if (strlen($_POST[$fieldName] > 0)) {
        $result = (strpos($_POST[$fieldName],"=") ===  false) ? true : false;
    }
    else {
        $result = false;
    }
    return $result;
}
if (checkIfBad("custName")) {  $errorMessage[] = "custName message"; }
if (checkIfBad("pizzaSize")) {  $errorMessage[] = "pizzaSize message"; }
```

Reducing Redundancy

Create the function:

```
function displayForm($custName, $pizzaSize,
  $pizzaToppings) {
      //echo Form code Here
}
```

And then use it whenever you need it:

```
else {
  displayForm($_POST['custName'], $_POST['pizzaSize'],
  $_POST['pizzaToppings']);
}
```

Quick Note on Scope

Variables are passed in "by value" by default:

```
function changeNumber($myNumber) {
    $myNumber = 5;
}

$myNumber = 11;

changeNumber($myNumber);

echo $myNumber;
// 11
```

Passing by Reference

```
function changeNumber(&$myNumber) {
    $myNumber = 5;
}

$myNumber = 11;

changeNumber($myNumber);

echo $myNumber;
// 5
```

Required Parameters

```
function changeNumber(&$myNumber, $changeTo) {
        $myNumber = $changeTo;
}
$myNumber = 11;

changeNumber($myNumber);
```

```
Warning: Missing argument 2 for changeNumber(), called in
    /your/file/path/file.php on line 9 and defined in
    /your/file/path/file.php on line 3
```

Optional Parameters

```
function changeNumber(&$myNumber, $changeTo = 5) {
        $myNumber = $changeTo;
}
$myNumber = 11;

changeNumber($myNumber);
echo $myNumber;
// 5

changeNumber($myNumber, 7);
echo $myNumber;
// 7
```

A Few More Functions

- **strtoupper($string)**

  ```
  echo strtoupper("this is my phrase");
  // THIS IS MY PHRASE
  ```

- **strtolower($string)**

  ```
  echo strtolower("HELLO");
  // hello
  ```

- **substr($string, $start, $length)**

  ```
  echo substr("the quick fox", 4, 5);
  // quick
  echo substr("the quick fox", -3);
  // fox
  ```

A Few More Functions

- **trim($string)**

```
echo trim("       phrase");
// phrase
```

- **str_word_count($string, $format, $charlist)**

```
echo str_word_count("the quick fox",0);
// 3
var_dump(str_word_count("quick fox", 1));
// array(2) { [0]=> string(5) "quick" [1]=> string(3) "fox" }
echo str_word_count("ab lsab lsab 12 45",0);
// 3
echo str_word_count("ab lsab lsab 12 45",0,"45");
// 4
```

Printable receipt

- Nice format for printing
- No header/footer graphics

Accessing the Data

Sessions:

- Server-side
- Less picky on header timing

Cookies:

- Client-side
- Must occur before headers are sent

Both:

- Allow data to be stored by one script and accessed by another
- Accessible via superglobal array

Using Sessions

Place this at the very top of your page:

```
session_start();
```

This must occur before headers are sent. Things that will send the headers:

- the HTML declarations
- Whitespace
- echo'ing anything

Using Sessions

In our script, add this below the confirmation code:

```
$_SESSION['custName'] = $_POST['custName'];
$_SESSION['pizzaSize'] = $_POST['pizzaSize'];
$_SESSION['pizzaToppings'] = $_POST['pizzaToppings'];
```

Faster, but could cause security concerns:

```
$_SESSION['data'] = $_POST;
//$_SESSION['data']['pizzaToppings']
```

Using Sessions

Place this where you want the print link to display:

```
echo "<a href=\"printReceipt.php\">Printable
  Receipt</a>";
```

Script for Printing

```
<?php
   session_start();

   echo "<p>This order is for {$_SESSION['custName']}</p>
   <p>Size: ".$_SESSION['pizzaSize']."</p>
   <p>Toppings:</p><ul>";
   if(is_array($_SESSION['pizzaToppings'])) {
       foreach($_SESSION['pizzaToppings'] as $topping)  {
              echo "<li>$topping</li>";
       }
   }
   echo "</ul>";
?>
```

Got cookies?

Order form code needs to be reorganized so that the validation occurs before any HTML is outputted to the browser. Then add:

