50 PHP Interview Questions — Detailed Answers (English + Hindi)

This document contains 50 theoretical PHP interview questions with detailed explanations in both English and Hindi. Use this for study, interview prep, or as a reference. If you want this exported as a downloadable PDF, or converted into an SEO-optimized blog or bilingual booklet, tell me and I'll prepare it.

---

1. What is PHP and how does it work on the server?

English: PHP (Hypertext Preprocessor) is a server-side scripting language designed primarily for web development. A PHP file is processed on the web server by the PHP interpreter which executes PHP code and produces HTML (or other output) that is sent to the client browser. Typical flow: client requests a .php page → web server (e.g., Apache, Nginx) passes it to PHP engine (mod_php, PHP-FPM) → PHP runs code, may access database/files → generates output (HTML/JSON) → server sends response to client.

Hindi: PHP (हाइपरटेक्स्ट प्रीप्रोसेसर) एक सर्वर-साइड स्क्रिप्टिंग भाषा है जो मुख्यतः वेब डेवलपमेंट के लिए उपयोग होती है। जब कोई क्लाइंट .php फ़ाइल का अनुरोध करता है, तो वेब सर्वर PHP इंटरप्रेटर को वह फ़ाइल भेजता है। PHP कोड चलता है, डेटाबेस/फाइलों तक पहुँच सकता है और HTML/JSON जैसा आउटपुट बनाकर सर्वर के माध्यम से क्लाइंट को भेज देता है।

---

2. What is the difference between compiled and interpreted languages?

English: A compiled language translates source code into machine code ahead of time (e.g., C), producing an executable. An interpreted language translates and executes code on the fly at runtime (e.g., classic PHP). Modern PHP uses an intermediate compilation step: PHP source is compiled into opcodes (bytecode) by the Zend Engine, then these opcodes are executed. Opcode caches (like OPcache) store compiled opcodes to improve performance.

Hindi: कम्पाइल की गई भाषा पहले से सोर्स को मशीन कोड में बदल देती है और एक executable बनाती है। इंटरप्रेटेड भाषा रनटाइम पर कोड को पढ़कर निष्पादित करती है। PHP पारंपरिक रूप से इंटरप्रेटेड है, परन्तु इसकी अंदरूनी प्रक्रिया में स्रोत को पहले ऑपकोड (बाइटकोड) में बदला जाता है जिसे Zend Engine चलाती है। OPcache जैसी कैशिंग उस ऑपकोड को स्टोर कर प्रदर्शन बढ़ाती हैं।

---

3. Explain the PHP request lifecycle.

English: The lifecycle of a PHP request typically: (1) Client sends HTTP request; (2) Web server receives request and determines handler; (3) PHP interpreter (mod_php/PHP-FPM) executes PHP script; (4) PHP executes code, may interact with database, filesystem, sessions; (5) PHP outputs response (HTML/JSON); (6) Server sends response back; (7) Request ends and PHP cleans up (destructors, session write-close). Understanding lifecycle helps with performance, session handling, and resource cleanup.

Hindi: PHP रिक्वेस्ट का सामान्य जीवनचक्र: (1) क्लाइंट HTTP अनुरोध भेजता है, (2) वेब सर्वर अनुरोध प्राप्त कर हैंडलर तय करता है, (3) PHP इंटरप्रेटर स्क्रिप्ट चलाता है, (4) स्क्रिप्ट DB/फाइल/सेशन से इंटरैक्ट कर सकती है, (5) आउटपुट बनता है और सर्वर को भेजा जाता है, (6) सर्वर क्लाइंट को रिस्पॉन्स भेजता है, (7) रिक्वेस्ट खत्म होते ही PHP क्लीनअप करता है (डिस्ट्रक्टर, सेशन write-close)।

---

4. What are PHP configuration files (php.ini) and why are they important?

English: `php.ini` is the main configuration file controlling PHP settings: memory_limit, max_execution_time, error_reporting, display_errors, upload_max_filesize, session settings, extensions, and more. System administrators tune php.ini for security (disable functions, control file uploads), performance, and environment differences (development vs production). There can be multiple ini files (per-SAPI) and overrides (.htaccess, ini_set()).

Hindi: `php.ini` PHP की मुख्य कॉन्फ़िग फाइल है जो memory_limit, max_execution_time, error_reporting, upload_max_filesize, session सेटिंग्स और एक्सटेंशन्स जैसी सेटिंग्स को नियंत्रित करती है। PHP को सुरक्षा, प्रदर्शन और डेवलप/प्रोडक्शन वातावरण के अनुसार php.ini के माध्यम से ट्यून किया जाता है। स्थानीय रूप से भी ini_set() या .htaccess से ओवरराइड किया जा सकता है।

---

5. What is the difference between echo, print, and print_r?

English: `echo` and `print` both output strings; `echo` can take multiple parameters (comma-separated) and is marginally faster; it does not return a value. `print` returns 1 (so can be used in expressions) and takes a single argument. `print_r()` is for human-readable representation of arrays/objects — it prints structure, keys, and values (useful for debugging). For machine-readable output use `var_export()` or `json_encode()`.

