

Welcome to your guide...

25 PHP INTERVIEW QUESTIONS & ANSWERS

Copyright © How2Become.com. All Rights Reserved.
For personal use only.

Disclaimer:

How2Become is not responsible for anyone failing any part of any selection process as a result of the information contained within this resource. How2Become and their authors cannot accept any responsibility for any errors or omissions within this resource, however caused. No responsibility for loss or damage occasioned by any person acting, or refraining from action, as a result of the material in this publication can be accepted by How2Become.

IMPORTANT: All resources, products, content, and training from How2Become is intended for educational use only, as an aid to help you prepare and come up with your own honest answers. How2Become is not acting in conjunction with, or associated with, any third-party organisation.

Get more guides, books and training courses at the website www.How2Become.com.

Copyright © How2Become.com. All Rights Reserved. For personal use only.

Q1. What is PHP and what are its main features?

Answer: PHP, which stands for Hypertext Preprocessor, is a popular server-side scripting language primarily used for web development. It is an open-source language, which means it's free to use and has a large community supporting it. PHP is known for its simplicity, making it relatively easy to learn and use even for beginners. It's a cross-platform language that can run on different operating systems like Windows, Linux, and MacOS. One of the key features of PHP is its ability to be embedded within HTML, which allows developers to integrate dynamic content within web pages seamlessly. PHP also offers robust database support, enabling it to connect with various database systems such as MySQL, PostgreSQL, and Oracle. Additionally, it has comprehensive error reporting, which helps identify and debug issues during development.

Q2. How does PHP handle form data?

Answer: PHP handles form data primarily through superglobal arrays like `$_GET` and `$_POST`. When a form is submitted, the data is sent to the server, and PHP retrieves it using these arrays. For instance, if a form is submitted via the POST method, PHP will use `$_POST` to access the form data. This array holds the form fields as key-value pairs, where the keys correspond to the name attributes of the form elements. If the form is submitted via the GET method, the data is appended to the URL, and PHP retrieves it using the `$_GET` array. It's crucial to validate and sanitise this data before using it to avoid security vulnerabilities such as SQL injection and cross-site scripting (XSS).

Q3. What is the difference between `include()` and `require()` in PHP?

Answer: Both `include()` and `require()` are used to include files in PHP scripts, but they differ in how they handle errors. When you use `include()` to insert the contents of one PHP file into another, the script will continue to execute even if the file cannot be found, though a warning will be issued. On the other hand, `require()` is stricter; if the specified file is not found, a fatal error occurs, and the script execution halts immediately. Therefore, `require()` is typically used when the included file is essential for the application. In contrast, `include()` might be

used when the file is optional, or you want the script to continue even if the file is missing.

Q4. Can you explain the concept of sessions in PHP?

Answer: Sessions in PHP are a way for a single user to store information across multiple pages. Unlike cookies, where data is stored on the client side, session data is stored on the server. When a session is started using `session_start()`, PHP generates a unique session ID sent to the user's browser as a cookie. This ID is then used to retrieve session data for that user in subsequent requests. Sessions are particularly useful for maintaining user state in web applications, such as keeping a user logged in across multiple pages. Session data is stored in a superglobal array called `$_SESSION`, and it remains available until the session is destroyed or times out.

Q5. What is the difference between `$_SESSION` and `$_COOKIE` in PHP?

Answer: `$_SESSION` and `$_COOKIE` are used to store data across multiple pages but function differently. `$_SESSION` stores data on the server, and only a unique session ID is stored in the user's browser as a cookie. This ID retrieves the session data from the server on subsequent requests. Because the data is stored on the server, it is more secure and can store more information. `$_COOKIE`, on the other hand, stores data directly in the user's browser. Each cookie is a small piece of data that the server sends to the browser and is included in subsequent requests to the server. Depending on their expiration time, cookies can persist even after the browser is closed. However, because cookies are stored on the client side, they are subject to size limitations and can be more vulnerable to security risks like tampering.

