ScopeTreasure Courses

PHP COURSES · LESSON PH.3 · CODE SCOPE SERIES

Data Types and Operators

Text, whole numbers, decimals and true/false — learn the four main types, how to check a type, and why . joins text while + adds numbers.

Introduction

A variable holds a value, and every value has a **type** — what kind of thing it is. PHP has four types beginners use every day: `string` (text), `int` (whole numbers), `float` (numbers with a decimal point) and `bool` (true or false). There are a few more, but these four carry almost every program.

PHP decides the type from the value, not from a declaration. Write `$age = 20;` and PHP knows it is an integer. Write `$name = "Ali";` and it is a string. That freedom is handy — and it is also where the classic beginner bug lives: `"5" + 2` is 7, because PHP quietly turns the text "5" into the number 5 when you add. Helpful, until it surprises you.

Operators are the symbols that do things to values: `+ - * / %` for maths, `.` for joining text, `=` for storing, and comparison operators for asking questions. The one to tattoo on your memory is the dot: in PHP, `.` joins strings. In JavaScript the same job is done by `+`, which is why learners coming from JavaScript keep writing `"Ali" + "!"` and getting `0`.

(Roman Urdu: Type matlab value ki kisam — text, poora number, decimal, ya sach/jhoot. `.` text jorta hai, `+` number jama karta hai — ye farq yaad rakhna.)

Key points

- **The four everyday types:** `string` (text in quotes), `int` (whole numbers), `float` (decimals), `bool` (`true` / `false`).
- **`var_dump($x)` prints a value AND its type.** It is the fastest way to see what PHP thinks a value is.
- **`gettype($x)` returns the type name as text**, like `"integer"` or `"string"`.
- **Maths operators:** `+` add, `-` subtract, `*` multiply, `/` divide, `%` remainder, `**` power.
- **`.` joins strings.** `"Ali" . "!"` gives `Ali!`. This is the concatenation operator.
- **`+` on strings tries to add numbers.** `"5" + 2` is `7`; `"Ali" + 2` is `2` with a warning, because "Ali" becomes 0.
- **Comparison operators:** `==` (equal value), `===` (equal value AND type), `!=` (not equal), `>` `<` `>=` `<=`.
- **`==` vs `===` matters.** `5 == "5"` is true (PHP converts); `5 === "5"` is false (different types).
- **Assignment shortcuts exist:** `$x += 5` means `$x = $x + 5`; `$name .= "!"` appends to a string.
- **`is_string()`, `is_int()`, `is_float()`, `is_bool()`** ask "is this value this type?" and answer true or false.

Examples

Example 1 — Seeing the four types with var_dump

```
<?php
$name = "Ali";
$age = 20;
$height = 5.9;
$isStudent = true;
var_dump($name);
var_dump($age);
var_dump($height);
var_dump($isStudent);
?>
```

What you will get:

```
string(3) "Ali"
int(20)
float(5.9)
bool(true)
```

`var_dump` shows the type, and for a string also its length in characters. Keep it in your pocket — it settles every "what type is this?" argument.

Example 2 — The dot joins, the plus adds

```
<?php
echo "Ali" . " " . "Ahmed";
echo "<br>";
echo 5 + 2;
echo "<br>";
echo "5" + 2;
echo "<br>";
echo "5" . 2;
?>
```

What you will get:

```
Ali Ahmed
7
7
52
```

`"5" + 2` adds numbers (7), but `"5" . 2` joins text ("52"). Same values, different operator, different answer. When a result looks wrong, check the operator first.

Example 3 — Remainder and power

```
<?php
echo 10 % 3;
echo "<br>";
echo 2 ** 3;
echo "<br>";
echo 7 / 2;
?>
```

What you will get:

```
1
8
3.5
```

`%` gives the remainder after division (10 ÷ 3 leaves 1). `**` is power (2 × 2 × 2). Division of integers can give a float — `7 / 2` is `3.5`, not `3`.

Example 4 — == forgives the type, === does not

```
<?php
var_dump(5 == "5");
var_dump(5 === "5");
var_dump(5 != "6");
?>
```

What you will get:

```
bool(true)
bool(false)
bool(true)
```

`==` only checks the value, so 5 equals "5". `===` checks value AND type, so the integer 5 is not the string "5". In real code, prefer `===` — it catches the type surprises before your users do.

Example 5 — Assignment shortcuts

```
<?php
$cups = 0;
$cups += 3;
echo $cups;
echo "<br>";
$greeting = "Hello";
$greeting .= ", Ali!";
echo $greeting;
?>
```

What you will get:

```
3
Hello, Ali!
```

`+=` adds and stores in one step. `.=` joins and stores — the string version of the same idea.

Quick recap

- Four main types: string, int, float, bool; `var_dump` shows value and type.
- `.` joins text; `+` adds numbers — never mix them up.
- `%` is the remainder; `**` is power; `/` can give a decimal.
- `==` compares values; `===` compares value AND type — prefer `===`.
- `$x += 5` and `$str .= "!"` are shortcuts for changing a value in place.

Common mistakes

1. **Joining text with `+`** — `"Hello" + $name` gives 0, not a greeting. Fix: use `.` for text.
2. **Adding with `.`** — `"5" . "2"` gives "52", not 7. Fix: use `+` for numbers.
3. **Using `==` when the type matters** — `0 == "Ali"` can behave unexpectedly. Fix: use `===` unless you truly want conversion.
4. **Trusting a form value's type** — form input always arrives as a string, so `"90" + 10` works but `"90" === 90` is false. Fix: cast it first, `(int) $_POST["marks"]`.
5. **Forgetting that `/` gives a float** — `7 / 2` is 3.5, and code expecting 3 breaks. Fix: use `intdiv(7, 2)` for a whole-number answer, or round it.
6. **Reading `var_dump` output as an error** — it is a debugging tool, not a crash. Fix: remove `var_dump` lines before the page goes live.

Practice

1. Make four variables (one of each type) and `var_dump` all four — write down each type.
2. Print `"5" + 2` and `"5" . 2` on one page and write down why the answers differ.
3. Work out 17 % 5 and 3 ** 4 by hand first, then check with PHP.
4. Test `var_dump(10 == "10");` and `var_dump(10 === "10");` — write down the difference in one line.
5. Start `$total = 100;`, then use `+=` to add 50 and `.=` on a string to build "Total: 150" — print both.
6. Use `is_int()` on 20 and on "20" and print both answers — explain the difference in one line.
7. Store `$a = "12"; $b = 3;` and print `$a + $b`, `$a . $b`, and `$a * $b` — predict each answer before you run it.

Practice quiz

1. What does the dot . do in PHP?

- It adds two numbers
- It joins two strings together
- It ends a statement
- It starts a comment

Answer: It joins two strings together

2. What is "5" + 2 in PHP?

- "52"
- 7, because + adds numbers and PHP turns "5" into 5
- An error
- 3

Answer: 7, because + adds numbers and PHP turns "5" into 5

3. What is the difference between == and ===?

- They are the same
- == checks the value only; === checks the value AND the type
- === checks the value only; == checks value and type
- == is for text, === is for numbers

Answer: == checks the value only; === checks the value AND the type

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/ph-data-types-operators · More free tools and notes at https://scopetreasure.com