
PHP
Client-side language
Server-side language
Advantages
Applications
Variables
Datatypes
Operators

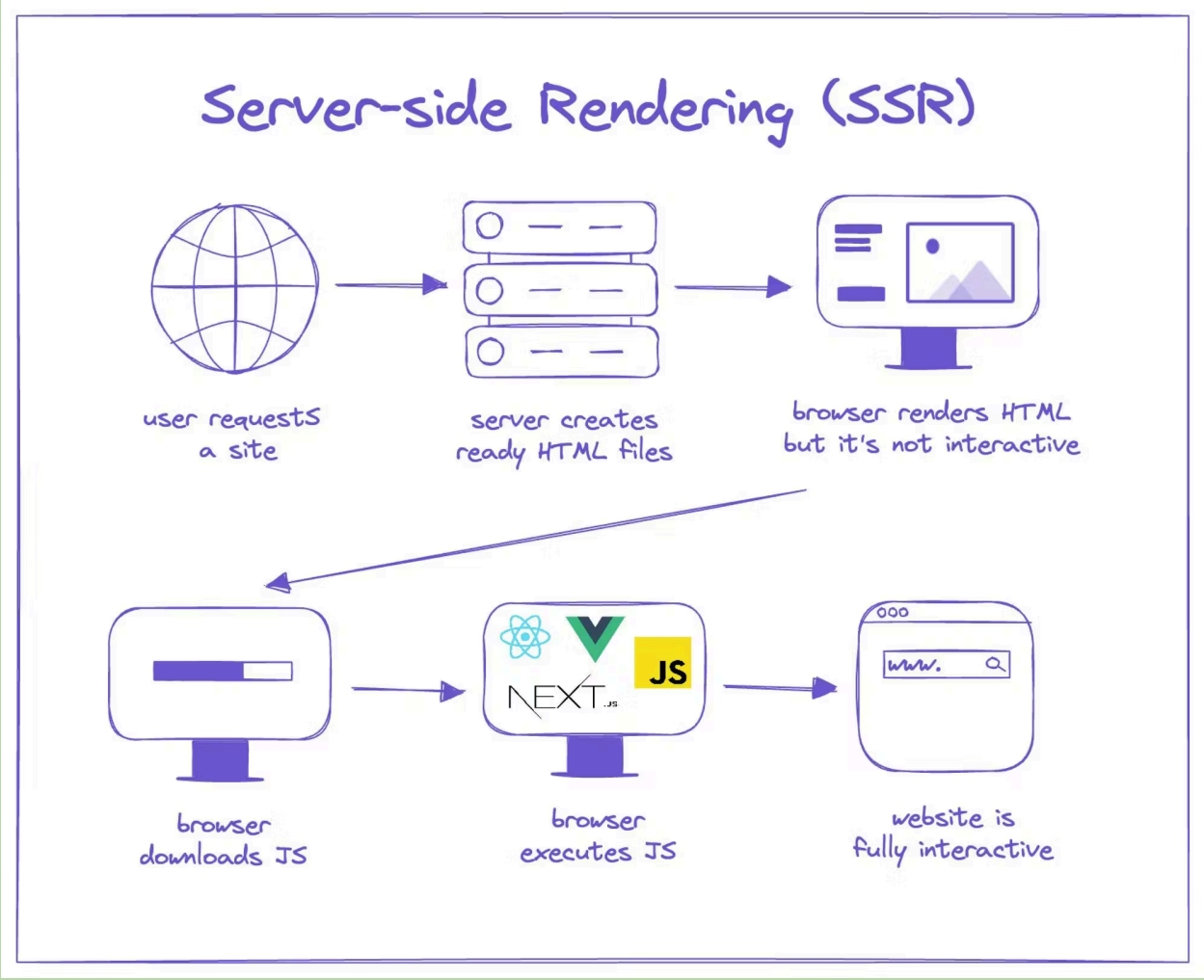
1. Introduction to Client-side vs Server-side Languages

Definition

- Client-side languages: Code that runs in the user's browser (frontend). Examples: HTML, CSS, JavaScript (and frameworks like React, Vue, Angular).
- Server-side languages: Code that runs on the web server (backend). Handles business logic, database operations, authentication, file handling, API responses. Examples: PHP, Node.js, Python (Django/Flask), Java (Spring Boot), Ruby on Rails, Go, C# (.NET).

How it works Browser → sends HTTP request → Web server (Apache/Nginx) executes server-side code → generates HTML/response → sends back to browser → browser renders client-side code.