Q6. What is a PHP namespace and why is it important?

Answer: A namespace in PHP is a way to encapsulate classes, interfaces, functions, and constants to avoid name collisions in larger projects. As projects grow, it's common to encounter naming conflicts, especially when integrating third-party libraries. Placing code into namespaces creates a unique scope, reducing the chance of conflicts. Namespaces are defined using the `namespace` keyword, and you can access them using the `use` keyword. This

feature is particularly important in modern PHP development as it promotes better organisation and modularity of code, making it easier to manage and maintain large-scale applications.

Q7. What is the purpose of the final keyword in PHP?

Answer: The final keyword in PHP prevents a class or method from being extended or overridden. When you declare a class as final, it cannot be inherited by any other class. Similarly, if a method is declared final, it cannot be overridden in any subclass. This is useful when you want to enforce a certain implementation that should not be altered by derived classes, ensuring that the behaviour defined in the final class or method is preserved.

Q8. How does PHP interact with databases?

Answer: PHP interacts with databases using various extensions and libraries. The MySQL or PDO (PHP Data Objects) extension is the most common way to interact with databases. The MySQL extension provides functions specifically for interacting with MySQL databases. At the same time, PDO is a more flexible, object-oriented approach that supports multiple database types, including MySQL, PostgreSQL, SQLite, and others. PHP can execute SQL queries, fetch results, and handle errors through these extensions. PHP typically uses functions like `mysqli_connect()` to connect to a database or create a new PDO instance. Once connected, PHP can execute SQL statements, retrieve results, and manage database records directly from the script.

Q9. What are prepared statements in PHP, and why should they be used?

Answer: Prepared statements in PHP are a feature of database interaction that allows you to execute the same SQL query multiple times with different parameters while ensuring that the input is properly sanitised. They are particularly useful for preventing SQL injection attacks, as the query structure is defined separately from the data. When using prepared statements, placeholders are used in the SQL query, and the actual values are bound to these placeholders at execution time. This ensures that the data is treated as

such rather than as part of the SQL command, reducing the risk of executing malicious code.

Q10. Can you explain the difference between echo and print in PHP?

Answer: Echo and print are both language constructs in PHP used to output data to the browser, but they have slight differences. The Echo can take multiple arguments and does not return a value, making it marginally faster. On the other hand, print can only take one argument and always returns a value of 1, which allows it to be used in expressions. Although the difference in performance is negligible, echo is generally preferred when there is no need to use the return value. In contrast, print might be used when a return value is necessary.

Q11. What is an associative array in PHP?

Answer: An associative array in PHP is an array that uses named keys instead of numerical indexes. This allows you to store and retrieve data using a key-value pair, where the key is a string or integer you define. Associative arrays are particularly useful for storing logically paired data, such as the names and values of form inputs, configuration settings, or database records. You can create an associative array in PHP by assigning values to specific keys within the array syntax. For example, `$user = array("name" => "John", "email" => "john@example.com");` creates an associative array where name and email are keys.

Q12. How does error handling work in PHP?

Answer: Error handling in PHP can be managed through various methods, including built-in error reporting functions, custom error handlers, and exceptions. PHP has several error types, such as notices, warnings, and fatal errors, each indicating different levels of severity. By default, PHP displays errors directly on the web page, but it is common practice to log errors instead in production environments. The `error_reporting()` function allows you to set which types of errors are reported, while `set_error_handler()` lets you define custom error-handling functions. Additionally, PHP supports exceptions

through the try, catch, and finally blocks, which provide a structured way to handle errors in a controlled manner.

Q13. What are traits in PHP, and how are they used?

Answer: Traits in PHP are a mechanism for code reuse in single inheritance languages like PHP. Traits allow you to group methods that you can include in multiple classes. They are useful for scenarios where you want to share functionality across different classes without using inheritance. A trait is defined using the trait keyword and can be included in a class using the use keyword. This allows you to inject methods from the trait directly into the class as if they were defined in it. Traits are particularly helpful for reducing code duplication and improving code organisation.