Hindi: `echo` और `print` दोनों स्ट्रिंग आउटपुट करते हैं; `echo` ज्यादा तेज और मल्टीपल आर्ग्यूमेंट ले सकता है, जबकि `print` एक वैल्यू लौटाता है। `print_r()` एरे/ऑब्जेक्ट का मानवीय-पठनीय विवरण दिखाने के लिए उपयोगी है (डिबगिंग में)। मशीन रीडेबल आउटपुट के लिए `var_export()` या `json_encode()` बेहतर हैं।

---

6. What is the difference between == and === in PHP?

English: `==` is the loose (type-converting) equality operator — it compares values after type juggling. `===` is strict equality — checks both value and type. Example: `0 == '0'` is true, but `0 === '0'` is false. Prefer `===` to avoid unexpected behavior from type coercion.

Hindi: `==` लूज़ (टाइप कॉन्कन्टिंग) तुलना है, यानी PHP आवश्यकतानुसार प्रकार बदलकर तुलना करता है। `===` कड़ाई से तुलना करता है — मूल्य और प्रकार दोनों मिलने चाहिए। उदाहरण: `0 == '0'` true है, पर `0 === '0'` false है। अप्रत्याशित परिणामों से बचने के लिए `===` का प्रयोग बेहतर है।

---

7. What are superglobal variables in PHP?

English: Superglobals are built-in variables accessible in all scopes without `global` keyword: `$_GET`, `$_POST`, `$_REQUEST`, `$_SERVER`, `$_COOKIE`, `$_SESSION`, `$_FILES`, `$_ENV`, `$GLOBALS`. They provide request data, server info, uploaded files, environment variables, and global symbol table. Use them carefully and sanitize input before use.

Hindi: सुपरग्लोबल PHP के बिल्ट-इन वेरिएबल हैं जो किसी भी स्कोप में सीधे उपलब्ध होते हैं: `$_GET`, `$_POST`, `$_REQUEST`, `$_SERVER`, `$_COOKIE`, `$_SESSION`, `$_FILES`, `$_ENV`, `$GLOBALS`। ये अनुरोध डेटा, सर्वर जानकारी, अपलोड फाइलें और पर्यावरण चर प्रदान करते हैं। इन्हें उपयोग करने से पहले sanitize करना जरूरी है।

---

8. What is the difference between GET and POST methods?

English: GET sends data via URL query string — suitable for idempotent requests and bookmarking; limited data length and visible in logs. POST sends data in the request body — better for sensitive or large payloads and operations that change server state. For file uploads you must use POST with `enctype="multipart/form-data"`.

Hindi: GET डेटा को URL क्वेरी स्ट्रिंग के माध्यम से भेजता है — बुकमार्क/कैश के लिए उपयुक्त और सीमित लंबाई का होता है। POST डेटा को रिक्वेस्ट बॉडी में भेजता है — संवेदनशील/बड़े डेटा और सर्वर स्टेट बदलने वाले ऑपरेशन्स के लिए उपयुक्त। फाइल अपलोड के लिए POST के साथ `enctype="multipart/form-data"` का उपयोग करें।

---

9. What is the use of `$_SERVER` superglobal?

English: `$_SERVER` contains server and execution environment information: request method, headers, script path, server name, client IP (`REMOTE_ADDR`), and other details like `REQUEST_URI`, `HTTP_HOST`, `SERVER_SOFTWARE`. It's useful for routing, logging, building URLs, and understanding the runtime context. Be careful: some entries can be spoofed (e.g., headers), so don't trust them blindly for security decisions.

Hindi: `$_SERVER` सर्वर और एग्जीक्यूशन वातावरण की जानकारी रखता है: रिक्वेस्ट मेथड, हेडर, स्क्रिप्ट पाथ, सर्वर नाम, क्लाइंट IP (`REMOTE_ADDR`), `REQUEST_URI`, `HTTP_HOST` आदि। यह राउटिंग, लॉगिंग और रनटाइम संदर्भ समझने में उपयोगी है। कुछ एंट्रीज़ (जैसे हेडर) spoof की जा सकती हैं, इसलिए सुरक्षा के लिए उन पर पूरा भरोसा न करें।

---

10. What are variable variables in PHP?

English: Variable variables let you use the value of one variable as the name of another: `$a = 'foo'; $$a = 'bar';` now `$foo` equals `'bar'`. Syntax can be confusing with complex expressions; use carefully. Useful for dynamic variable names but often better replaced with arrays/associative arrays for clarity and maintainability.

Hindi: वेरिएबल वेरिएबल्स में एक वेरिएबल के मान को दूसरे वेरिएबल के नाम के रूप में इस्तेमाल किया जाता है: `$a = 'foo'; $$a = 'bar';` अब `$foo` का मान `'bar'` होगा। जटिल एक्सप्रेशन्स के साथ सिंटैक्स भ्रमित कर सकता है; अक्सर स्पष्टता के लिए एरे/एसोसिएटिव एरे उपयोग करना बेहतर रहता है।

---

11. What are the data types supported by PHP?

English: PHP supports scalar types: `int`, `float (double)`, `string`, `bool`. Compound types: `array`, `object`, `callable`, `iterable`. Special types: `resource`, `null`. Since PHP 7, scalar type declarations and return type declarations are available. Understanding types and type coercion helps avoid bugs.

