PHP 4 Reference Card

by Steven R. Gould

Escaping HTML

preferred format	<?php ...?>
verbose	<script language="php">...</script>

Less portable, may be disabled in php.ini:

short-form	<?...?>
short-form expression	<?=*expr*?>
ASP-style	<%...%>
ASP-style expression	<%=*expr*%>

Basic syntax

statement terminator	;
C-style comments	/*...*/
C++-style comments	//...
shell-script-style comments	# ...
block delimiters	{ }
octal integers (prefix zero)	0
hexadecimal integers (prefix zero-ex)	0x or 0X
newline, cr, tab, backspace	\n, \r, \t, \b
special characters	\\, \?, \', \", \$

Data Types

boolean	TRUE/FALSE
integer	-101, 23, 69
floating point	3.141592
character string (parsed)	"..."
character string (unparsed)	'...'
class	class *name* {...}
resource (refer to PHP manual for details)	
array	array([*index*=>]*value*,...)

where *index* can be non-negative int or string

Predefined PHP variables

Many variables are defined that are specific to the web server and OS. Run **phpinfo()** for a complete list of these.

$argv	array of arguments passed to script
$argc	number of arguments in **$argv**
$PHP_SELF	filename of currently executing script

The following are only available if track_vars=On in php.ini

$HTTP_COOKIE_VARS	array of variables passed via cookies
$HTTP_GET_VARS	array of variables passed via GET
$HTTP_POST_VARS	array of variables passed via POST
$HTTP_POST_FILES	array of files uploaded via POST
$HTTP_ENV_VARS	array of variables from parent environment
$HTTP_SERVER_VARS	array of variables from HTTP server

Control Structures

include *filename*	include named file (only if executed)
include_once *filename*	include named file (at most once)
require *filename*	include named file, like C/C++ #include
require_once *filename*	include named file, if not already included

Flow of Control

exit from switch, while, do, for	break
next iteration of while, do, for	continue
go to (avoid if possible!)	goto *label*
label	*label*:
return value from function	return *expr*
terminate execution	exit(*arg*)

Flow Constructions (if/while/for/do/switch)

```
if (expr)  statement
else if (expr)  statement
else  statement
```

```
for (expr1; expr2; expr3)
  statement
```

```
while (expr)
  statement
```

```
do   statement
while(expr);
```

```
switch (expr) {
   case const1: statement1 break;
   case const2: statement2 break;
   default: statement
}
```

Functions

```
function name([arg,...,arg[=default]]) {
  statement
  return [value];
}
```

Classes and objects

A "class" is a collection of related variables and functions

Note$_1$: constructors in derived classes do not call constructors in base classes! You must do this, if you want this behavior.

Note$_2$: all members are public

Class definition

```
class name [extends base]{
  var name;                  declare member variables
  function name {...}        function declarations
};                           end of class definition
```

Using classes

create new instance of class	*var* = new *name*[(*arg*,...)]
accessing member variable	*object*->*member*
calling member function	*object*->*fnc*([*arg*,...])
current object, from within class	$this
to reference parent class from base class	parent::
to access members without class instance	::

Operators (decreasing precedence)

new operator	new
array member accessor	[]
not [logical operator]	!
ones compliment [bit operator]	~
increment, decrement	++, --
error control operator	@
multiply, divide, modulus (remainder)	*, /, %
addition, subtraction	+, -, .
left, right shift [bit operations]	<<, >>
comparison operators	>, >=, <, <=
equality operators	==, !=, ===, !==
bitwise and	&
bitwise exclusive-or (xor)	^
bitwise or (inclusive-or)	|
logical and	&&
logical or	||
conditional expression	$expr_1$?$expr_2$:$expr_3$
assignment operators	=, +=, -=, *=, ...
print operation	print
logical and	and
logical xor (exclusive-or)	xor
logical or (inclusive-or)	or
list operator	,

Predefined Apache variables

More commonly used variables defined by Apache web server are listed below. Run **phpinfo()** for a complete list.

$SERVER_NAME	name of the web server
$SERVER_SOFTWARE	server ID string, used in HTTP response
$SERVER_PROTOCOL	HTTP protocol used to request page
$REQUEST_METHOD	request method: GET, HEAD, POST, PUT
$QUERY_STRING	query string via which page was accessed
$DOCUMENT_ROOT	root directory under which script is running
$HTTP_REFERER	referring URL
$HTTP_USER_AGENT	user's browser string
$HTTP_REMOTE_ADDR	user's IP address
$HTTP_REMOTE_PORT	port used on user's machine
$SCRIPT_FILENAME	absolute path name of script
$SERVER_ADMIN	server administrator's e-mail address
$SERVER_PORT	port on server; e.g. HTTP 80, HTTPS 443
$PATH_TRANSLATED	path of script relative to filesystem
$SCRIPT_NAME	path of script relative to document root
$REQUEST_URI	the requested URI; e.g. '/index.html'

PHP 4 Reference Card

String Functions <string>

return specific character	chr(n)
return ASCII value of character	ord(c)
length of string	strlen(s)

