Array Functions

- array_diff (arr1, arr2 ...)
- array_filter (arr, function)
- array_flip (arr)
- array_intersect (arr1, arr2 ...)
- array_merge (arr1, arr2 ...)
- array_pop (arr)
- array_push (arr, var1, var2 ...)
- array_reverse (arr)
- array_search (needle, arr)
- array_walk (arr, function)
- count (count)
- in_array (needle, haystack)

String Functions

- crypt (str, salt)
- explode (sep, str)
- implode (glue, arr)
- nl2br (str)
- sprintf (frmt, args)
- strip_tags (str, allowed_tags)
- str_replace (search, replace, str)
- strpos (str, needle)
- strrev (str)
- strstr (str, needle)
- strtolower (str)
- strtoupper (str)
- substr (string, start, len)

Filesystem Functions

- clearstatcache ()
- copy (source, dest)
- fclose (handle)
- fgets (handle, len)
- file (file)
- filemtime (file)
- filesize (file)
- file_exists (file)
- fopen (file, mode)
- fread (handle, len)
- fwrite (handle, str)
- readfile (file)

fopen() Modes

r	Read
r+	Read and write, prepend
w	Write, truncate
w+	Read and write, truncate
a	Write, append
a+	Read and write, append

Regular Expression Functions

- ereg (pattern, str)
- split (pattern, str)
- ereg_replace (pattern, replace, str)
- preg_grep (pattern, arr)
- preg_match (pattern, str)
- preg_match_all (pattern, str, arr)
- preg_replace (pattern, replace, str)
- preg_split (pattern, str)

Regular Expressions Syntax

^	Start of string
$	End of string
.	Any single character
(a|b)	a or b
(...)	Group section
[abc]	Item in range (a, b or c)
[^abc]	Not in range (not a, b or c)
\s	White space
a?	Zero or one of a
a*	Zero or more of a
a*?	Zero or more of a, ungreedy
a+	One or more of a
a+?	One or more of a, ungreedy
a{3}	Exactly 3 of a
a{3,}	3 or more of a
a{,6}	Up to 6 of a
a{3,6}	3 to 6 of a
a{3,6}?	3 to 6 of a, ungreedy
\	Escape character
[:punct:]	Any punctuation symbol
[:space:]	Any space character
[:blank:]	Space or tab

PCRE Modifiers

i	Case-insensitive
s	Period matches newline
m	^ and $ match lines
U	Ungreedy matching
e	Evaluate replacement
x	Pattern over several lines

Date and Time Functions

- checkdate (month, day, year)
- date (format, timestamp)
- getdate (timestamp)
- mktime (hr, min, sec, month, day, yr)
- strftime (formatstring, timestamp)
- strtotime (str)
- time ()

Available free from **AddedBytes.com**

Date Formatting

Y	4 digit year (2008)
y	2 digit year (08)
F	Long month (January)
M	Short month (Jan)
m	Month [4] (01 to 12)
n	Month (1 to 12)
D	Short day name (Mon)
l	Long day name (Monday) (lowercase L)
d	Day [4] (01 to 31)
j	Day (1 to 31)
h	12 Hour [4] (01 to 12)
g	12 Hour (1 to 12)
H	24 Hour [4] (00 to 23)
G	24 Hour (0 to 23)
i	Minutes [4] (00 to 59)
s	Seconds [4] (00 to 59)
w	Day of week [1] (0 to 6)
z	Day of year (0 to 365)
W	Week of year [2] (1 to 53)
t	Days in month (28 to 31)
a	am or pm
A	AM or PM
B	Swatch Internet Time (000 to 999)
S	Ordinal Suffix (st, nd, rd, th)
T	Timezone of machine (GMT)
Z	Timezone offset (seconds)
O	Difference to GMT (hours) (e.g., +0200)
I	Daylight saving (1 or 0)
L	Leap year (1 or 0)
U	Seconds since Epoch [3]
c	ISO 8601 (PHP 5) 2008-07-31T18:30:13+01:00
r	RFC 2822 Thu, 31 Jul 2008 18:30:13 +0100

1. 0 is Sunday, 6 is Saturday.

