

Strings And Patterns

*Orlando PHP Meetup
Zend Certification Training
April 2009*



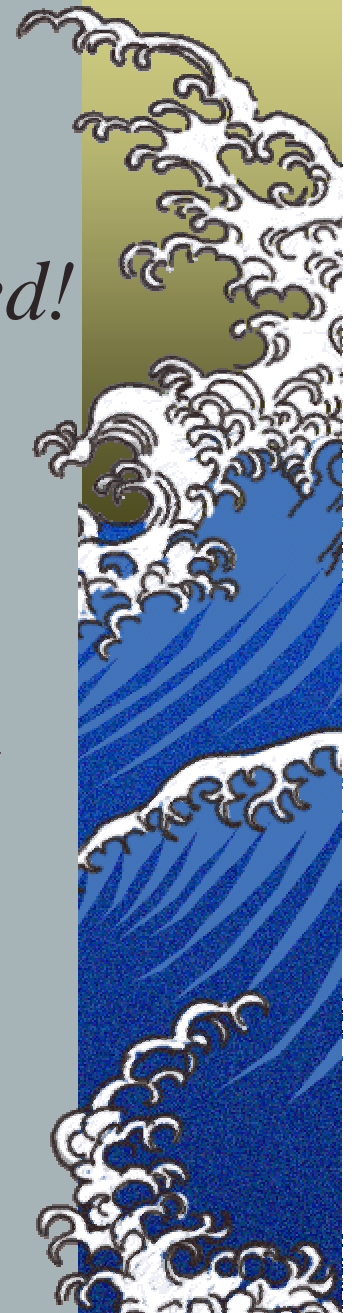
3 Ways to Make Strings

- ▶ *Single Quote – Only ‘ needs to be escaped. No variable interpolation.*
- ▶ *Double Quote – Variable interpolation and special escape tags.*
 - ▶ *\$var = “This \$string needs {\$string}s. \n”;*
- ▶ *Heredoc (<<<) for multi-line strings*
 - ▶ *\$string = <<<EOSTRING*
this is a \$interpreted “string”
\ton multiple \$lines[10]
EOSTRING



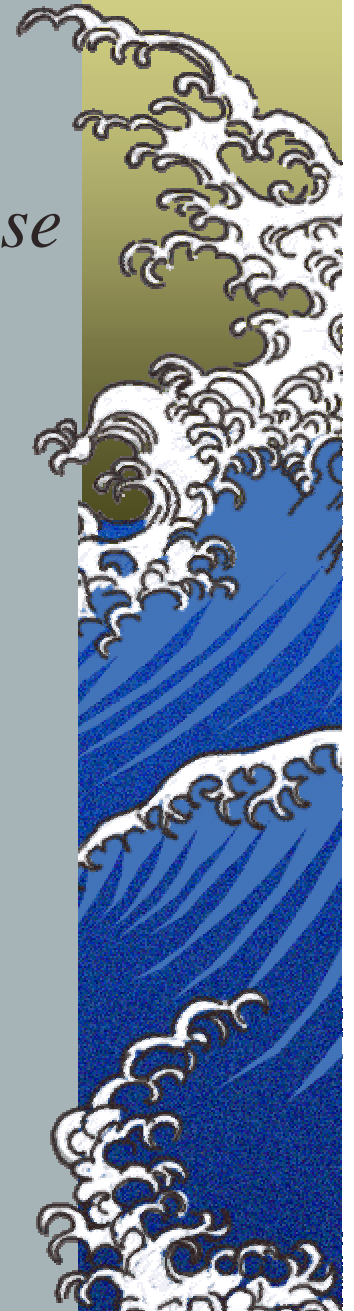
String Functions

- ★ *Strings as arrays: Don't forget: zero-based!*
 - ★ *`$var = "abcdef";`*
`$a = $var[0];`
- ★ *`strlen($string)` returns length in bytes*
- ★ *`strtr($string, $find, $replace)` - Transform*



