ScopeTreasure Courses

PHP COURSES · LESSON PH.4 · CODE SCOPE SERIES

Conditionals: if, else and switch

Programs make decisions with if, elseif and else — learn comparisons, the difference between = and == and ===, and switch for many fixed choices.

Introduction

So far, every program ran every line from top to bottom. Real programs do not work like that. A program must decide: if the marks are 50 or more, print "Pass"; otherwise, print "Fail". If the user is logged in, show the page; otherwise, show a login message. Decisions are what make programs feel smart.

PHP makes decisions with `if`, `elseif` and `else`. You write a **condition** — a test that is either `true` or `false` — and PHP runs only the block that matches. The block is the lines between the braces `{ }`.

For one value matched against many fixed options, PHP also has `switch` — cleaner than a long ladder of `if` statements when you are checking the same variable again and again.

(Roman Urdu: If ka matlab hai "agar" — agar shart poori ho to aik kaam karo, warna doosra. `elseif` matlab "warna agar", aur `else` matlab "warna".)

Decision code goes wrong in two quiet ways. First, a condition that is almost right: `if ($age = 18)` stores 18 instead of asking a question, and it is always true. Second, an `elseif` chain checked in the wrong order, so a loose test catches everyone before the strict test is ever reached. Read each condition out loud as PHP reads it, not as a person would.

Key points

- **A condition is a test with a true or false answer.** Examples: `$marks >= 50`, `$age == 18`, `$name == "Ali"`.
- **Comparison operators:** `==` (equal value), `===` (equal value and type), `!=` (not equal), `>` `<` `>=` `<=`.
- **`=` stores, `==` compares.** `$age = 20` stores 20; `$age == 20` asks "is age 20?". This is the most common beginner mix-up.
- **`if` runs its block only when the condition is true.** Otherwise PHP skips to the next part.
- **`else` catches everything the `if` did not catch.** It has no condition of its own.
- **`elseif` chains several conditions in order.** PHP checks them one by one and runs only the first block that is true.
- **`&&` needs both tests true.** `||` needs at least one test true. `!` flips true to false and false to true.
- **Every block needs braces `{ }`.** Without them, only the very next line belongs to the `if`.
- **`switch` checks one value against many cases.** Each `case` needs a `break`, or the code falls through into the next case.

- **Check the strictest condition first in a chain.** In a grade ladder, test `>= 90` before `>= 50`, or everyone passes the first test.

Examples

Example 1 — Pass or fail

```
<?php
$marks = 65;
if ($marks >= 50) {
    echo "Pass";
} else {
    echo "Fail";
}
?>
```

What you will get:

```
Pass
```

Change `$marks` to 40 and run it again — you will get `Fail`, because the condition `$marks >= 50` is now false and PHP runs the `else` block instead.

Example 2 — Grades with an elseif chain

```
<?php
$marks = 82;
if ($marks >= 90) {
    echo "Grade: A";
} elseif ($marks >= 75) {
    echo "Grade: B";
} elseif ($marks >= 50) {
    echo "Grade: C";
} else {
    echo "Grade: F";
}
?>
```

What you will get:

```
Grade: B
```

PHP checks from the top: 82 is not 90 or more, so it skips the first block; 82 is 75 or more, so it runs the second block and then skips the rest. Only one block ever runs in a chain.

Example 3 — Combining conditions with &&, || and !

```
<?php
$age = 20;
$hasTicket = true;
if ($age >= 18 && $hasTicket) {
    echo "You can enter<br>";
}
if ($age < 18 || !$hasTicket) {
    echo "You cannot enter<br>";
}
$day = "Sunday";
if ($day == "Saturday" || $day == "Sunday") {
    echo "It is the weekend";
}
?>
```

What you will get:

```
You can enter
It is the weekend
```

The second `if` is false here (age is 20 and the ticket exists), so its message does not print. The weekend test is true because `$day` equals `"Sunday"`, even though it does not equal `"Saturday"` — `||` needs only one side true.

Example 4 — switch: one value, many fixed options

```
<?php
$day = 3;
switch ($day) {
    case 1:
        echo "Monday";
        break;
    case 2:
        echo "Tuesday";
        break;
    case 3:
        echo "Wednesday";
        break;
    default:
        echo "Another day";
}
?>
```

What you will get:

```
Wednesday
```

`switch` compares `$day` with each case in turn. Without `break`, PHP would keep running into the next case — that fall-through is the classic switch bug, so write `break` every time until you deliberately want it.

