PHP FOR C PROGRAMMERS
QUICK REFERENCE

By SaEd Mansour

Code Wrappers		The rule: PHP code goes inside the wrappers. **<?php** /*PHP code*/ **?>** <html> <head></head> <body> </body> **<?php** //PHP CODE **?>** </html> **<?php** /*PHP code*/ **?>** Don’t forget: -Files that contain PHP code have the file extension “php”. (filename.php)

Variables	int x = 5;
	double x = 5.5;
	char *x = “hello world”;
		$x = 5;
		$x = 5.5
		$x = “hello world”
		Variables don’t have types.
		Variables aren’t declared.
		Use $ before a variable.

Operators		Arithmetic ,assignment and auto increment/decrement are all the same in PHP.

Output	**printf**(“hello world”);
	
	char* x = “ world”;
	printf(“hello%s”, world);
	
	int x = 5;
	printf(“hello %d”, x);
		echo “hello world”;
		
		$x = “ world”;
		echo “hello$x”;
		
		$x = 5;
		printf(“hello %d”, $x);
		//printf also in php
		
		$x = “world”;
		echo ‘hello $x’;
		(output: hello $x)
		
		Single quotes don’t allow variable substitution.

Arrays	**char arr[] = {**1,2,3,4**}**;
	
	arr[0];
	arr[1];
		$arr = array(1,2,3,4**)**;
		
		$arr[0];
		$arr[1];
		
		Multi types:
		
		$arr = array (1, 2.4444, “string”);

Flow Control (if, for, while)	for (int i = 0; i < 10; i++)
	{
	printf(“what\n”);
	}
	
	If (x > 5)
	{
	x = 7;
	}
	else if(x == 5)
	{
	x = 20;
	}
	else
	{
	x = 1;
	}
	
	while (x < 5)
	{
	x++;
	}
	
	int array[10] = {0,1,2,3,4,5,6,7,8,9};
	
	for(int i = 0; I < 10; i++)
	{
	printf(“%d\n”, array[i]);
	}
		for ($i = 0; $i < 10; $i++)
		{
		echo “what<br />”;
		}
		
		if ($x > 5)
		{
		$x = 7;
		}
		elseif ($x == 5)
		{
		$x = 20;
		}
		else
		{
		$x = 1;
		}
		
		while ($x < 5)
		{
		$x++;
		}
		
		$array = array(0,1,2,3,4,5,7,8,9);
		
		foreach($array as $val)
		{
		echo “${val}<br />”;
		}
		
		Foreach runs over all array elements, each iteration $val gets the value of the next array element.
		
		${val} is used to make it know that the variable isn’t “$val<br” because the variable $val<br doesn’t exist…
		
		continue/ break are also in php.

String Concatenation	char str[10] =”what”;
	strcat(str, “ the!”);
	printf(“%s”, str);
	(output: what the!)
		echo ‘what’ **.** ‘ the!’;
		(output: what the!)
		
		Operator dot attaches the two strings together (concatenation).

Functions	**double mul**(double a, double b)
	{
	  return a * b;
	}
		function mul($a, $b)
		{
		  return $a * $b;
		}

Switch	**s**witch(x)
	{
	case 1:
	    x = 1;
	   break;
	case 2:
	    x = 2;
	    break;
	default:
	     x = 0;
	    break;
	}
		switch($x)
		{
		case 1:
		   $x = 1;
		   break;
		case 2:
		    $x = 2;
		    break;
		default:
		     $x = 0;
		    break;
		}
		
		switch accepts strings:
		
		$scream = “fire!”;
		switch($scream)
		{
		case **“boom!”**:
		    echo “hide!”;
		case **“fire!”**:
		   echo “run away!”;
		}

Comments	// this is a single line comment /* this is a comment */	// this is a single line comment /* this is a comment */ **# this is a single line comment**

Pointers		There are no pointers, use references.
		
		function swap(**&$a**, **&$b**)
		{
		   $temp = $a;
		   $a = $b;
		   $b = $temp;
		}

<table>
<tr><td></td><td></td><td>Function call:
swap($a, $b);

if there was no & in swap’s declaration the values wouldn’t be swapped.</td></tr>
<tr><td colspan="3"><b>Useful functions in php:</b>

<b>require “fileName”;</b>
//includes the text of fileName into the file where the function require was called</td></tr>
</table>