```
setcookie("custName", $_POST['custName']);
setcookie("pizzaSize", $_POST['pizzaSize']);
setcookie("pizzaToppings",
  serialize($_POST['pizzaToppings']);
```

Got cookies?

Your print script is updated to:

```
<?php
   echo "<p>This order is for {$_COOKIE['custName']}</p>
   <p>Size: ".$_COOKIE['pizzaSize']."</p>
   <p>Toppings:</p><ul>";
   $pizzaToppings =
   unserialize(stripslashes($_COOKIE['pizzaToppings']));
   if(is_array($pizzaToppings)) {
      foreach($pizzaToppings as $topping)  {
         echo "<li>$topping</li>";
      }
   }
   echo "</ul>";
?>
```

OOP

```
class pizza{
    public $custName, $pizzaSize, $pizzaToppings;

    public function __construct($name, $size, $toppings) {
        $this-> custName = $name;
        $this-> pizzaSize = $size;
        $this-> pizzaToppings = $toppings;
    }
    public function checkout() {
        echo “Thank you, {$this->custName}. You are purchasing a {$this->pizzaSize} with the
        following toppings:<ul>";
        if(is_array($this->pizzaToppings) {
            foreach($this->pizzaToppings as $topping) {
                echo "<li>$toppings<li>";
            }
        }
    }
}
```

OOP

```
$myPizza = new pizza("Beth", "Medium", $toppings);

$myPizza->checkout();

// Thank you, Beth. You are purchasing a medium with the
following toppings:<ul><li>Extra Cheese</li><li>Black
Olives</li></ul>";


echo $myPizza->custName;

// Beth
```

Child Classes

```
class thinCrust extends pizza{
        public $sides;

        public function __construct($chosenSides, $name, $size, $toppings) {
                $this->sides = $chosenSides;
                parent::__construct($name, $size, $toppings);
        }
        public function checkout() {
                echo “Thank you, {$this->custName}. You are purchasing a thin crust {$this->pizzaSize} with the following {$this->sides} and toppings:<ul>";
                if(is_array($this->pizzaToppings) {
                        foreach($this->pizzaToppings as $topping) {
                                echo "<li>$toppings<li>";
                        }
                }
        }
}
```

Child Classes

```
$myPizza = new thinCrust("salad","Beth", "Medium", $toppings);
$myPizza->checkout();
// Thank you, Beth. You are purchasing a thin crust medium with
the following salad and toppings:<ul><li>Extra
Cheese</li><li>Black Olives</li></ul>";

echo $myPizza->custName;
// Beth
```

More Permanent Storage

- Storing in a database, MySQL
  - id
  - name
  - pizzaSize
  - pizzaToppings
  - orderDate

mysqli

- Need MySQL 4.1+
- If you are using PHP 5.2.9+, you can use the OOP format.

Connecting

```
$myConnection= new mysqli('hostname', 'username', 'password',
        'databaseName');
if($myConnection->connect_error) {
        die('Connection Error: ' . $myConnection->connect_errno .
        ": " . $myConnection->connect_error);
}
```

```
$myConnection= mysqli_connect('hostname', 'username',
        'password', 'databaseName');
if(mysqli_connect_error()) {
        die('Connection Error: ' . mysqli_connect_errno() . ": " .
        mysqli_connect_error());
}
```

Querying

```
$resultSet= $myConnection->query('select * from books');
$resultSet= mysqli_query($myConnection, 'select * from books');

echo $resultSet->num_rows;
echo mysqli_num_rows($resultSet);

while($row = $resultSet->fetch_assoc()) {
        echo "{$row['title']} was written by {$row['author']}<br/>";
}
while($row = mysqli_fetch_assoc($resultSet)) {
        echo "{$row['title']} was written by {$row['author']}<br/>";
}
```

Inserting Data

```
$myConnection->query("insert into books (title,
        author) values ('New Book', 'New Author')");
mysqli_query($myConnection, "insert into books (title,
        author) values ('New Book', 'New Author')");


echo $myConnection->affected_rows;
echo mysqli_affected_rows($myConnection);
```

Basic Security

```
$_POST['username'] = "attacker";
$_POST['password'] = "x' or 'a' = 'a";

$username = $myConnection->real_escape_string($_POST['username'] );
$password = $myConnection->real_escape_string($_POST['password'] );