Hindi: PHP में scalar प्रकार हैं: `int`, `float`, `string`, `bool`. कंपाउन्ड प्रकार: `array`, `object`, `callable`, `iterable`. विशेष प्रकार: `resource`, `null`। PHP 7 से टाइप डिक्लेरेशन और रिटर्न टाइप की सुविधाएँ आईं। प्रकार और टाइप कर्जिंग को समझना बग्स से बचने में मदद करता है।

---

12. What is type juggling in PHP?

English: Type juggling is PHP's implicit conversion between types during operations (e.g., string to number). Example: `'5' + 3` becomes `8`. While convenient, it can cause unexpected results (`'0' == false` is true). Use strict comparisons (`===`) or enable strict typing for functions to avoid surprises.

Hindi: टाइप जग्लिंग का मतलब PHP द्वारा आवश्यकतानुसार प्रकारों का स्वतः परिवर्तन है (जैसे स्ट्रिंग से नंबर)। उदाहरण: `'5' + 3` = `8`। यह सुविधा कभी-कभी अनपेक्षित परिणाम दे सकती है (`'0' == false` true)। आश्चर्य से बचने के लिए strict comparisons (`===`) या strict typing का उपयोग करें।

---

13. What is strict typing and how do you enable it?

English: Strict typing enforces parameter and return types declared for functions — PHP will throw a TypeError if the types don't match. Enable per-file by adding `declare(strict_types=1);` at the top of a PHP file. Note: it applies only to scalar type declarations and only in that file's scope.

Hindi: Strict typing फ़ंक्शनों में डिक्लेयर किए गए पैरामीटर और रिटर्न टाइपों को सख्ती से लागू करता है — टाइप मेल नहीं होने पर `TypeError` आता है। इसे फ़ाइल के शीर्ष पर `declare(strict_types=1);` लिखकर सक्षम किया जाता है। यह केवल scalar type declarations पर लागू होता है और केवल उसी फ़ाइल में प्रभावी होता है।

---

14. What is the difference between arrays and objects?

English: Arrays are ordered maps (indexed by integers or strings) used for lists or key-value pairs. Objects are instances of classes containing properties and methods and support visibility, inheritance, and behavior. Arrays are great for simple collections; objects encapsulate data and behavior and suit domain modeling.

Hindi: एरे ordered maps होते हैं जो integer या string keys पर इंडेक्स होते हैं और सूची/कुंजी-मूल्य जोड़ों के लिए उपयोगी हैं। ऑब्जेक्ट क्लास के इंस्टेंस होते हैं जिनमें प्रॉपर्टीज और मेथड्स होती हैं, visibility और inheritance का समर्थन करते हैं। सरल कलेक्शन्स के लिए एरे, और domain modeling के लिए ऑब्जेक्ट बेहतर है।

---

15. What is the difference between indexed, associative, and multidimensional arrays?

English: Indexed arrays use numeric keys (0..n). Associative arrays use string keys (`['name' => 'Amit']`). Multidimensional arrays are arrays of arrays (matrix-like structures). PHP arrays are flexible and can mix key types, but consistent structure improves readability and performance.

Hindi: Indexed arrays numeric keys (0..n) उपयोग करते हैं। Associative arrays string keys (`['name' => 'Amit']`) उपयोग करते हैं। Multidimensional arrays arrays के अंदर arrays होते हैं (जैसे मैट्रिक्स)। PHP arrays लचीले होते हैं और key types mix कर सकते हैं, पर सुसंगत संरचना बेहतर रहती है।

---

16. What is the difference between include, require, include_once, and require_once?

English: `include` and `require` both include PHP files. If `include` fails, it raises a warning and script continues; if `require` fails, it raises a fatal error and script halts. `_once` variants ensure a file is included only once (prevent redeclaration). Use `require` for critical dependencies, and `_once` when you must guard against multiple includes.

Hindi: `include` और `require` दोनों फ़ाइलें शामिल करते हैं। `include` नाकाम होने पर warning देता है और स्क्रिप्ट जारी रहती है; `require` नाकाम होने पर fatal error और स्क्रिप्ट रुक जाती है। `_once` वाले एक ही फ़ाइल को केवल एक बार शामिल करते हैं। महत्वपूर्ण निर्भरताओं के लिए `require` और पुनः-शामिल से रोकने के लिए `_once` का उपयोग करें।

---

17. What is variable scope — global, local, static?

English: Local variables exist inside functions. Global variables exist in the global scope; to access them inside functions use `global $var` or `$GLOBALS['var']`. `static` variables inside functions retain their value between calls (without using globals). Understanding scope prevents unintended side-effects and improves modular code.