Explanation:



Key Differences

Aspect	Client-side	Server-side
Execution location	User's browser	Web server
Languages	HTML, CSS, JS	PHP, Node.js, Python, Java, etc.
Access to resources	Limited (no direct DB access)	Full (DB, files, APIs, secrets)
Security exposure	Visible to the user	Hidden (but vulnerable if not secured)
Performance impact	Affects user's device	Affects server load & response time

Key History & Evolution

- 1990s: Static HTML → CGI scripts (Perl) → Server-side scripting born
- 1995–2005: PHP, ASP, JSP dominant
- 2010s: Node.js (JavaScript everywhere), Python/Django rise
- 2020s–2026: Full-stack JS (Next.js), serverless (Vercel, AWS Lambda), edge computing

2. Introduction to PHP

Definition: PHP (Hypertext Preprocessor) is an open-source, server-side scripting language designed for web development. It is embedded into HTML and executed on the server.

History Timeline:

- 1994: Created by Rasmus Lerdorf as "Personal Home Page Tools"
- 1995: PHP 1.0
- 1998: PHP 3.0 (Zend Engine introduced)
- 2000: PHP 4.0
- 2004: PHP 5.0 (full OOP support)
- 2015: PHP 7.0 (massive performance boost)
- 2020: PHP 8.0 (JIT compiler, union types, attributes, named arguments)
- Current (Jan 2026): PHP 8.5.2 (released 15 Jan 2026 – bug-fix release)
- PHP 8.4 is still under security support until Dec 2026
- PHP 8.1 reached EOL on 31 Dec 2025

Current Usage: PHP powers 72.2% of all websites with a known server-side language.

Working Procedure: .php file → Apache/Nginx + PHP-FPM/Mod_php → Server executes PHP → outputs HTML/JSON → sent to browser.

Popular Tools:

- Frameworks: Laravel (most loved), Symfony
- CMS: WordPress (≈43% of web), Drupal, Joomla
- Servers: Nginx + PHP-FPM (recommended), Apache
- IDEs: PhpStorm, VS Code + Intelephense