$username = mysqli_real_escape_string($myConnection,
            $_POST['username'] );
$password = mysqli_real_escape_string($myConnection,
            $_POST['password'] );

$query = "select * from users where username = '$username' and
            password = '$password'";

// select * from users where username = 'attacker' and password = 'x\' or
            \'a\' = \'a'
```

Prepared Statements

```
$preparedQuery= $myConnection->prepare("insert into books (title, author,
price)    values (?, ?,?)");
$preparedQuery->bind_param("Comic Books and You", "StanleeJ. "Siegel",
          19.99);
$preparedQuery->execute();
$resultSet= $preparedQuery->get_result();

$preparedQuery= mysqli_stmt_init($myConnection);
mysqli_stmt_prepare($preparedQuery, "insert into books (title, author,
price)    values (?, ?, ?)");
mysqli_stmt_bind_param($preparedQuery, "ssd", "Comic Books and You",
          "StanleeJ. "Siegel", 19.99);
mysqli_stmt_execute($preparedQuery);
$resultSet= mysqli_stmt_get_result($preparedQuery);
```

PDO

- PHP Data Objects
- Data-access abstraction layer, not database abstraction
- Requires PHP 5.1+
- Requires PDO driver for your specific database: http://php.net/manual/en/pdo.drivers.php

Connecting

```
$databaseInfo='mysql:dbname=testdb;host=127.0.0.1';
$username='dbuser';
$password='dbpass';

try {
            $dbConnection=newPDO($ databaseInfo,$username,$password);
} catch(PDOException$error) {
            echo'Connectionfailed:'.$error->getMessage();
}
```

Querying

```
$query='SELECT name, price FROM products';

foreach($dbConnection->query($query)as$row) {
        echo "One {$row['name']} costs \${$row['price']}";
}
```

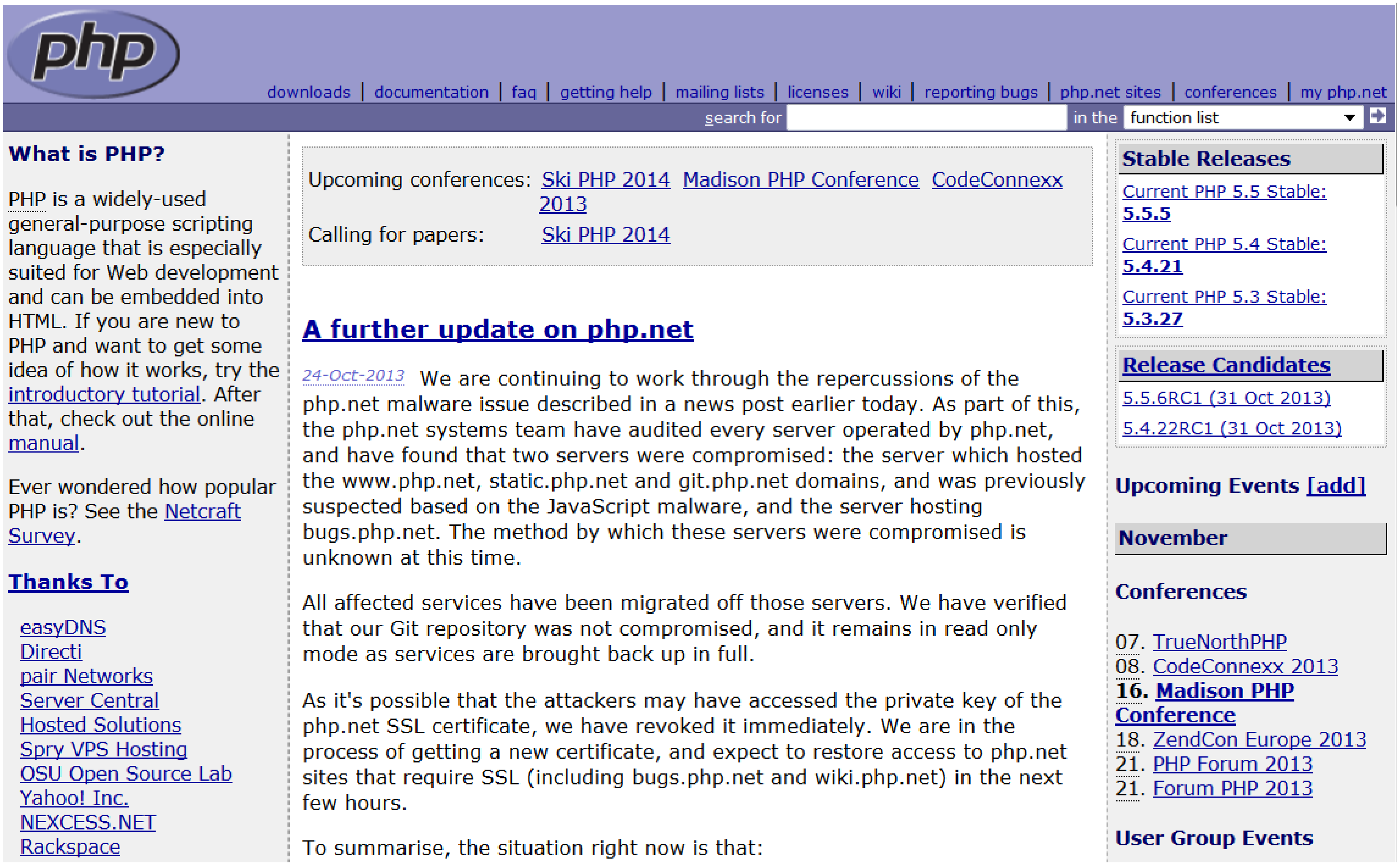
Prepared Statements

