

What's PHP? — Reference Tables

Companion to the book

John Rouda

This booklet collects every reference table from *What's PHP? — Learn It in 8 Hours* in one place, so you can keep it beside the keyboard while you work.

Table P.1 – The book, by the hour

Hour	Chapters	What you'll be able to do
1	1-3	Run PHP from the command line and the browser, store values, and understand types
2	4-6	Work with text and output, make decisions, and repeat things with loops
3	7-8, Checkpoint 1	Write your own functions and manage lists and maps with arrays; first self-test
4	9-12	Read the request, handle a form, query the database with PDO, and render a page
5	13-16, Checkpoint 2	Remember users with sessions and cookies, read and write files, handle dates; second self-test
6	17-20	Classes and objects, inheritance, Composer, and real error handling
7	21-24, Checkpoint 3	Serve JSON to the browser, secure the app, meet the wider world, and build the capstone; final self-test
8	25-27	Classic bugs, lasting style, the recipe cookbook, and the appendices as your desk reference

Table 1.1 – Milestones in PHP's history

Milestone	Year	Milestone	Year
“Personal Home Page Tools”	1994	PHP 7 (twice as fast, real types)	2015
PHP 3, first real language	1998	PHP 8 (JIT, match, enums)	2020
PHP 5, modern object model	2004	Annual releases (8.1, 8.2, 8.3...)	2021-today

Table 3.1 – PHP’s everyday types

Type	Example	What it holds
string	"Cafe Latte"	Text
int	42	Whole numbers
float	4.50	Numbers with a decimal point
bool	true / false	A yes/no value
array	[2.75, 4.50]	An ordered list or map (Chapter 8)
null	null	The deliberate absence of a value

Table 4.1 – Everyday string functions

Function	Does	Example result
strlen(\$s)	Length in bytes	strlen("latte") → 5
strtoupper / strtolower	Change case	strtoupper("hi") → HI
trim(\$s)	Remove surrounding whitespace	trim(" hi ") → hi
str_replace(\$a,\$b,\$s)	Replace text	str_replace("a","4","java") → j4v4
str_contains(\$h,\$n)	Is one string inside another?	str_contains("latte","att") → true
explode(\$sep,\$s)	Split into an array	explode(", ", "a,b") → ["a","b"]

Function	Does	Example result
<code>implode(\$sep,\$arr)</code>	Join an array into a string	<code>implode("-", ["a","b"]) → a-b</code>

Table 8.1 – Everyday array functions

Function	Does
<code>count(\$a)</code>	Number of elements
<code>in_array(\$v, \$a)</code>	Is a value present?
<code>array_keys(\$a) / array_values(\$a)</code>	Just the keys / just the values
<code>array_sum(\$a)</code>	Total of a numeric list
<code>sort / rsort / usort</code>	Sort in place (ascending / descending / custom)
<code>array_column(\$a, "price")</code>	Pull one field out of a list of maps

Table 9.1 – The superglobals you’ll use

Superglobal	Holds
<code>\$_GET</code>	Values from the URL’s query string (<code>?category=drink</code>)
<code>\$_POST</code>	Values submitted in a form with <code>method="post"</code>
<code>\$_REQUEST</code>	<code>\$_GET</code> and <code>\$_POST</code> combined (avoid; be specific instead)
<code>\$_SERVER</code>	Info about the request and server (method, path, headers)
<code>\$_COOKIE</code>	Cookies the browser sent (Chapter 14)
<code>\$_SESSION</code>	Per-visitor server-side storage (Chapter 13)

Superglobal	Holds
<code>\$_FILES</code>	Uploaded files (Chapter 15)

Table 14.1 – Cookie vs. session

	Cookie	Session
Data lives	In the browser	On the server
Visitor can edit?	Yes	No
Good for	Preferences, “remember me” token	Login state, cart, anything sensitive
Lifespan	Whatever expiry you set	Until browser closes (by default)
Size limit	~4 KB	Practically unlimited

Table 22.1 – The core attacks and their defenses

Attack	Defense
SQL injection	Prepared statements — never concatenate input into SQL
XSS	<code>htmlspecialchars()</code> on all outside data at output
Stolen password database	<code>password_hash()</code> / <code>password_verify()</code> ; never plain text or MD5
CSRF	Per-session token on every state-changing form; <code>hash_equals()</code>
Eavesdropping	HTTPS everywhere
Leaked secrets	Environment variables, never in code or git