PHP Cheatsheet

A comprehensive cheat sheet covering essential PHP syntax, functions, and best practices for web development.

Core Syntax & Data Types

Basic Syntax

Opening tag	`<?php`
Closing tag	`?>`
Statement terminator	`;`
Comments (single-line)	`//` or `#`
Comments (multi-line)	`/* ... */`
Echoing output	`echo 'Hello, world!';` or `print 'Hello, world!';`

Data Types

Scalar Types:

- `int`: Integer
- `float`: Floating-point number
- `string`: Sequence of characters
- `bool`: Boolean (true or false)

Compound Types:

- `array`: Ordered map
- `object`: Instance of a class
- `callable`: can be used as parameter for functions such as `call_user_func()`

Special Types:

- `resource`: A reference to an external resource
- `null`: Represents a variable with no value

Variable Declaration

Declaration	`$variable_name = value;`
Example	`$name = "John Doe";` `$age = 30;`

Operators & Control Structures

Operators

Arithmetic	`+, -, *, /, %`
Assignment	`=, +=, -=, *=, /=, %=`
Comparison	`==, ===, !=, !==, >, <, >=, <=`
Increment/Decrement	`++, --`
Logical	`&& (and), || (or), ! (not)`
String	`. (concatenation), .= (concatenation assignment)`

Control Structures

Conditional Statements:

```
if (condition) {
  // code to be executed if condition is true
} elseif (condition) {
  // code to be executed if first condition is false and this condition is
true
} else {
  // code to be executed if all conditions are false
}
```

Switch Statement:

```
switch (expression) {
  case value1:
    // code to be executed if expression = value1
    break;
  case value2:
    // code to be executed if expression = value2
    break;
  default:
    // code to be executed if expression is different from both value1 and
value2
}
```

Loops:

```
for ($i = 0; $i < 10; $i++) {
  // code to be executed
}

while (condition) {
  // code to be executed
}

do {
  // code to be executed
} while (condition);

foreach ($array as $value) {
  // code to be executed
}
```

Functions & Arrays

Functions

Definition	`function functionName($arg1, $arg2) {`
	`  // code to be executed`
	`  return $returnValue;`
	`}`

Calling a function	`functionName(value1, value2);`

Example with default argument	`function greet($name = "Guest") {`
	`  echo "Hello, $name!";`
	`}`

Arrays

Indexed Arrays:

```
$colors = array("Red", "Green", "Blue");
echo $colors[0]; // Output: Red
```

Associative Arrays:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
```

Multidimensional Arrays:

```
$cars = array(
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".";
```

Array Functions

`count()`	Returns the number of elements in an array.
`array_push()`	Adds one or more elements to the end of an array.
`array_pop()`	Removes the last element from an array.
`array_shift()`	Removes the first element from an array.
`array_unshift()`	Adds one or more elements to the beginning of an array.
`array_merge()`	Merges one or more arrays into one array.
`in_array()`	Checks if a value exists in an array.

Classes & Objects

Class Definition

```
class ClassName {
  // Properties
  public $property1;
  private $property2;
  protected $property3;

  // Methods
  public function method1() {
    // Code
  }

  private function method2() {
    // Code
  }

  protected function method3() {
    // Code
  }
}
```

Visibility:

- `public`: Accessible from anywhere.
- `private`: Accessible only within the class.
- `protected`: Accessible within the class and by inheriting classes.

Object Instantiation

Creating an object	`$object = new ClassName();`

Accessing properties and methods	`$object->property1 = "Value";`
	`echo $object->method1();`

Constructors & Destructors

Constructor:

```
class MyClass {
  public function __construct() {
    // Code to be executed when an object is created
  }
}
```

Destructor:

```
class MyClass {
  public function __destruct() {
    // Code to be executed when an object is destroyed
  }
}
```

Inheritance

Extending a class	`class ChildClass extends ParentClass {`
	`  // Code`
	`}`

Overriding methods	A child class can override methods of the parent class.