```
$query = 'SELECT name, price FROM products
            WHERE name LIKE :name AND price <= :price';

$runQuery = $dbConnection->prepare($query);

$runQuery>execute(array(':name' => '%fishing%',':price' => 20));
$fishingBooks = $runQuery->fetchAll();

$runQuery->execute(array(':name' => '%cookie%',':price' => 10));
$cookieBooks = $runQuery->fetchAll();
```

More Prepared Statements

```
$query = 'SELECT name, price FROM products
            WHERE name LIKE ? AND price <= ?';
$runQuery=$dbConnection->prepare($query);
$runQuery->execute(array('%fishing%', 20));

$query='SELECT name, price FROM products
            WHERE name LIKE ? AND price <= ?';
$runQuery=$dbConnection->prepare($query);
$runQuery->bindParam(1, '%fishing%', PDO::PARAM_STR, 9);
$runQuery->bindParam(2, 20, PDO::PARAM_INT);

$query='SELECT name, price FROM products
            WHERE name LIKE :name AND price <= :price';
$runQuery->bindParam(':name', '%fishing%', PDO::PARAM_STR, 9);
$runQuery->bindParam(':price', 20, PDO::PARAM_INT);
```

php.net


php
downloads | documentation | faq | getting help | mailing lists | licenses | wiki | reporting bugs | php.net sites | conferences | my php.net
search for
in the function list

What is PHP?

PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. If you are new to PHP and want to get some idea of how it works, try the introductory tutorial. After that, check out the online manual.

Ever wondered how popular PHP is? See the Netcraft Survey.

Thanks To

easyDNS
Directi
pair Networks
Server Central
Hosted Solutions
Spry VPS Hosting
OSU Open Source Lab
Yahoo! Inc.
NEXCESS.NET
Rackspace

Upcoming conferences: Ski PHP 2014 Madison PHP Conference CodeConnexx 2013
Calling for papers: Ski PHP 2014

A further update on php.net

24-Oct-2013 We are continuing to work through the repercussions of the php.net malware issue described in a news post earlier today. As part of this, the php.net systems team have audited every server operated by php.net, and have found that two servers were compromised: the server which hosted the www.php.net, static.php.net and git.php.net domains, and was previously suspected based on the JavaScript malware, and the server hosting bugs.php.net. The method by which these servers were compromised is unknown at this time.

