Cheat Sheet for comprehensive PHP

Basic Syntax and Structure

- **PHP Tags**:

- `<?php ?>`: Standard PHP tags.
- `<? ?>`: Short tags (disabled by default, use `short_open_tag` in `php.ini`).
- `<?= $variable ?>`: Short echo tag.

- **Comments**:

- `// Single-line comment`
- `# Single-line comment`
- `/* Multi-line comment */`

- **Case Sensitivity**:

- Keywords (e.g., `if`, `else`, `while`) are case-insensitive.
- Function names and class names are case-insensitive.
- Variable names are case-sensitive.

Variables and Data Types

- **Variable Declaration**:

- `$variableName = value;`
- Example: `$name = "John";`

- **Data Types**:

  - **Scalar Types**:

- `string`: `"Hello"`
- `int`: `42`
- `float`: `3.14`
- `bool`: `true` or `false`

  - **Compound Types**:

- `array`: `array("apple", "banana")`
- `object`: `new stdClass()`
- `callable`: `function() { ... }`
- `iterable`: `foreach ($iterable as $item) { ... }`

  - **Special Types**:

- `null`: `null`
- `resource`: `fopen("file.txt", "r")`

- **Type Casting**:

- `(int) $var`, `(float) $var`, `(string) $var`, `(bool) $var`, `(array) $var`, `(object) $var`

Operators

- **Arithmetic Operators**:

- `+`, `-`, `*`, `/`, `%`, `**` (exponentiation)

- **Assignment Operators**:

- `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `.=`

- **Comparison Operators**:

- `==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, `<=`

- **Logical Operators**:

- `&&`, `||`, `!`

- **String Operators**:

- `.` (concatenation)

- **Array Operators**:

- `+` (union), `==`, `===`, `!=`, `!==`

Control Structures

- **Conditional Statements**:

- `if (condition) { ... }`
- `if (condition) { ... } else { ... }`
- `if (condition) { ... } elseif (condition) { ... } else { ... }`
- `switch (expression) { case value: ... break; default: ... }`

- **Loops**:

- `while (condition) { ... }`
- `do { ... } while (condition);`
- `for (init; condition; increment) { ... }`
- `foreach ($array as $value) { ... }`
- `foreach ($array as $key => $value) { ... }`

- **Jump Statements**:

- `break`: Exits the loop.
- `continue`: Skips the current iteration.
- `return`: Exits the function.
- `exit()`: Stops the script execution.

Functions

- **Function Declaration**:

- `function functionName($param1, $param2 = default) { ... }`
- Example: `function greet($name) { echo "Hello, $name!"; }`

- **Function Parameters**:

- **Default Parameters**: `function greet($name = "Guest") { ... }`

- **Variable-length Argument Lists**: `function sum(...$numbers) { ... }`

- **Return Values**:

- `return $value;`
- Example: `function add($a, $b) { return $a + $b; }`

- **Anonymous Functions**:

- `$greet = function($name) { echo "Hello, $name!"; };`

- **Arrow Functions**:

- `$greet = fn($name) => "Hello, $name!";`

Arrays

- **Array Declaration**:

- `$array = array("apple", "banana");`
- `$array = ["apple", "banana"];`

- **Accessing Array Elements**:

- `$array[0]`

- **Associative Arrays**:

- `$array = ["name" => "John", "age" => 30];`
- `$array["name"]`

- **Multidimensional Arrays**:

- `$array = [["a", "b"], ["c", "d"]];`
- `$array[0][1]`

- **Array Functions**:

- `count($array)`: Returns the number of elements.
- `array_push($array, $value)`: Adds an element to the end.
- `array_pop($array)`: Removes the last element.
- `array_merge($array1, $array2)`: Merges arrays.
- `array_keys($array)`: Returns all keys.
- `array_values($array)`: Returns all values.
- `in_array($value, $array)`: Checks if a value exists.
- `sort($array)`: Sorts the array.
- `rsort($array)`: Sorts the array in reverse.
- `ksort($array)`: Sorts by keys.
- `krsort($array)`: Sorts by keys in reverse.

Strings

- **String Concatenation**:

- `$fullName = $firstName . " " . $lastName;`

- **String Functions**:

- `strlen($string)`: Returns the length.
- `str_replace($search, $replace, $string)`: Replaces occurrences.
- `strpos($string, $search)`: Finds the position of the first occurrence.
- `substr($string, $start, $length)`: Returns a part of the string.
- `trim($string)`: Removes whitespace.
- `explode($delimiter, $string)`: Splits a string into an array.
- `implode($glue, $array)`: Joins array elements into a string.