Hindi: Local वेरिएबल फ़ंक्शन के अंदर मौजूद होते हैं। Global वेरिएबल ग्लोबल स्कोप में होते हैं; फ़ंक्शन के भीतर इन्हें एक्सेस करने के लिए `global $var` या `$GLOBALS['var']` इस्तेमाल करें। `static` वेरिएबल फ़ंक्शन के कॉल्स के बीच अपना मान बनाए रखते हैं। स्कोप समझना अनचाहे side-effects से बचाता है।

---

18. What are anonymous functions (closures) in PHP?

English: Anonymous functions are functions without a name that can be assigned to variables or passed as callbacks. Closures can capture variables from parent scope using `use` keyword: `$fn = function() use ($a) { ... };`. They're widely used for callbacks, functional-style programming, and encapsulating small behavior.

Hindi: Anonymous functions बिना नाम की फ़ंक्शन होती हैं जिन्हें वेरिएबल में असाइन किया जा सकता है या callback के रूप में पास किया जा सकता है। Closures पेरेंट स्कोप की वैरिएबल्स को `use` की मदद से कैप्चर कर सकते हैं: `$fn = function() use ($a) { ... };`। इन्हें callback और छोटे व्यवहार को कैप्सुलेट करने के लिए उपयोग किया जाता है।

---

19. What is recursion in PHP?

English: Recursion is a function calling itself to solve smaller instances of a problem (e.g., factorial, tree traversal). Important considerations: base case to stop recursion, stack depth (risk of stack overflow), and performance — sometimes iterative solutions are preferred for efficiency.

Hindi: Recursion वह प्रक्रिया है जहाँ फ़ंक्शन स्वयं को कॉल करके समस्या के छोटे हिस्सों को हल करता है (जैसे factorial, tree traversal)। जरूरी चीजें: recursion की रोकने वाली base case, स्टैक की गहराई (stack overflow का खतरा), और प्रदर्शन — कई बार iterative समाधान बेहतर होते हैं।

---

20. What is the difference between pass-by-value and pass-by-reference?

English: Pass-by-value gives a function a copy of the variable; changes inside the function don't affect the caller. Pass-by-reference (using `&`) passes the variable itself, so changes persist outside. Example: `function f($x) {}` vs `function f(&$x) {}`. Use references carefully to avoid unexpected side-effects.

Hindi: Pass-by-value में फ़ंक्शन को वेरिएबल की कॉपी मिलती है; फ़ंक्शन के अंदर बदलाव कॉलर पर असर नहीं करते। Pass-by-reference (`&`) में वास्तविक वेरिएबल पास होता है, इसलिए बदलाव बाहर भी दिखेंगे। उदाहरण: `function f($x) {}` बनाम `function f(&$x) {}`। रेफरेंसेज़ का सावधानीपूर्वक उपयोग करें।

---

21. What is a session and how does PHP manage sessions?

English: A session stores user-specific data across multiple requests (server-side storage). PHP sessions associate a session id (usually via cookie `PHPSESSID`) with server-side data (by default stored in files, but can be stored in DB/Redis). Flow: session_start() loads session data; modify `$_SESSION`; session_write_close() saves it. Sessions expire based on settings (`session.gc_maxlifetime`). Secure session management includes regenerating session IDs and storing minimal sensitive data.

Hindi: Session उपयोगकर्ता-विशिष्ट डेटा को कई अनुरोधों के बीच सर्वर पर स्टोर करने के लिए होता है। PHP session id (आम तौर पर `PHPSESSID` कुकी) के जरिए session डेटा से जोड़ता है। `session_start()` डेटा लोड करता है; `$_SESSION` को बदलें; `session_write_close()` सेव करता है। sessionsExpire `session.gc_maxlifetime` से नियंत्रित होते हैं। सुरक्षित प्रबंधन के लिए session id regenerate करना और संवेदनशील डेटा सीमित रखना चाहिए।

---

22. What are cookies and how do they differ from sessions?

English: Cookies store small pieces of data on the client (browser); sessions store data on the server and use a cookie (session id) to link client to server data. Cookies persist across sessions depending on expiry, are visible to user and can be modified, so not secure for sensitive data. Sessions are better for confidential data; cookies useful for preferences, remember-me flags, and tracking.

Hindi: Cookies ब्राउज़र पर छोटे डेटा पैकेट स्टोर करती हैं; sessions सर्वर पर डेटा स्टोर करते हैं और cookie (session id) के माध्यम से क्लाइंट को लिंक करते हैं। Cookies का डाटा क्लाइंट पर दिखाई दे सकता है और बदला जा सकता है, इसलिए संवेदनशील डेटा के लिए सुरक्षित नहीं। Sessions गोपनीय डेटा के लिए बेहतर हैं; cookies preference और remember-me जैसी सुविधाओं के लिए उपयोगी हैं।

---

23. What is session fixation?

English: Session fixation is an attack where an attacker sets or forces a victim to use a known session id, then takes over the session after the victim authenticates. Mitigations: regenerate session id on login (`session_regenerate_id(true)`), use secure cookies, use short lifetimes, and tie session to client properties (IP/user-agent) carefully.