All affected services have been migrated off those servers. We have verified that our Git repository was not compromised, and it remains in read only mode as services are brought back up in full.

As it's possible that the attackers may have accessed the private key of the php.net SSL certificate, we have revoked it immediately. We are in the process of getting a new certificate, and expect to restore access to php.net sites that require SSL (including bugs.php.net and wiki.php.net) in the next few hours.

To summarise, the situation right now is that:

Stable Releases

Current PHP 5.5 Stable: **5.5.5**

Current PHP 5.4 Stable: **5.4.21**

Current PHP 5.3 Stable: **5.3.27**

Release Candidates

5.5.6RC1 (31 Oct 2013)

5.4.22RC1 (31 Oct 2013)

Upcoming Events [add]

November

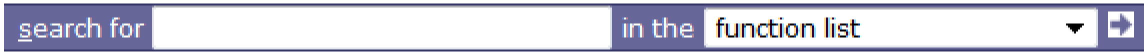
Conferences

07. TrueNorthPHP
08. CodeConnexx 2013
16. Madison PHP Conference
18. ZendCon Europe 2013
21. PHP Forum 2013
21. Forum PHP 2013

User Group Events

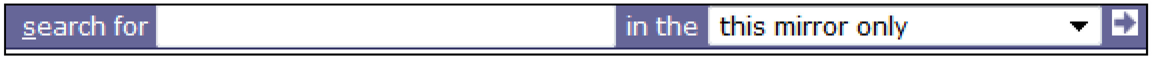
Searching php.net

- http://www.php.net/strlen
- Search the function list:



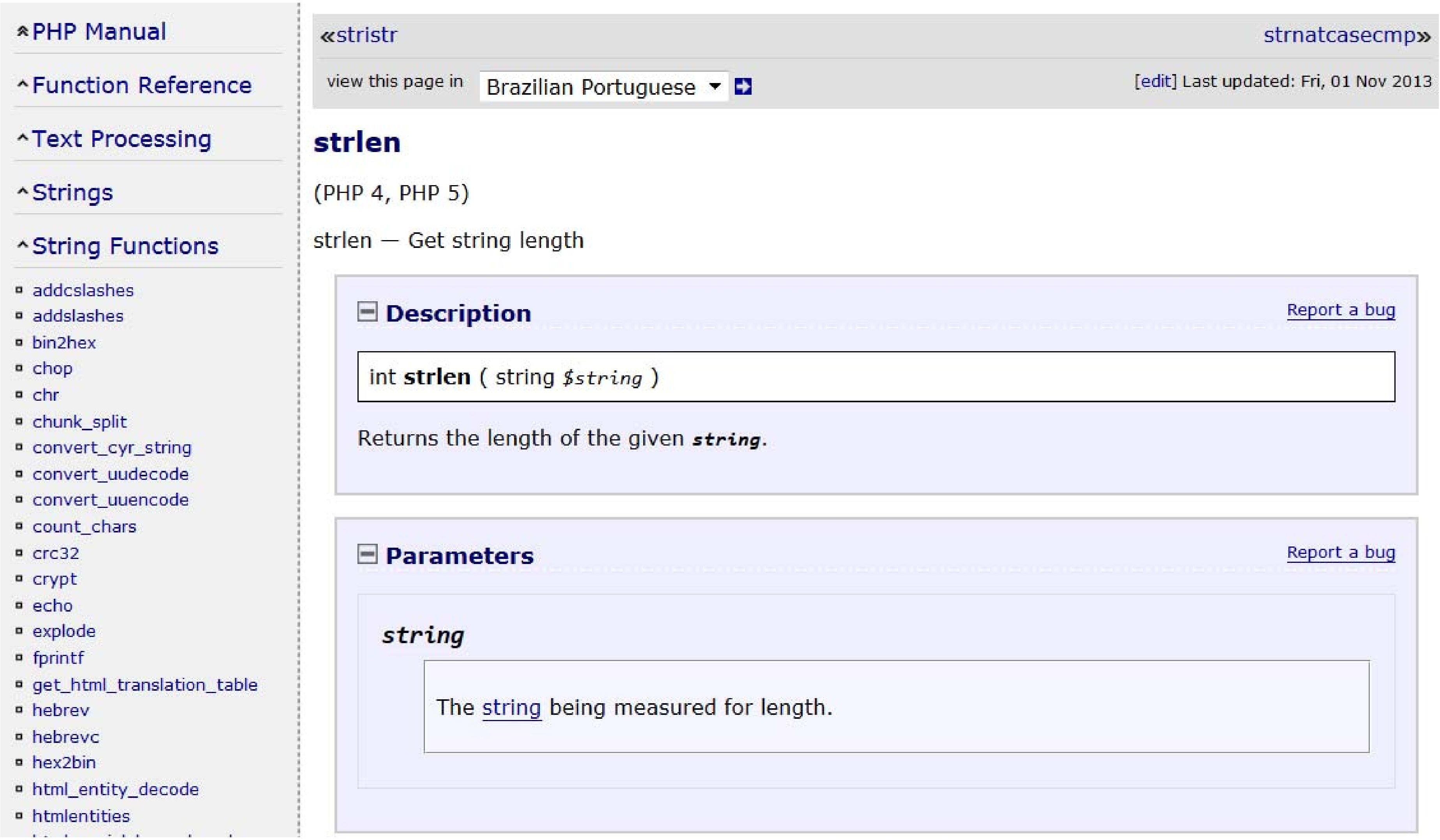
search for in the function list

- Search the website content:



search for in the this mirror only

Function Pages



- ^PHP Manual
- ^Function Reference
- ^Text Processing
- ^Strings
- ^String Functions

- addcslashes
- addslashes
- bin2hex
- chop
- chr
- chunk_split
- convert_cyr_string
- convert_uudecode
- convert_uuencode
- count_chars
- crc32
- crypt
- echo
- explode
- fprintf
- get_html_translation_table
- hebrev
- hebrevc
- hex2bin
- html_entity_decode
- htmlentities

«stristr strnatcasecmp»

view this page in Brazilian Portuguese [edit] Last updated: Fri, 01 Nov 2013

strlen

(PHP 4, PHP 5)

strlen — Get string length

Description

Report a bug