Q14. What is the difference between == and === in PHP?

Answer: The == operator in PHP is used for comparison, but it only checks if the values are equal, performing type juggling if necessary. This means that different data types can be considered equal if PHP can convert one type into another to match the value. On the other hand, the === operator is a strict comparison operator that checks both the value and the data type. For example, 5 == "5" would return true because PHP converts the string to a number before comparing, but 5 === "5" would return false because one is an integer and the other is a string. Using === is generally preferred to avoid unexpected conversions and ensure the data types match.

Q15. How do you create a constant in PHP?

Answer: A constant in PHP is a value that cannot be changed after defining it. Constants are useful for defining values that should remain consistent throughout the execution of a script, such as configuration settings, API keys, or mathematical constants. In PHP, you can create a constant using the define() function, where the first argument is the name of the constant and the second argument is its value. Once defined, constants can be accessed anywhere in the script without a dollar sign (\$). Additionally, PHP provides a way to define constants within classes using the const keyword.

Q16. What is the purpose of the `isset()` function in PHP?

Answer: The `isset()` function in PHP checks if a variable is set and is not NULL. It returns true if the variable exists, has a value other than NULL, and false otherwise. This function commonly verifies that a variable has been initialised before performing operations. For example, `isset($_POST['username'])` would check if the username field was submitted in a form. It is particularly useful in scenarios where you want to avoid errors related to undefined variables or when you need to ensure that optional form fields or parameters are provided.

Q17. Explain how to handle file uploads in PHP.

Answer: Handling file uploads in PHP involves several steps. First, you must create an HTML form with the `enctype` attribute set to `multipart/form-data` and an input field of type `file`. PHP stores the uploaded file information in the `$_FILES` superglobal array when the form is submitted. This array contains details about the uploaded file, such as its name, type, size, and temporary location on the server. To process the uploaded file, you typically check for any errors, validate the file type and size, and then move the file from the temporary location to a desired directory using the `move_uploaded_file()` function. Implementing proper validation and security checks is important to prevent issues like file tampering or uploading of malicious files.

Q18. What are magic methods in PHP?

Answer: Magic methods in PHP are special methods that start with a double underscore (`__`) and are automatically called in certain situations. These methods allow you to define how objects should behave in specific contexts. For example, `__construct()` is a magic method that acts as a constructor, automatically invoked when a new object is created. Another common magic method is `__toString()`, which defines how an object should be represented as a string. Magic methods like `__get()` and `__set()` handle dynamic access to properties that are not explicitly defined. These methods are useful for controlling object behaviour and providing additional functionality without directly modifying the object's properties or methods.

Q19. What is the purpose of the header() function in PHP?

Answer: The header() function in PHP sends raw HTTP headers to the browser before any output is sent. This function is commonly used to perform tasks such as redirection, setting content types, and controlling caching behaviour. For example, header("Location: http://example.com") redirects the user to another page, while header("Content-Type: application/json") sets the content type to JSON. It's important to note that the header() function must be called before any output is sent to the browser, as HTTP headers must be sent before the actual content.

Q20. How do you connect to a MySQL database in PHP?

Answer: To connect to a MySQL database in PHP, you can use the mysqli extension or the PDO (PHP Data Objects) extension. The mysqli extension provides a procedural and object-oriented interface for interacting with MySQL databases, while PDO offers a more flexible, object-oriented approach that supports multiple database types. To connect using mysqli, you typically call mysqli_connect() with the database host, username, password, and database name as arguments. If the connection succeeds, you can execute SQL queries using functions like mysqli_query(). With PDO, you create a new instance of the PDO class, passing the DSN (Data Source Name), username, and password as parameters. PDO provides a consistent API for database interaction, making switching between different database systems easier.

Q21. What is the purpose of the mysqli_real_escape_string() function in PHP?