Hindi: Session fixation एक हमला है जिसमें अटैकर उपयोगकर्ता को किसी ज्ञात session id का उपयोग करने के लिए मजबूर कर देता है और बाद में जब उपयोगकर्ता लॉगिन कर लेता है, अटैकर उस session को हथिया लेता है। रोकथाम: लॉगिन पर `session_regenerate_id(true)` चलाएँ, secure cookies का उपयोग करें, छोटे समय-सीमाएं रखें, और session को IP/यूजर-एजेंट से सावधानीपूर्वक बाँधें।

---

24. What is session hijacking?

English: Session hijacking is when an attacker steals a user's active session id (via XSS, sniffing unsecured traffic, or other means) and uses it to impersonate the user. Mitigations: use HTTPS, set `HttpOnly` and `Secure` flags on cookies, regenerate IDs, and implement additional checks (IP, user-agent) where appropriate.

Hindi: Session hijacking तब होता है जब अटैकर किसी उपयोगकर्ता का सक्रिय session id चुरा कर उनका नक़ल कर लेता है (XSS, unsecured traffic sniffing आदि से)। रोकथाम: HTTPS का उपयोग, cookie पर `HttpOnly` और `Secure` फ्लैग, session id regenerate और अतिरिक्त सत्यापन (IP, user-agent) लागू करें।

---

25. How does PHP handle session IDs?

English: PHP generates a session id (random string) and sends it to the client via a cookie or URL (less secure). The id maps to server-side session storage. Functions: `session_start()`, `session_id()`, `session_regenerate_id()`, `session_destroy()`. Configure entropy and cookie parameters in php.ini for stronger ids.

Hindi: PHP एक session id (random string) बनाता है और उसे कुकी या URL से क्लाइंट को भेजता है (URL कम सुरक्षित)। यह id सर्वर पर session डेटा से जुड़ी रहती है। कार्य: `session_start()`, `session_id()`, `session_regenerate_id()`, `session_destroy()`। मजबूत IDs के लिए php.ini में entropy और cookie पैरामीटर कॉन्फ़िगर करें।

---

26. What is OOP in PHP?

English: OOP (Object-Oriented Programming) structures code using classes and objects, encapsulating data and behavior. PHP supports OOP features: classes, objects, inheritance, interfaces, traits, visibility (public/protected/private), abstract classes, constructors/destructors, and magic methods. OOP helps organize complex code, improve reuse, and model real-world entities.

Hindi: OOP (ऑब्जेक्ट-ओरिएंटेड प्रोग्रामिंग) कोड को क्लासेस और ऑब्जेक्ट्स में व्यवस्थित करता है, जिससे डाटा और बिहेवियर कैप्सुलेट होते हैं। PHP में OOP के फीचर मौजूद हैं: क्लासेस, inheritance, interfaces, traits, visibility, abstract क्लासेस, constructors/destructors और magic methods। OOP जटिल कोड को व्यवस्थित और पुन:उपयोगी बनाता है।

---

27. What are classes and objects?

English: A class is a blueprint defining properties (data) and methods (functions). An object is an instance of a class with its own property values. Example: `class User { public $name; function greet(){} } $u = new User();`. Use classes to encapsulate logic and data together.

Hindi: क्लास एक ब्लूप्रिंट है जो प्रॉपर्टीज़ और मेथड्स को परिभाषित करती है। ऑब्जेक्ट क्लास का एक इंस्टेंस होता है जिसमें अपनी प्रॉपर्टीज़ के मान होते हैं। उदाहरण: `class User { public $name; function greet(){} } $u = new User();`। क्लासेस लॉजिक और डेटा को एक साथ कैप्सुलेट करने के लिए उपयोगी हैं।

---

28. What is inheritance in PHP?

English: Inheritance lets a class (child) derive from another (parent) using `extends`. Child inherits public/protected members and can override methods. It promotes code reuse and polymorphism. Multiple inheritance is not supported directly; use interfaces or traits to compose behavior.

Hindi: Inheritance से एक क्लास (child) दूसरी क्लास (parent) से `extends` के साथ गुण प्राप्त कर सकती है। Child public/protected मेम्बर्स विरासत में पाता है और मेथड ओवरराइड कर सकता है। इससे कोड पुनरुत्पादन होता है। PHP में डायरेक्ट मल्टीपल इनहेरिटेंस नहीं है; इसके लिए interfaces या traits का प्रयोग करें।

---

29. What is polymorphism?

English: Polymorphism allows objects of different classes to be treated via a common interface (methods share names but different implementations). Example: different classes implement `render()`; code can call `$obj->render()` without knowing the concrete class. Interfaces and inheritance enable polymorphism.

Hindi: Polymorphism वह क्षमता है जिससे विभिन्न क्लासों के ऑब्जेक्ट्स को एक आम इंटरफ़ेस के रूप में ट्रीट किया जा सकता है — यानी समान मेथड नाम पर अलग-अलग implementations हो सकती हैं। उदाहरण: अलग क्लासेस `render()` implement करती हैं; कोड सीधे `$obj->render()` कॉल कर सकता है। Interfaces और inheritance polymorphism को सक्षम करते हैं।

---

30. What is encapsulation?

English: Encapsulation bundles data and methods together and restricts direct access using visibility keywords (`public`, `protected`, `private`). It enforces controlled access through getters/setters and prevents external code from corrupting internal state. Encapsulation improves maintainability and API boundaries.