```
int strlen ( string $string )
```

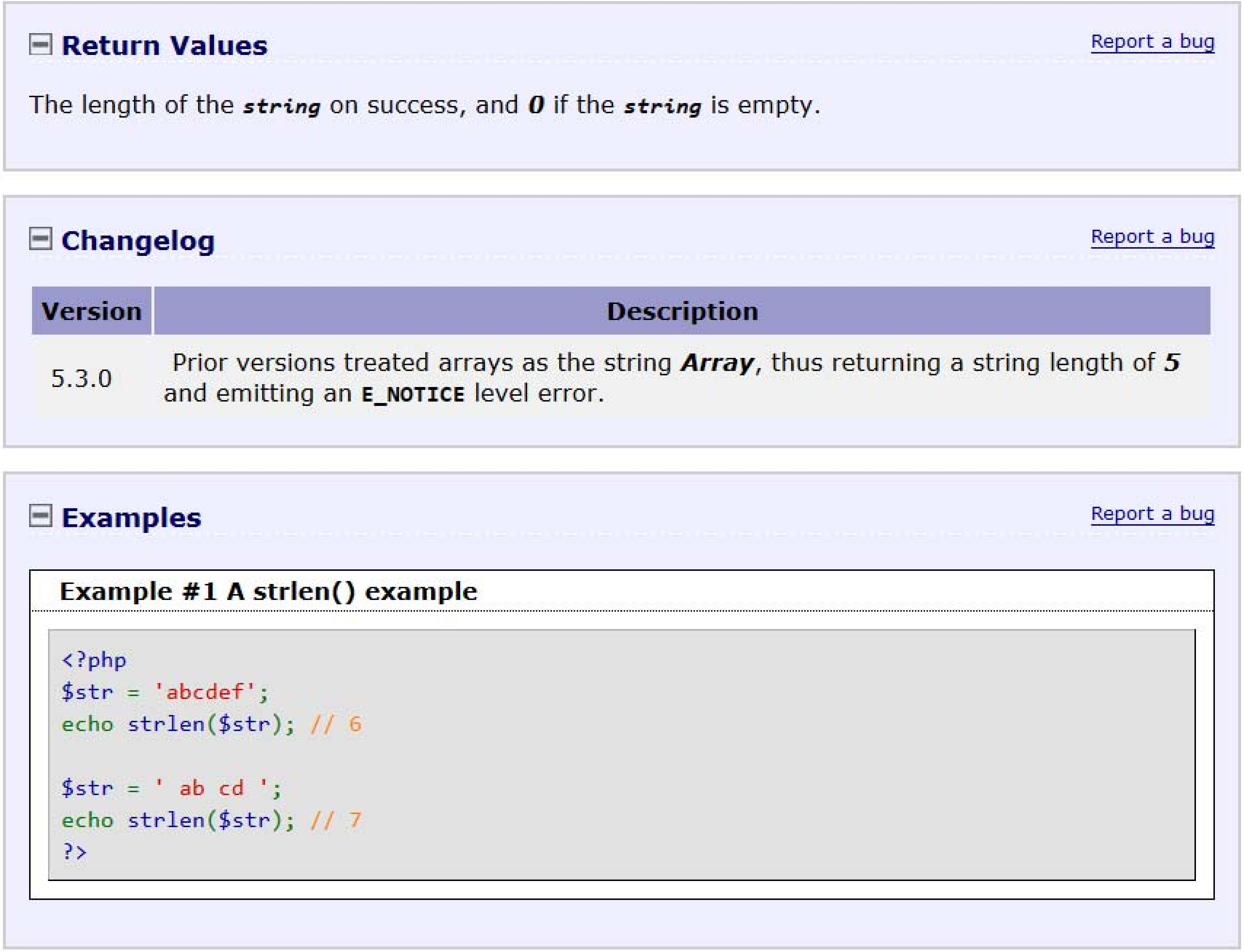
Returns the length of the given ***string***.

Parameters

Report a bug

string

The string being measured for length.



Return Values

The length of the ***string*** on success, and ***0*** if the ***string*** is empty.