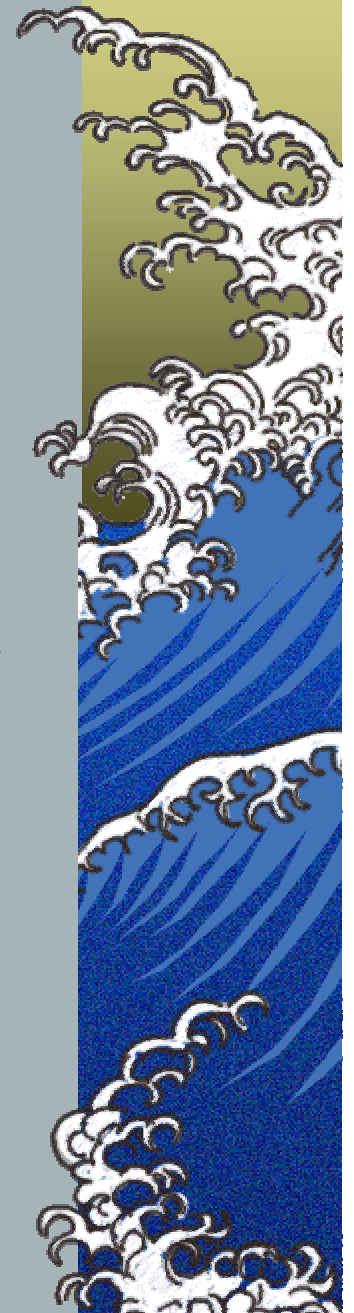
Searching

- ▶ *Most string functions have case sensitive and case insensitive versions*
- ▶ *Most parameters can also be arrays of strings.*
- ▶ *strcmp(\$st1, \$st2) returns 0 if equal*
- ▶ *strpos(\$haystack, \$needle) returns integer or FALSE if not found. Be sure to compare ===*
- ▶ *strstr(\$haystack, \$needle) returns rest of string, starting at needle, or FALSE*



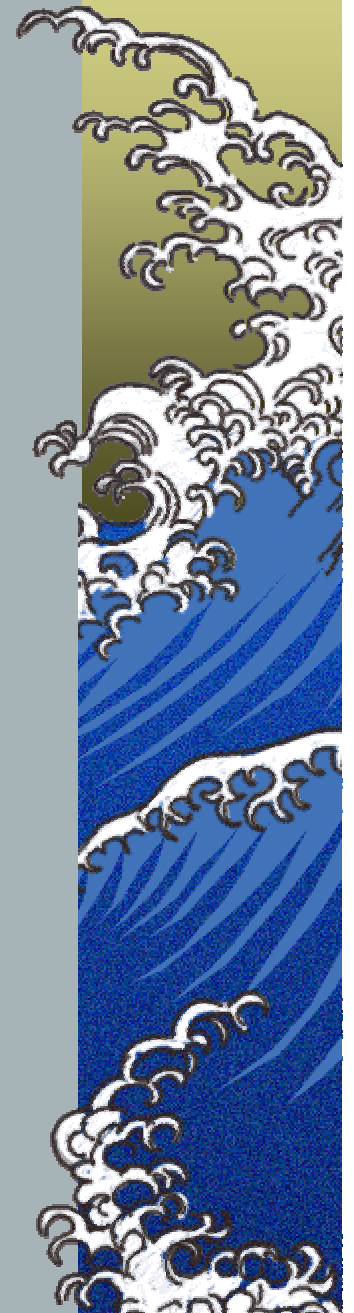
Replacing

- ▶ *str_replace(\$find, \$replace, \$haystack)*
 - ▶ *Replaces all instances of \$find with \$replace*
- ▶ *substr(\$string, \$start, \$length)*
 - ▶ *Returns a slice of the string*
 - ▶ *\$start may be negative, for starting from end.*
- ▶ *substr_replace(\$haystack, \$replacement, \$start, \$length)*
 - ▶ *Only replaces a slice of the haystack*