Hindi: Encapsulation डाटा और मेथड्स को एक साथ बाइंड करता है और visibility (`public`, `protected`, `private`) के ज़रिये सीधे एक्सेस को नियंत्रित करता है। getters/setters के माध्यम से नियंत्रित एक्सेस प्रदान करके आंतरिक स्थिति को अनियमित परिवर्तन से बचाता है। यह maintainability और API boundaries बेहतर बनाता है।

---

31. What are interfaces?

English: Interfaces declare method signatures without implementations. Classes `implement` interfaces and must define those methods. Interfaces define contracts, enabling polymorphism and decoupling. A class can implement multiple interfaces.

Hindi: Interfaces मेथड सिग्नेचर घोषित करते हैं पर implementation नहीं देते। क्लासें interfaces को `implement` करके उन तरीकों को परिभाषित करती हैं। Interfaces contract तय करते हैं और polymorphism तथा decoupling को सक्षम करते हैं। एक क्लास कई interfaces implement कर सकती है।

---

32. What are abstract classes?

English: Abstract classes can contain implemented methods and abstract methods (without body). They cannot be instantiated directly; child classes must implement abstract methods. Use abstract classes to provide common base functionality with enforced overrides.

Hindi: Abstract classes कुछ implemented methods और abstract methods (बिना बॉडी) दोनों रख सकती हैं। इन्हें सीधे instantiate नहीं किया जा सकता; child क्लासों को abstract methods को implement करना होता है। ये सामान्य बेस कार्यक्षमता और जरूरी overrides प्रदान करने के लिए उपयोगी हैं।

---

33. What is the difference between traits and interfaces?

English: Interfaces define method signatures (contracts) without implementation. Traits provide reusable method implementations that classes can include to share behavior (composition). Traits help solve code reuse when multiple inheritance is desired; interfaces ensure a class exposes specific methods.

Hindi: Interfaces केवल मेथड सिग्नेचर बताती हैं (contract)। Traits re-usable method implementations देती हैं जिन्हें क्लास `use` कर सकती है। Traits तब उपयोगी हैं जब मल्टीपल इनहेरिटेंस जैसा व्यवहार चाहिए; interfaces यह सुनिश्चित करती हैं कि क्लास कुछ विशेष मेथड दे।

---

34. What is method overloading in PHP?

English: PHP does not support classical compile-time method overloading by signature. However, you can emulate dynamic behavior using magic method `__call()` / `__callStatic()` to handle calls to undefined methods or using default/optional parameters and type checks.

Hindi: PHP पारंपरिक तरीके से method overloading signature के आधार पर सपोर्ट नहीं करता। फिर भी dynamic व्यवहार __call() / __callStatic() magic methods या default/optional पैरामीटर और type checks से इम्यूलेट किया जा सकता है।

---

35. What are magic methods like `__construct`, `__call`, `__get`, `__set`?

English: Magic methods are special methods with double-underscore names automatically invoked by PHP on certain events: `__construct` (object creation), `__destruct` (destruction), `__call` / `__callStatic` (invoking inaccessible methods), `__get` / `__set` (accessing inaccessible properties), `__toString`, `__invoke`, etc. Use them sparingly—overuse can reduce clarity.

Hindi: Magic methods वे विशेष मेथड हैं जिनके नाम द्वि-अण्डरस्कोर से शुरू होते हैं और PHP द्वारा विशेष घटनाओं पर ऑटोमैटिक कॉल होते हैं: `__construct`, `__destruct`, `__call`, `__get`, `__set`, `__toString`, `__invoke` आदि। इनका संयमित उपयोग करें क्योंकि ज्यादा उपयोग से कोड अस्पष्ट हो सकता है।

36. What is the difference between errors and exceptions?

English: Errors are low-level problems (parse errors, fatal errors) indicating code cannot continue; exceptions are objects representing runtime issues that can be caught and handled with try/catch. PHP 7 unified many fatal errors into `Error` exceptions which can be caught. Use exceptions for controllable error paths and custom error handling.

Hindi: Errors निचले स्तर की समस्याएँ होती हैं (parse/fatal errors) जो कोड को जारी नहीं रहने देतीं; exceptions runtime समस्याओं का प्रतिनिधित्व करते हैं जिन्हें try/catch से संभाला जा सकता है। PHP 7 से कई fatal errors `Error` exceptions में बदल दिए गए जिन्हें catch किया जा सकता है। नियंत्रित गलती-पथ के लिए exceptions का उपयोग करें।

---

37. What are the different error levels in PHP?

English: PHP defines error levels like `E_ERROR`, `E_WARNING`, `E_PARSE`, `E_NOTICE`, `E_DEPRECATED`, `E_STRICT`, `E_USER_*` variants. `error_reporting()` sets which levels are reported. In production, report and log errors but don't display them to users (`display_errors = Off`)—use logs for diagnostics.