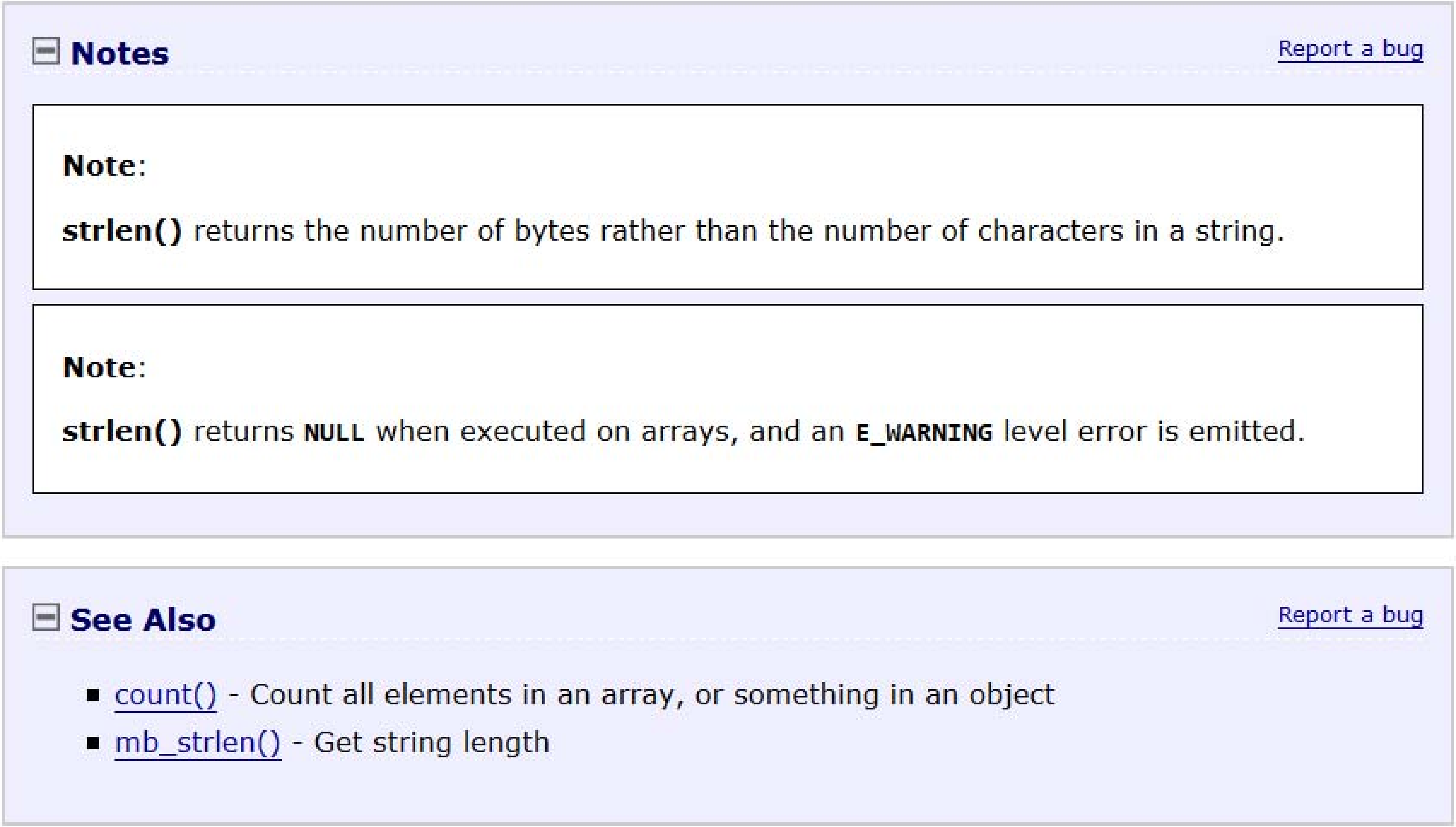
Changelog

Version	Description
5.3.0	Prior versions treated arrays as the string ***Array***, thus returning a string length of ***5*** and emitting an **E_NOTICE** level error.

Examples

Example #1 A strlen() example