Example 5 — A nested decision: teen with or without ID

```
<?php
$age = 16;
$hasId = false;
if ($age >= 18) {
    echo "Adult";
} elseif ($age >= 13) {
    if ($hasId) {
        echo "Teen with ID";
    } else {
        echo "Teen without ID";
    }
} else {
    echo "Child";
}
?>
```

What you will get:

```
Teen without ID
```

Age 16 reaches the `elseif`, and inside it the inner `if` checks the ID. Change `$hasId` to `true` and run it again to see the other branch. Keep nesting shallow — two levels is plenty; deeper nesting gets hard to read.

Example 6 — Real-world case: a shop discount

```
<?php
$bill = 6000;
$isStudent = true;
if ($bill >= 5000 || $isStudent) {
    echo "Discount: 10%<br>";
    echo "You pay: " . ($bill - $bill * 10 / 100);
} else {
    echo "No discount";
}
?>
```

What you will get:

```
Discount: 10%
You pay: 5400
```

The bill is 6000, 10% of it is 600, and 6000 − 600 = 5400. Change `$bill` to 3000 and `$isStudent` to `false`, and the program correctly prints "No discount" — test both paths, not just the happy one.

Quick recap

- `if` runs a block when its condition is true; `else` runs when it is not.
- `elseif` chains several conditions; only the first true block runs.
- `==` compares value; `===` compares value and type; `=` stores. Never mix them up.
- `&&` needs both true; `||` needs one true; `!` flips a value.
- `switch` matches one value against many cases — never forget `break`.
- Braces `{ }` mark the block; the strictest test goes first.

Common mistakes

1. **Writing `=` instead of `==` in a condition** — `if ($age = 18)` stores 18 and is always true. Fix: always use `==` (or `===`) when comparing.
2. **Forgetting `break` in a switch case** — the code falls through and runs the next cases too. Fix: end every case with `break` (except a deliberate fall-through, which deserves a comment).
3. **Checking conditions in the wrong order** — in a grade chain, testing `>= 50` before `>= 90` means everyone passes the first test and never reaches A. Fix: test the strictest (highest) condition first.
4. **Writing `if ($age == 18 || 20)`** — PHP reads this as two separate tests, and `20` alone is always truthy, so the test is always true. Fix: write the full test twice: `if ($age == 18 || $age == 20)`.
5. **Comparing text with a number using `===`** — `"50" === 50` is false, because the types differ. Fix: cast first, for example `if ((int) "50" === 50)`.
6. **Giving `else` a condition** — `else ($marks < 50)` is an error; `else` catches everything left over and takes no test of its own. Fix: use `elseif ($marks < 50)` if you need a test, or plain `else`.
7. **Leaving out the braces** — without `{ }`, only the first line after `if` belongs to it, and the next line runs always. Fix: always use braces, even for one line.

Practice

1. Write a program that stores marks in a variable and prints "Pass" or "Fail" using 50 as the pass mark. Test it with 45 and with 75.
2. Extend it into a grade chain: A for 90+, B for 75+, C for 50+, F below 50. Test it with 92, 80, 60 and 30.

3. Write a test for a discount: a shopper gets a discount if their age is under 18 OR their total bill is over 5000. Test three different shoppers.
4. Deliberately write `if ($age = 20)` and read what happens. Then fix it to `==` and write down, in one line, what the mistake taught you.
5. Ticket prices: under 12 pays 100, 12 to 59 pays 300, 60 and over pays 150. Write the chain, and test it with ages 8, 30 and 65.
6. Write a `switch` that prints the day name for numbers 1 to 7, with a `default` for anything else. Test it with 1, 5 and 9.
7. A library fines a late book 20 rupees per day, but students pay half. Write a program with variables for days late and student status, and print the fine for 5 days late, student and non-student.

Practice quiz

1. What is wrong with if ($age = 18)?

- Nothing, it asks whether age is 18
- It stores 18 in $age instead of asking a question, and is always true
- It is a syntax error
- It only works for ages over 18

Answer: It stores 18 in $age instead of asking a question, and is always true

2. In a grade chain, which condition must be tested first?

- The lowest, >= 50
- The strictest, >= 90, or everyone passes the first test
- The middle one
- The order does not matter

Answer: The strictest, >= 90, or everyone passes the first test

3. What happens in a switch case if you forget break?

- Nothing, break is optional decoration
- The code falls through and runs the next cases too
- The switch stops with an error
- It jumps back to the first case

Answer: The code falls through and runs the next cases too

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-conditionals · More free tools and notes at https://scopetreasure.com