Hindi: PHP में error levels जैसे `E_ERROR`, `E_WARNING`, `E_PARSE`, `E_NOTICE`, `E_DEPRECATED`, `E_STRICT`, और `E_USER_*` होते हैं। `error_reporting()` से रिपोर्टिंग स्तर नियंत्रित होते हैं। प्रोडक्शन में errors को दिखाने की बजाय लॉग करें (`display_errors = Off`) ताकि यूज़र अनुभव प्रभावित न हो।

---

38. What is exception handling? How does try–catch work?

English: Exception handling uses `try { ... } catch (Exception $e) { ... } finally { ... }`. Code inside `try` that throws an exception jumps to the first matching `catch`. `finally` always runs whether exception occurred or not. Use custom exception classes for finer-grained handling.

Hindi: Exception handling `try { ... } catch (Exception $e) { ... } finally { ... }` का उपयोग करता है। `try` ब्लॉक में उठाई गई exception सबसे नज़दीकी matching `catch` पर जाती है। `finally` हमेशा चलता है। बेहतर हैंडलिंग के लिए custom exception classes का प्रयोग करें।

---

39. What are custom exceptions?

English: Custom exceptions are user-defined classes extending `Exception` (or `\Throwable`). They allow categorizing errors (e.g., `DatabaseException`, `ValidationException`) and catching specific error types. They help keep error handling organized and expressive.

Hindi: Custom exceptions उपयोगकर्ता-परिभाषित क्लासें होती हैं जो `Exception` (या `\Throwable`) को एक्स्टेंड करती हैं। इससे आप त्रुटियों को वर्गीकृत कर सकते हैं (जैसे `DatabaseException`) और विशिष्ट प्रकारों को catch कर सकते हैं। यह error handling को व्यवस्थित बनाता है।

---

40. What is XSS and how can PHP prevent it?

English: Cross-Site Scripting (XSS) occurs when attackers inject malicious scripts into pages viewed by others. Prevent XSS by escaping output (`htmlspecialchars()` / `ENT_QUOTES`, proper context-aware escaping for HTML attributes, JS, URLs), using Content Security Policy (CSP), validating/sanitizing input, and avoiding dangerous dynamic HTML generation.

Hindi: XSS तब होता है जब अटैकर खतरनाक स्क्रिप्ट्स किसी पेज में इंजेक्ट कर देता है जो अन्य उपयोगकर्ता देखेंगे। रोकथाम: आउटपुट को escape करें (`htmlspecialchars()`), कंटेंट सिक्योरिटी पॉलिसी (CSP) का उपयोग करें, इनपुट validate/sanitize करें और डायनामिक HTML निर्माण से सावधानी बरतें।

---

41. What is CSRF and how to protect against it?

English: Cross-Site Request Forgery (CSRF) tricks an authenticated user's browser into performing unwanted actions on a site. Protect by using anti-CSRF tokens (synchronizer tokens stored in session and included in forms/requests), verifying `Origin` / `Referer` headers, using SameSite cookies, and requiring re-authentication for critical actions.

Hindi: CSRF में उपयोगकर्ता के ब्राउज़र से बिना उसकी मंशा के साइट पर एक्शन कराया जाता है। सुरक्षा: anti-CSRF tokens (session में स्टोर) फॉर्म/रिक्वेस्ट में शामिल करें, `Origin` / `Referer` वैरिफाई करें, SameSite cookies का उपयोग करें और संवेदनशील क्रियाओं के लिए दोबारा प्रमाणीकरण माँगें।

---

42. What is SQL Injection and how to prevent it in PHP?

English: SQL Injection happens when untrusted input is concatenated into SQL queries, allowing attackers to alter queries. Prevent by using prepared statements with parameter binding (PDO or MySQLi), input validation, least-privilege DB accounts, and avoiding dynamic query construction using raw input.

Hindi: SQL Injection तब होता है जब अनट्रस्टेड इनपुट SQL क्वेरीज़ में जोड़ दिया जाता है और अटैकर क्वेरी बदल सकता है। रोकथाम: prepared statements और parameter binding (PDO/MySQLi) का प्रयोग करें, इनपुट validate करें, DB अकाउंट को न्यूनतम अधिकार दें और raw input से डायनामिक क्वेरी निर्माण से बचें।

---

43. What is password hashing? Why use password_hash?

English: Password hashing converts passwords into fixed-length values using a one-way algorithm and salt. `password_hash()` (PHP) uses secure algorithms (bcrypt, Argon2 in newer PHP) and handles salt and cost automatically. Use `password_verify()` to check passwords. Never store plain-text or unsalted hashes.

Hindi: Password hashing पासवर्ड को एक-तरफ़ा एल्गोरिद्म और salt के साथ फिक्स्ड-लंबाई मान में बदल देता है। `password_hash()` सुरक्षित एल्गोरिद्म (bcrypt, Argon2) और salt/cost को ऑटोमैटिक संभालता है। पासवर्ड जांचने के लिए `password_verify()` का प्रयोग करें। प्लेन-टेक्स्ट या बिना salt के hashes कभी न रखें।

---

44. Why should you avoid using `md5()` or `sha1()` for passwords?

English: `md5()` and `sha1()` are fast cryptographic hashes without sufficient cost factor and are vulnerable to brute-force and rainbow-table attacks. For passwords use slow, adaptive hashing like bcrypt or Argon2 (via `password_hash()`), which include salt and tunable cost.