Answer: The mysqli_real_escape_string() function in PHP is used to escape special characters in a string before using it in an SQL query. This function helps prevent SQL injection attacks by ensuring that any potentially harmful characters in user input are neutralised before being included in a query. For example, characters like single quotes ('), which could be used to manipulate the query, are escaped so that they are treated as part of the data rather than as SQL syntax. This function is essential when inserting user input directly into

a database query, as it helps maintain the security and integrity of the database.

Q22. What is a PHP framework, and why would you use one?

Answer: A PHP framework is a platform or set of tools that provides a structured way to build web applications. Frameworks typically offer pre-built components and libraries that simplify common tasks such as routing, database interaction, form validation, and authentication. Popular PHP frameworks include Laravel, Symfony, and CodeIgniter. Using a framework can significantly speed up development by reducing the need to write repetitive code and providing a foundation following best practices. Frameworks also promote organised and maintainable code by enforcing a particular structure and design pattern, such as MVC (Model-View-Controller). Additionally, they often include built-in security features to help protect against common vulnerabilities like SQL injection and cross-site scripting (XSS).

Q23. What is an interface in PHP, and how is it different from an abstract class?

Answer: An interface in PHP defines a contract that any implementing class must follow. It contains method signatures without implementation, meaning that the methods have no body. Classes implementing an interface must define all the methods declared in the interface. An abstract class, on the other hand, can include both fully implemented methods and abstract methods, which any subclass must implement. The key difference is that a class can implement multiple interfaces, allowing for more flexible design. In contrast, a class can only extend one abstract class due to PHP's single inheritance model. Interfaces are useful when you want to define a consistent API that multiple classes can adhere to, even if they are otherwise unrelated.

Q24. How does PHP support object-oriented programming (OOP)?

Answer: PHP supports object-oriented programming (OOP) by providing features such as classes, objects, inheritance, encapsulation, and polymorphism. You can define classes with properties and methods and create instances of those classes as objects. PHP supports inheritance, allowing one class to inherit properties and methods from another, promoting code reuse. Encapsulation is achieved by controlling access to class members using visibility keywords like public, protected, and private. Polymorphism in PHP is supported through interfaces and abstract classes, allowing objects to be treated as instances of their parent classes or interfaces. OOP in PHP helps organise code into manageable, reusable, and scalable components, making developing and maintaining complex applications easier.

Q25. What are exceptions in PHP, and how are they handled?

Answer: Exceptions in PHP are a way to handle errors and exceptional conditions in a controlled manner. They provide an alternative to traditional error handling using the try, catch, and finally blocks. When an exception is thrown using the throw keyword, the normal flow of the program is interrupted, and PHP looks for a catch block that can handle the exception. The catch block catches the exception and allows the developer to define how the error should be handled, such as logging it or displaying a user-friendly message. The final block, if used, is executed after the try-and-catch blocks, regardless of whether an exception was thrown, and is typically used for clean-up operations. Exceptions improve error handling by making it easier to separate error-handling code from regular code and by providing more informative error messages.

How to Use These Interview Questions and Answers:

These interview questions and answers are intended to guide you in your preparation for your job interview. These questions have been picked by the How2Become team because we believe that they are the best representative of what you will face in your interview.

The sample answers in this resource are collated from years of experience and research in the recruitment sector. The answers confidently display the appropriate qualities and competencies that the interviewer expects from successful candidates.

Disclaimer:

How2Become is not responsible for anyone failing any part of any selection process as a result of the information contained within this resource. How2Become and their authors cannot accept any responsibility for any errors or omissions within this resource, however caused. No responsibility for loss or damage occasioned by any person acting, or refraining from action, as a result of the material in this publication can be accepted by How2Become.

IMPORTANT: All resources, products, content, and training from How2Become is intended for educational use only, as an aid to help you prepare and come up with your own honest answers. How2Become is not acting in conjunction with, or associated with, any third-party organisation.

Get more guides, books and training courses at the website www.How2Become.com.

Copyright © How2Become.com. All Rights Reserved. For personal use only.