```php
<?php
$str = 'abcdef';
echo strlen($str); // 6

$str = ' ab cd ';
echo strlen($str); // 7
?>
```



Notes

Note:

strlen() returns the number of bytes rather than the number of characters in a string.

Note:

strlen() returns **`NULL`** when executed on arrays, and an **`E_WARNING`** level error is emitted.

See Also

- count() - Count all elements in an array, or something in an object
- mb_strlen() - Get string length

Followed by User Comments

Common Problems

Parse error: syntax error, unexpected '{' in /your/path/file.php on line 7

```
if(empty($myVar) {
        echo "This is empty!";
}
```

Common Problems

Parse error: syntax error, unexpected T_STRING, expecting ',' or ';' in /your/path/file.php on line 18

```
echo "Hello;
echo "Hi Again";
if ($this === $that) {
        echo "Hi a third time";
}
```

Common Problems

```
$myVar = 5;

if ($myVar = "10") {
	echo "They match!";
else {
	echo "Try Again.";
}

//Always outputs "They match!"
```

Yoda Syntax

```
$myVar = 5;

if ("10" = $myVar) {
        echo "They match!";
else {
        echo "Try Again.";
}
```

```
Parse error: syntax error, unexpected '=' in
/your/path/file.php on line 5
```

Code Samples

```
$phrase = "PHP is awesome!";
$makeUpper = true;

for($n = 0; $n < strlen($phrase); $n++) {
  if(ctype_alpha($phrase[$n])) {
    if($makeUpper) {
      $phrase[$n] = strtoupper($phrase[$n]);
    } else {
      $phrase[$n] = strtolower($phrase[$n]);
    }
    $makeUpper = !$makeUpper;
  }
}

echo "$phrase";  // PhP iS aWeSoMe!
```

Find Me

- Twitter: e3betht
- Madison PHP
  http://www.madisonphp.com
- Slides Available:
  http://www.TreelineDesign.com/slides

Want more? Take a PHP course! Visit:
www.phparch.com
and click on "TRAINING" for registration info.

php[architect]

FREE ISSUES!

Ask me about writing articles for the magazine!

http://www.phparch.com

April 2013
VOLUME 12 - ISSUE 4
phparch.com
Security
Securing File Uploads
Best Practices For Password Hashing
When a Password Isn't Enough: Two-factor Authentication and You
Unit Testing Essentials: Mock Objects
INSIDE:

May 2013
phparch.com
Getting Started Writing Your Own Micro-framework
Ron Dobley
Cleared for Takeoff: Getting Off the Ground with Flight
Andrew Podner
Columns
Editorial: Who Needs Frameworks?
Education Station: Simple HTTP Clients with Guzzle
finally(): Open Acceptance in Open Source
micro-frameworks
Also
Learning How to Learn
Avoiding System Dependency by Decoupling Your Dev Environment

www.phparch.com
June 2013
php[architect]
PHP Inside
The LAMP Behind a #1 Mobile App
An Introduction to LUCID Development
Continuous Testing in PHP
ALSO INSIDE
MySQL Forks: Which One is Right for Me?
Editorial: A Group Effort
Education Station: Simple HTTP Clients with Guzzle (Part 2)
finally(): The Changing PHP Community
php

Feedback or Questions

Joind.in:

https://joind.in/9962

E-mail:

Beth@Musketeers.me