String formatting/output

output string(s)	echo(s[,...])
output string	print(s)
output formatted string	printf(s[,*arg*])
return a formatted string	sprintf(s[,*arg*])

String comparison

binary safe case-sensitive compare	strcmp(s_1,s_2)
binary safe case-sensitive compare	strncmp(s_1,s_2,len)
binary safe case-insensitive compare	strcasecmp(s_1,s_2)
binary safe case-insensitive compare	strncasecmp(s_1,s_2,len)

Searching strings

find position of 1st occurrence of char.	strpos(h,n[,offset])
find position of last occurrence of char.	strrpos(h,n)
find first occurrence of string	strstr(h,n)
case-insensitive version of strstr	stristr(h,n)
find last occurrence of char.	strrchr(h,n)

String manipulation

convert to upper/lower case	strtoupper(s)/strtolower(s)
trim whitespace from start of string	ltrim(s[,w])
trim whitespace from end of string	rtrim(s[,w])
trim whitespace from start & end	trim(s[,w])
strip HTML&PHP tags from string	strip_tags(s[,allow])
reverse a string	strrev(s)
replace s_1 with s_2 in str	str_replace(s_1,s_2,*str*)
translate characters	strtr(*str*,s_1,s_2)
extract part of a string	substr(s,*start*[,*len*])

Filesystem Functions <filesystem>

open file	fopen(*filename*,*mode*)

modes: r (read from beginning), w (overwrite), a (append)
modifiers: + (open for read & write), b (binary mode)

close file	fclose(*fp*)
retrieve current position in file	ftell(*fp*)
jump to position in file	fseek(*fp*,*offset*[,*whence*])
get next character from file	fgetc(*fp*)
get line from file	fgets(*fp*[,*len*])
read entire file into array	file(*filename*)
test for End Of File	feof(*fp*)
binary-safe file read	fread(*fp*,*len*)
write to file	fputs(*fp*,s,*len*)
flush output buffer	fflush(*fp*)
parse input from file	fscanf(*fp*,*format*[,*var*...])
binary-safe write to file	fwrite(*fp*,s,*len*)
copy a file	copy(*src*,*dest*)
available disk space	diskfreespace(*dir*)
test for existence of file	file_exists(*filename*)
echo all remaining data	fpassthru(*fp*)
is file readable?	is_readable(*filename*)
is file writable?	is_writeable(*filename*)

Mathematical Functions <math>

trig functions	sin(x), cos(x), tan(x)
inverse trig functions	asin(x), acos(x), atan(x)
arctan(y/x)	atan2(y,x)
hyperbolic trig functions	sinh(x), cosh(x), tanh(x)
exponentials & logs	exp(x), log(x), log10(x)
powers	pow(x,y), sqrt(x)
rounding	ceil(x), floor(x), abs(x)
minimum, maximum	min(x,...), max(x,...)
random number	rand(),rand(*min*,*max*)

Unified ODBC Functions <odbc>

connect to data source	odbc_connect(*dsn*,*user*,*pwd*)
close connection(s)	odbc_close(*id*),odbc_close_all()
retrieve last error/msg	odbc_error(),odbc_errormsg()
prepare SQL statement	odbc_prepare(*id*,*query*)
execute prepared SQL statement	odbc_execute(*id*[,*arg*])
prepare & execute SQL statement	odbc_exec(*id*,*query*)
get row as an array	odbc_fetch_into(*id*[,*row*,*result*])
fetch a result row	odbc_fetch_row(*id*[,*row*])
get result from a field	odbc_result(*id*,*field*)
free result resources	odbc_free_result(*id*)
number of rows in result	odbc_num_rows(*id*)
output results in HTML table	odbc_result_all(*id*[,*format*])

Transactions

toggle autocommit on/off	odbc_autocommit(*id*)
commit transaction	odbc_commit(*id*)
rollback transaction	odbc_rollback(*id*)

Session Handling Functions <session>

initialize session data	session_start()
destroy current session data	session_destroy()
get/set session name	session_name([s])
get/set session ID	session_id([s])
register variables in session	session_register(*name*[,...])
unregister variable	session_unregister(*name*)
variable is registered?	session_is_registered(*name*)
get cookie parameters	session_get_cookie_params()
set cookie params	session_set_cookie_params(l[,p[,s]])
write data & close session	session_write_close()

Miscellaneous Functions <misc>

evaluate string as PHP code	eval(s)
terminate script	exit(x),exit(s)

Date/Time functions

format a local date/time	date(format[,timestamp])
current time in secs. since Jan.1, 1970	time()
current time in microseconds	microtime()

External program execution

The following can be used to execute an external program. They differ in their handling of the output.

output returned in result array	exec(prg[,result,status])
output returned as string	shell_exec(prg)
display output	system(prg[,status])
display raw output	passthru(prg[,status])

Reference

PHP web site	http://www.php.net/
Zend Technologies	http://www.zend.com/
PHP Builder	http://www.phpbuilder.com/
Knowledge Base	http://php.faqts.com/
Apache web server	http://httpd.apache.org/

Send comments and corrections to Steven R. Gould, Publishing Writes, Dallas, TX 75252, USA. (sgould@publishingwrites.com)