Hindi: `md5()` और `sha1()` तेज़ एल्गोरिद्म हैं और brute-force तथा rainbow-table हमलों के प्रति कमजोर होते हैं। पासवर्ड के लिए slow/adaptive hashing जैसे bcrypt या Argon2 (जो `password_hash()` देता है) का उपयोग करें—ये salt और cost को संभालते हैं।

---

45. What is opcode caching?

English: Opcode caching stores the compiled PHP bytecode (opcodes) in shared memory so subsequent requests skip the parse/compile steps and directly execute opcodes, drastically improving performance. OPcache is the common built-in extension. Opcode caches reduce CPU and improve response times.

Hindi: Opcode caching PHP के कंपाइलबाइटकोड (opcodes) को साझा मेमोरी में स्टोर करता है ताकि अगले अनुरोधों में parsing/compilation की आवश्यकता न रहे और सीधे opcodes execute हों—इससे प्रदर्शन में काफी सुधार होता है। OPcache एक सामान्य अंतर्निर्मित एक्सटेंशन है।

---

46. What is Composer and why is it used?

English: Composer is PHP's dependency manager. It declares and installs project dependencies, resolves versions, and provides autoloading via PSR-4/PSR-0. Composer helps manage third-party libraries, ensure reproducible installs using `composer.lock`, and promotes modular code.

Hindi: Composer PHP का dependency manager है। यह प्रोजेक्ट dependencies को घोषित और इंस्टॉल करता है, वर्ज़न को हल करता है और PSR-4/PSR-0 के माध्यम से autoloading देता है। Composer तीसरे पक्ष की लाइब्रेरी मैनेज करने और reproducible installs (`composer.lock`) सुनिश्चित करने में मदद करता है।

---

47. What is autoloading in PHP?

English: Autoloading automatically loads class files when a class is referenced, avoiding manual `require` calls. Composer provides PSR-compliant autoloaders, and PHP supports `spl_autoload_register()` for custom autoloaders. Autoloading improves code organization and performance by loading classes on demand.

Hindi: Autoloading क्लास फाइलों को स्वतः लोड करता है जब कोई क्लास रेफर की जाती है, जिससे मैन्युअल `require` की ज़रूरत कम हो जाती है। Composer PSR-compliant autoloaders प्रदान करता है और PHP `spl_autoload_register()` के माध्यम से कस्टम autoloaders का समर्थन करता है। Autoloading कोड संगठन और प्रदर्शन दोनों में सुधार लाती है।

---

48. What is the difference between require_once and autoload?

English: `require_once` explicitly includes a file and guarantees single inclusion. Autoloading defers loading until a class is referenced and uses conventions or mappings to find files automatically. Autoloading scales better in large applications and avoids many manual includes.

Hindi: `require_once` एक फ़ाइल को स्पष्ट रूप से शामिल करता है और एक बार शामिल होने की गारंटी देता है। Autoloading तब तक लोडिंग टालती है जब तक क्लास का संदर्भ न आए और फ़ाइलों को स्वतः ढूँढती है। बड़े एप्स में autoloading बेहतर स्केलेबल होता है और मैनुअल includes कम करता है।

---

49. What are namespaces in PHP?

English: Namespaces provide a way to group related classes, functions, and constants and avoid naming collisions (`namespace App\Controllers;`). They allow using the same class name in different contexts via fully-qualified names or `use` statements. Namespaces are essential for large codebases and third-party libraries.

Hindi: Namespaces संबंधित क्लासेस, फंक्शन्स और constants को समूहित करने का तरीका हैं और नाम-टकराव से बचाते हैं (`namespace App\Controllers;`)। ये एक ही नाम को विभिन्न संदर्भों में उपयोग करने की अनुमति देते हैं। बड़े कोडबेस और थर्ड-पार्टी लाइब्रेरीज़ के लिए namespaces आवश्यक हैं।

---

50. What is MVC architecture and how does PHP support it?

English: MVC (Model-View-Controller) separates application concerns: Model (data/business logic), View (presentation), Controller (request handling & coordination). PHP frameworks (Laravel, Symfony, CodeIgniter) implement MVC patterns, providing routing, controllers, templating, and models/ORMs. MVC improves testability, maintainability, and separation of concerns.

Hindi: MVC (Model-View-Controller) एप्लिकेशन के हिस्सों को अलग करता है: Model (डेटा/बिज़नेस लॉजिक), View (प्रेजेंटेशन), Controller (रिक्वेस्ट हैंडलिंग)। PHP frameworks (Laravel, Symfony आदि) MVC पैटर्न लागू करते हैं—routing, controllers, templating और ORM जैसी सुविधाएँ देते हैं। MVC टेस्टिंग, maintainability और separation of concerns में मदद करता है।

If you want this as a downloadable PDF, bilingual blog post, or separated into shorter flashcards and a printable cheat-sheet, tell me which format you prefer and I'll produce it.