Essential PHP Interview Questions & Answers

1. What is PHP and what does it stand for?

Answer: PHP originally stood for "Personal Home Page" but now stands for "PHP: Hypertext Preprocessor" (recursive acronym). It's a server-side scripting language designed for web development and can be embedded into HTML.

2. What are the differences between GET and POST methods?

Answer:

- **GET**: Data is sent via URL parameters, visible in browser history, limited data size (~2048 characters), cached by browsers, used for retrieving data
- **POST**: Data is sent in request body, not visible in URL, no size limitations, not cached, used for sending/modifying data

3. What are PHP Data Types?

Answer: PHP supports 8 primitive data types:

- **Scalar types**: boolean, integer, float, string
- **Compound types**: array, object, callable, iterable
- **Special types**: resource, NULL

4. What is the difference between include() and require()?

Answer:

- **include()**: Generates a warning if file not found, script continues execution
- **require()**: Generates a fatal error if file not found, script stops execution
- **include_once()** and **require_once()**: Same as above but include file only once

5. What are Sessions and Cookies?

Answer:

- **Sessions**: Server-side storage, more secure, data stored on server, expires when browser closes
- **Cookies**: Client-side storage, less secure, data stored on user's computer, can have expiration dates

```
php
```

```php
// Session example
session_start();
$_SESSION['username'] = 'john';

// Cookie example
setcookie('user', 'john', time() + 3600);
```

6. What is the difference between echo and print?

Answer:

- **echo**: No return value, can take multiple parameters, slightly faster
- **print**: Returns 1, takes only one parameter, slower than echo

```php
echo "Hello", " World"; // Valid
print "Hello World";    // Valid
// print "Hello", " World"; // Invalid
```

7. What are PHP Superglobals?

Answer: Built-in variables available in all scopes:

- `$_GET` - HTTP GET data
- `$_POST` - HTTP POST data
- `$_SESSION` - Session data
- `$_COOKIE` - Cookie values
- `$_SERVER` - Server information
- `$_FILES` - File upload information
- `$_ENV` - Environment variables
- `$GLOBALS` - Global variables

8. What is SQL Injection and how to prevent it?

Answer: SQL Injection is a code injection attack where malicious SQL code is inserted into application queries.

Prevention methods:

- Use prepared statements (PDO/MySQLi)
- Input validation and sanitization
- Use stored procedures
- Escape special characters

```php
// Vulnerable code
$query = "SELECT * FROM users WHERE id = " . $_GET['id'];

// Secure code with prepared statement
$stmt = $pdo->prepare("SELECT * FROM users WHERE id = ?");
$stmt->execute([$_GET['id']]);
```

9. What are Magic Methods in PHP?

Answer: Special methods that start with double underscores (__):

- `__construct()` - Constructor
- `__destruct()` - Destructor
- `__get()` - Get inaccessible properties
- `__set()` - Set inaccessible properties
- `__toString()` - Convert object to string
- `__clone()` - Clone object

```php
class Example {
    private $data = [];
    
    public function __get($name) {
        return $this->data[$name] ?? null;
    }
    
    public function __set($name, $value) {
        $this->data[$name] = $value;
    }
}
```

10. What is the difference between abstract classes and interfaces?

Answer:

- **Abstract Class**: Can have both abstract and concrete methods, supports properties, single inheritance
- **Interface**: Only method declarations (until PHP 8), no properties, multiple inheritance

```php
// Abstract class
abstract class Animal {
    protected $name;
    abstract public function makeSound();
    
    public function getName() {
        return $this->name;
    }
}

// Interface
interface Flyable {
    public function fly();
}
```

11. What is Composer in PHP?

Answer: Composer is a dependency management tool for PHP that allows you to declare libraries your project depends on and manages them for you.

```bash
# Install Composer package
composer require monolog/monolog

# Autoload classes
require 'vendor/autoload.php';
```

12. What are Namespaces in PHP?

Answer: Namespaces are a way of encapsulating items to avoid naming conflicts and organize code better.

```php
```

```php
namespace MyProject\Database;

class Connection {
    // Class implementation
}

// Usage
use MyProject\Database\Connection;
$conn = new Connection();
```

13. What is the difference between public, private, and protected?

Answer:

- **public**: Accessible from anywhere
- **private**: Accessible only within the same class
- **protected**: Accessible within the class and its subclasses

```php
class MyClass {
    public $publicVar = 'Public';
    private $privateVar = 'Private';
    protected $protectedVar = 'Protected';
}
```

14. What are PHP Design Patterns?

Answer: Common design patterns include:

- **Singleton**: Ensures only one instance of a class
- **Factory**: Creates objects without specifying exact classes
- **Observer**: Defines one-to-many dependency between objects
- **MVC**: Model-View-Controller architectural pattern

```php
```

```
// Singleton pattern
class Database {
    private static $instance = null;

    public static function getInstance() {
        if (self::$instance === null) {
            self::$instance = new self();
        }
        return self::$instance;
    }

    private function __construct() {}
}
```

15. What is the difference between array_merge() and array + array?

Answer:

- **array_merge()**: Re-indexes numeric keys, combines arrays
- **Array + operator**: Preserves keys, doesn't overwrite existing keys

```php
$arr1 = [0 => 'a', 1 => 'b'];
$arr2 = [0 => 'c', 2 => 'd'];

$merged = array_merge($arr1, $arr2);
// Result: [0 => 'a', 1 => 'b', 2 => 'c', 3 => 'd']

$added = $arr1 + $arr2;
// Result: [0 => 'a', 1 => 'b', 2 => 'd']
```

16. What is PDO in PHP?

Answer: PDO (PHP Data Objects) is a database access layer providing a uniform method of access to multiple databases. It supports prepared statements and multiple database drivers.

```php
```

```
try {
    $pdo = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
    $stmt = $pdo->prepare('SELECT * FROM users WHERE email = ?');
    $stmt->execute([$email]);
    $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}
```

17. What are PHP Traits?

Answer: Traits are a mechanism for code reuse in single inheritance languages like PHP. They allow you to include methods in multiple classes.

```php
trait Logger {
    public function log($message) {
        echo "Logging: " . $message;
    }
}

class User {
    use Logger;
    
    public function create() {
        $this->log('User created');
    }
}
```

18. What is autoloading in PHP?

Answer: Autoloading automatically loads PHP classes when they are needed, without requiring explicit include/require statements.

```php
```

```php
// PSR-4 autoloader example
spl_autoload_register(function ($className) {
    $file = str_replace('\\', DIRECTORY_SEPARATOR, $className) . '.php';
    if (file_exists($file)) {
        require $file;
    }
});
```

19. What are the differences between MySQL and MySQLi?

Answer:

- **MySQL**: Old extension, procedural interface, no prepared statements
- **MySQLi**: Improved extension, both procedural and OOP, supports prepared statements, enhanced debugging

php

```php
// MySQLi with prepared statements
$stmt = $mysqli->prepare("SELECT id, name FROM users WHERE age > ?");
$stmt->bind_param("i", $age);
$stmt->execute();
$result = $stmt->get_result();
```

20. What is CSRF and how to prevent it?

Answer: CSRF (Cross-Site Request Forgery) is an attack that forces users to execute unwanted actions on web applications.

Prevention:

- Use CSRF tokens
- Check HTTP Referer header
- Require re-authentication for sensitive operations

php

```php
// Generate CSRF token
session_start();
if (!isset($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}

// Verify CSRF token
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die('CSRF token mismatch');
}
```