2. Week that overlaps two years belongs to year that contains most days of that week. Hence week number for 1st January of a given year can be 53 if week belongs to previous year. date("W", mktime(0, 0, 0, 12, 8, $year)) always gives correct number of weeks in $year.

3. The Epoch is the 1st January 1970.

4. With leading zeroes

regularexpressions

Anchors

^	Start of line +
\A	Start of string +
$	End of line +
\Z	End of string +
\b	Word boundary +
\B	Not word boundary +
\<	Start of word
\>	End of word

Character Classes

\c	Control character
\s	White space
\S	Not white space
\d	Digit
\D	Not digit
\w	Word
\W	Not word
\xhh	Hexadecimal character hh
\Oxxx	Octal character xxx

POSIX Character Classes

[:upper:]	Upper case letters
[:lower:]	Lower case letters
[:alpha:]	All letters
[:alnum:]	Digits and letters
[:digit:]	Digits
[:xdigit:]	Hexadecimal digits
[:punct:]	Punctuation
[:blank:]	Space and tab
[:space:]	Blank characters
[:cntrl:]	Control characters
[:graph:]	Printed characters
[:print:]	Printed characters and spaces
[:word:]	Digits, letters and underscore

Assertions

?=	Lookahead assertion +
?!	Negative lookahead +
?<=	Lookbehind assertion +
?!= or ?<!	Negative lookbehind +
?>	Once-only Subexpression
?()	Condition [if then]
?()|	Condition [if then else]
?#	Comment

Note Items marked + should work in most regular expression implementations.

Sample Patterns

([A-Za-z0-9-]+)	Letters, numbers and hyphens
(\d{1,2}\/\d{1,2}\/\d{4})	Date (e.g. 21/3/2006)
([^\s]+(?=\.(jpg|gif|png))\.\2)	jpg, gif or png image
(^[1-9]{1}$\|^[1-4]{1}[0-9]{1}$|^50$)	Any number from 1 to 50 inclusive
(#?([A-Fa-f0-9]){3}(([A-Fa-f0-9]){3})?)	Valid hexadecimal colour code
((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,15})	8 to 15 character string with at least one upper case letter, one lower case letter, and one digit (useful for passwords).
(\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,6})	Email addresses
(\<(/?[^\>]+)\>)	HTML Tags

Note *These patterns are intended for reference purposes and have not been extensively tested. Please use with caution and test thoroughly before use.*

Quantifiers

*	0 or more +
*?	0 or more, ungreedy +
+	1 or more +
+?	1 or more, ungreedy +
?	0 or 1 +
??	0 or 1, ungreedy +
{3}	Exactly 3 +
{3,}	3 or more +
{3,5}	3, 4 or 5 +
{3,5}?	3, 4 or 5, ungreedy +

Special Characters

\	Escape Character +
\n	New line +
\r	Carriage return +
\t	Tab +
\v	Vertical tab +
\f	Form feed +
\a	Alarm
[\b]	Backspace
\e	Escape
\N{name}	Named Character

String Replacement (Backreferences)

$n	nth non-passive group
$2	"xyz" in /^(abc(xyz))$/
$1	"xyz" in /^(?:abc)(xyz)$/
$`	Before matched string
$'	After matched string
$+	Last matched string
$&	Entire matched string
$_	Entire input string
$$	Literal "$"

Ranges

.	Any character except new line (\n) +
(a|b)	a or b +
(...)	Group +
(?:...)	Passive Group +
[abc]	Range (a or b or c) +
[^abc]	Not a or b or c +
[a-q]	Letter between a and q +
[A-Q]	Upper case letter + between A and Q +
[0-7]	Digit between 0 and 7 +
n	nth group/subpattern +

Note *Ranges are inclusive.*

Pattern Modifiers

g	Global match
i	Case-insensitive
m	Multiple lines
s	Treat string as single line
x	Allow comments and white space in pattern
e	Evaluate replacement
U	Ungreedy pattern

Metacharacters (must be escaped)

^	[	.
$	{	*
(	\	+
)	|	?
<	>	

Available free from AddedBytes.com