Formatting

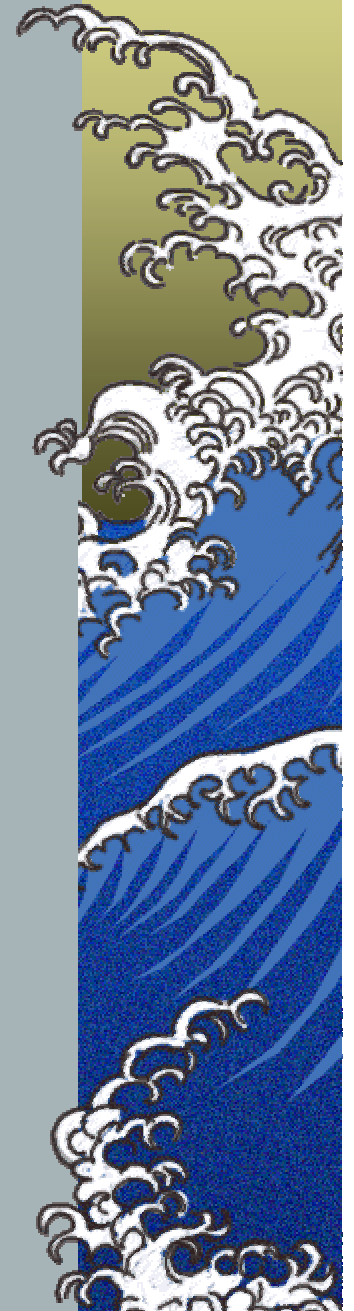
- ▶ *number_format(\$num, \$decimals, \$decimal_point, \$thousands_sep)*
- ▶ *Generic Formatting*
 - ▶ *Format String defines type (string, decimal), length, sign, special handling.*
 - ▶ *%d (decimal), %f (float), %s (string)*
 - ▶ *Modifiers: %4.3f shows 1234.56*
 - ▶ *sprintf(‘%4.3f is %d long’, 1234.563, 7) = “1234.56 is 7 long”*



Perl Regular Expressions

- ▶ *A pattern matching language that defines what to find when you don't know exactly what you are looking for.*
- ▶ *Match patterns*

.	<i>Any character except newline</i>
^	<i>Beginning of line</i>
\$	<i>End of line</i>
[]	<i>Character class. Ex: [a-zA-z0-9\s]</i>
\s	<i>Whitespace (space, tab) \S is non-whitespace</i>
\d	<i>Digit. Equal to [0-9]</i>
\w	<i>Word (a-z or underline)</i>



Regular Expressions 2

Quantifiers

*	<i>Zero or more times</i>
+	<i>One or more times</i>
?	<i>Zero or one time</i>
{min,max}	<i>Match the pattern between minimum and maximum times.</i>

Patterns are wrapped in delimiters. Usually // but can be any character but backslash

Sub expressions

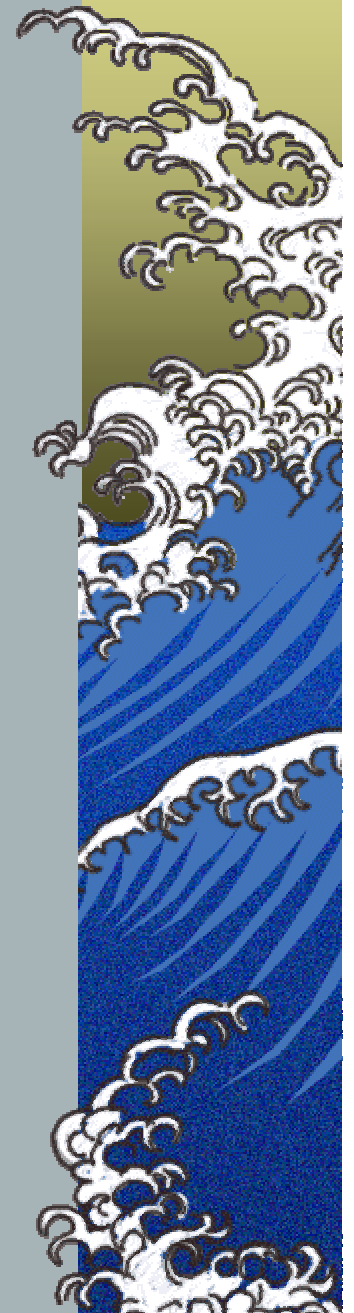
Parentheses around sub-matches

/abc(def[0-9])+ghi/ will match abcdef1ghi or abcdef1def2ghi



Regular Expression Functions

- ▶ *preg_match(\$pattern, \$haystack, \$matches)*
 - ▶ *Fills \$matches array with strings from \$pattern*
 - ▶ *Returns FALSE on no match*
- ▶ *preg_replace(\$pattern, \$replace, \$haystack)*
 - ▶ *For each pattern, replace match(es) with \$replace*
 - ▶ *If \$replace is an array it will use each match to use one element from the array*



Summary

- ▶ *Strings are the most flexible of core data types. Can hold text or binary data.*
 - ▶ *Read the documentation to find string functions that do what you need.*
 - ▶ *Use “simple” string functions for search and replace when you can.*
 - ▶ *Be careful of PHP5 only functions, like the case insensitive versions.*
 - ▶ *Regular expressions are powerful and there are often more than one way to match a pattern.*