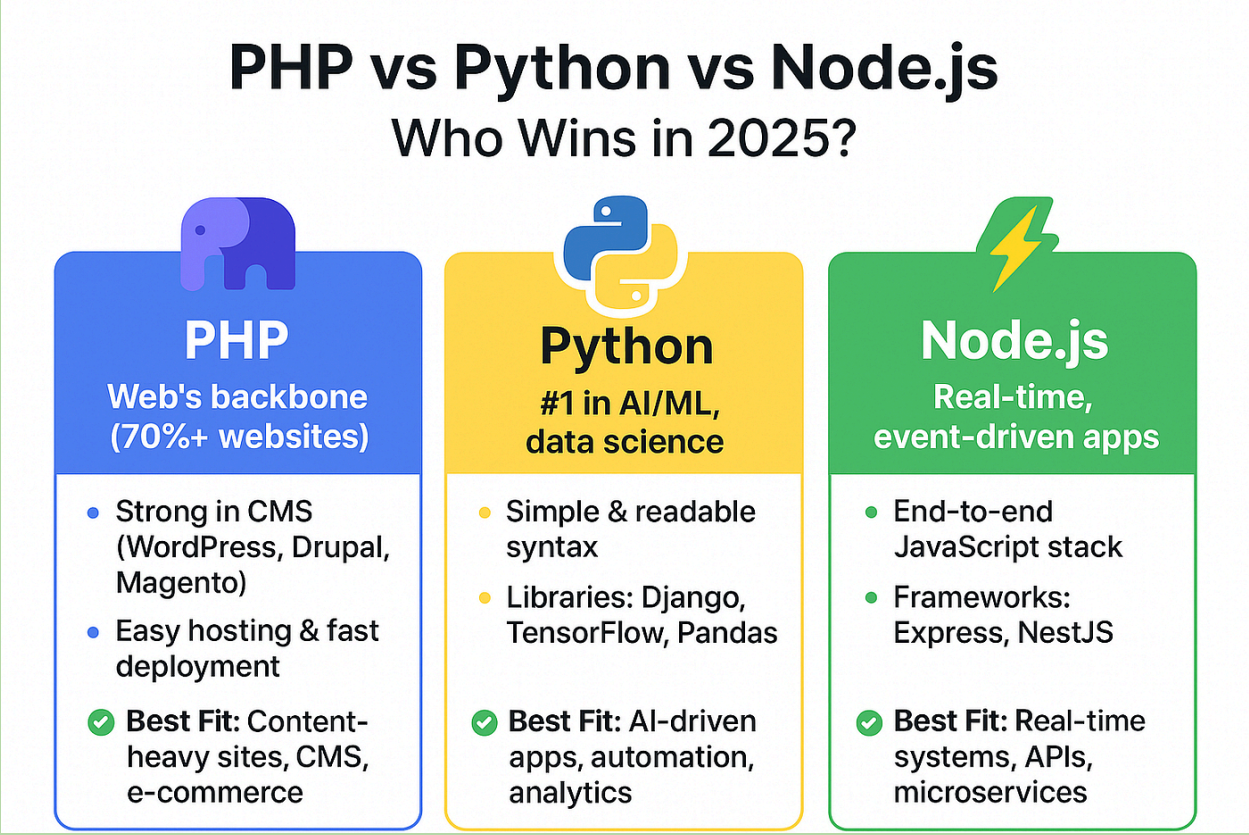
3. Advantages of PHP

Major Advantages:

1. Easy to learn & rapid development – Simple syntax, embedded in HTML
2. Excellent hosting support – Runs on almost every shared hosting (cheap & easy deployment)
3. Huge ecosystem – WordPress, Laravel, Composer packages
4. Performance – PHP 8+ with JIT is very fast for web workloads
5. Cross-platform – Windows, Linux, macOS
6. Strong community & documentation
7. Built-in database support (MySQLi, PDO)
8. Mature security tools, when used correctly (password_hash, prepared statements)

Differences vs Others:

- vs Node.js: PHP is synchronous by default (easier for beginners), Node.js excels in real-time (WebSockets)
- vs Python: PHP is web-first & faster deployment; Python better for data science/ML
- vs Java: PHP is lightweight & quick to start; Java better for large enterprise apps



4. Applications of PHP

- Content Management Systems (WordPress powers 43%+ of websites)
- E-commerce (WooCommerce, Magento, Laravel-based shops)
- RESTful / GraphQL APIs (Laravel Sanctum, Lumen)
- SaaS applications
- Social networking platforms
- Enterprise web apps (Symfony)
- Serverless (Bref on AWS Lambda)
- Real-time apps with Laravel Octane + Swoole/RoadRunner

High-traffic examples: Wikipedia, Facebook (early versions), Slack backend parts, Etsy, etc.

5. Variables, Data Types, and Operators in PHP

Variables

- Start with $ (e.g., $name)
- Loosely typed (no need to declare type)
- Case-sensitive
- Superglobals: $_GET, $_POST, $_SESSION, $_COOKIE, $_FILES, $_SERVER

Data Types (PHP 8.5)

Type	Example	Notes
String	`$name = "Alice";`	Single or double quotes
Integer	`$age = 25;`	Whole numbers
Float	`$price = 19.99;`	Decimal numbers
Boolean	`$is_active = true;`	true/false
Array	`$fruits = ["apple", "banana"];`	Indexed or associative
Object	`$user = new stdClass();`	Instances of classes
NULL	`$var = null;`	No value
Resource	File handles, DB connections	Special external resources

Type declarations (PHP 7+): function add(int $a, int $b): int

Operators

- Arithmetic: +, -, *, /, %, ** (power)
- Comparison: ==, === (strict), !=, !==, >, <, >=, <=, <=> (spaceship)
- Logical: &&, ||, !, and, or, xor
- Assignment: =, +=, -=, etc.
- Concatenation: .
- Ternary: $status = $age >= 18 ? "Adult" : "Minor";
- Null coalescing: ?? and ??=
- Nullsafe: ?-> (PHP 8+)

Examples:

```
<?php

// Variables & Data Types

$name = "Raju";                  // String

$age = 28;                       // Integer

$salary = 75000.50;              // Float

$is_developer = true;            // Boolean

$skills = ["PHP", "SQL", "Laravel"]; // Array

echo "Hello, my name is $name and I am $age years old.<br>";

// Operators

$bonus = 5000;

$total = $salary + $bonus;       // Arithmetic

$status = ($age >= 18) ? "Adult" : "Minor";

if ($is_developer && in_array("PHP", $skills)) {

    echo "You are eligible for the role!";  }

// Superglobal example (form data)

if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    $email = filter_var($_POST['email'], FILTER_SANITIZE_EMAIL); }
```