File Handling

- **Opening a File**:

- `$file = fopen("file.txt", "r");`

- **Reading from a File**:

- `fread($file, filesize("file.txt"));`
- `fgets($file);`

- **Writing to a File**:

- `fwrite($file, "Content");`

- **Closing a File**:

- `fclose($file);`

- **File Functions**:

- `file_exists("file.txt")`: Checks if a file exists.
- `is_file("file.txt")`: Checks if it's a file.
- `is_dir("directory")`: Checks if it's a directory.
- `mkdir("directory")`: Creates a directory.
- `unlink("file.txt")`: Deletes a file.

Error Handling

- **Error Reporting**:

- `error_reporting(E_ALL);`
- `ini_set('display_errors', 1);`

- **Custom Error Handling**:

- `set_error_handler("customError");`
- Example:

```
function customError($errno, $errstr) {
    echo "Error: [$errno] $errstr";
}
```

- **Exceptions**:

- `try { ... } catch (Exception $e) { ... } finally { ... }`
- Example:

```
try {
    throw new Exception("Error occurred");
} catch (Exception $e) {
    echo $e->getMessage();
}
```

Object-Oriented Programming (OOP)

- **Class Declaration**:

- ```php

class MyClass {

```
    public $property;

    public function method() { ... }

}
```

```
- **Object Instantiation**:
  - `$object = new MyClass();`

- **Accessing Properties and Methods**:
  - `$object->property`
  - `$object->method()`

- **Constructors and Destructors**:
  - ```php
    class MyClass {
        public function __construct() { ... }
        public function __destruct() { ... }
    }
```

- **Inheritance**:

- ```php

```
class ChildClass extends ParentClass {

    public function childMethod() { ... }

}
```

```
- **Interfaces and Abstract Classes**:
  - ```php
    interface MyInterface {
        public function method();
    }

    abstract class AbstractClass {
        abstract public function method();
    }
```

Namespaces and Autoloading

- **Namespace Declaration**:

- ```php

namespace MyNamespace;

class MyClass { ... }

```
- **Using Namespaces**:
  - `use MyNamespace\MyClass;`

- **Autoloading**:
  - ```php
    spl_autoload_register(function ($class) {
        include $class . '.php';
    });
```

Web Development

- **Handling Forms**:

- ```php

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

$name = $_POST['name'];

}

```
- **Cookies**:
  - `setcookie("name", "value", time() + 3600);`
  - `$_COOKIE["name"]`

- **Sessions**:
  - `session_start();`
  - `$_SESSION["name"] = "value";`
  - `unset($_SESSION["name"]);`
  - `session_destroy();`

- **HTTP Headers**:
  - `header("Location: http://example.com");`
  - `header("Content-Type: application/json");`

### Database Interaction

- **Connecting to a Database**:
  - ```php
    $conn = new mysqli("localhost", "user", "password", "database");
```

- **Executing Queries**:

- ```php

  $result = $conn->query("SELECT * FROM users");

  while ($row = $result->fetch_assoc()) {

    echo $row['name'];

  }

```
- **Prepared Statements**:
  - ```php
    $stmt = $conn->prepare("INSERT INTO users (name, age) VALUES (?, ?)");
    $stmt->bind_param("si", $name, $age);
    $stmt->execute();
```

Miscellaneous

- **Date and Time**:

- `date("Y-m-d H:i:s")`
- `time()`
- `strtotime("next Monday")`

- **Regular Expressions**:

- `preg_match("/pattern/", $string)`
- `preg_replace("/pattern/", "replacement", $string)`

- **JSON Handling**:

- `json_encode($array)`
- `json_decode($jsonString)`

- **Command Line Execution**:

- `exec("command")`
- `shell_exec("command")`
- `system("command")`

Best Practices

- **Code Formatting**:

- Use consistent indentation (4 spaces).

- Follow PSR standards (PHP-FIG).

- **Security**:

- Use prepared statements for database queries.
- Validate and sanitize user inputs.
- Use `htmlspecialchars()` to prevent XSS.

- **Performance**:

- Minimize database queries.
- Use caching mechanisms.
- Optimize loops and recursive functions.

- **Documentation**:

- Use PHPDoc for function and class documentation.
- Example:

```
/**
 * Sum two numbers.
 *
 * @param int $a
 * @param int $b
 * @return int
 */
function sum($a, $b) {
    return $a + $b;
}
```

This cheat sheet provides a comprehensive overview of essential PHP features, syntax, and best practices. Use it as a quick reference guide for your PHP development needs.

By Ahmed Baheeg Khorshid

ver 1.0