

# PHP

## Notes for Professionals

### Chapter 18: Working with Dates and Times

#### Section 18.1: Getting the difference between two dates

The most flexible way is to use the `DateInterval` class.

```
An example:
<?php
// Create a date time object, which has the value of = two years ago
$startDate = new DateTime('2014-07-18 20:05:00');
// Create a date time object, which has the value of = now
$now = new DateTime('2016-07-21 00:05:00');

// Calculate the diff
$diff = $now->diff($startDate);

// $diff->y contains the difference in years between the two dates
$yearsDiff = $diff->y;
// $diff->m contains the difference in months between the two dates
$monthsDiff = $diff->m;
// $diff->d contains the difference in days between the two dates
$daysDiff = $diff->d;
// $diff->h contains the difference in hours between the two dates
$hoursDiff = $diff->h;
// $diff->i contains the difference in minutes between the two dates
$minutesDiff = $diff->i;
// $diff->s contains the difference in seconds between the two dates
$secondsDiff = $diff->s;

// Total days diff, that is the number of days between the two dates
$totalDaysDiff = $diff->days;

// Dump the diff altogether just to get some details :)
var_dump($diff);

// Note, comparing two dates is much easier, just use the Comparison operators
var_dump($startDate < $now);
```

#### Section 18.2: Convert a date into another format

The simplest way to convert one date format into another is to use `strtotime()` with `strtotime()` to convert the date into a Unix Timestamp. The Unix Timestamp can then be passed to `date()` to convert it into a new format.

```
$timestamp = strtotime('2008-07-01 12:35:17.00');
echo date('Y-m-d H:i:s', $timestamp);
```

### Chapter 24: String formatting

#### Section 24.1: String Interpolation

You can also use interpolation to interpolate (insert) a variable within a string. Interpolation works in double quoted strings and the heredoc syntax only.

```
$name = 'Jock';

// $name will be replaced with 'Jock'
echo "opello $name, Nice to see you.<?php>";
// "opello Jock, Nice to see you.<?php>"

// Single Quotes: outputs $name as the raw text (without interpreting \<?php>)
echo 'Hello $name, Nice to see you.'; // Careful with this notation
// "Hello $name, Nice to see you."
```

The complex (curly) syntax format provides another option which requires that you wrap your variable within curly braces {}. This can be useful when embedding variables within textual content and helping to prevent possible ambiguity between textual content and variables.

```
$name = 'Jock';

// Example using the curly brace syntax for the variable $name
echo "opello {name} nice to see you.<?php>";
// "opello Jock nice to see you.<?php>"

// This line will throw an error (as $name is not defined)
echo "opello {name} nice to see you.<?php>";
// Notice: Undefined variable: name
```

The {} syntax only interpolates variables starting with a \$ into a string. The {} syntax does not evaluate arbitrary PHP expressions.

```
// Example trying to interpolate a PHP expression
echo "1 + 2 = {1 + 2}";
// "1 + 2 = {1 + 2}"

// Example using a constant
define('HELLO_WORLD', 'Hello World!');
echo "My constant is {HELLO_WORLD}";
// "My constant is Hello World!"
```

However, the {} syntax does evaluate any array access, property access and function/method calls on variables and array elements or properties:

```
// Example accessing a value from an array -- multidimensional access is allowed
$compositions = ['Bach' => 'Johann Sebastian Bach', 'Mozart' => 'Wolfgang Amadeus Mozart'];
echo "The best composer is: { $compositions[0]['name'] }";
```

PHP Notes for Professionals

### Chapter 64: Sending Email

Parameter	Details
string \$to	The recipient email address
string \$subject	The subject line
string \$message	The body of the email
string \$additional_headers	Optional: headers to add to the email command line
string \$additional_parameters	Optional: arguments to pass to the configured mail send application in the command line

#### Section 64.1: Sending Email - The basics, more details, and a full example

A typical email has three main components:

1. A recipient (represented as an email address)
2. A subject
3. A message body

Sending mail in PHP can be as simple as calling the built-in function `mail()`. `mail()` takes up to five parameters but the first three are all that is required to send an email (although the four parameters is commonly used as will be demonstrated below). The first three parameters are:

1. The recipient's email address (string)
2. The email's subject (string)
3. The body of the email (string) (e.g. the content of the email)

A minimal example would resemble the following code:

```
mail('recipient@example.com', 'Email Subject', 'This is the email message body');
```

The simple example above works well in limited circumstances such as hardcoding an email alert for an internal system. However, it is common to place the data passed as the parameters for `mail()` in variables to make the code cleaner and easier to manage (for example, dynamically building an email from a form submission).

Additionally, `mail()` accepts a fourth parameter which allows you to have additional mail headers sent with your email. These headers can allow you to set:

- the From name and email address the user will see
- the Reply-To email address the user's response will be sent to
- additional non-standard headers like X-Mailer when can tell the recipient this email was sent via PHP

The optional fifth parameter can be used to pass additional flags as command line options to the program configured to be used when sending mail, as defined by the `sendmail_path` configuration setting. For example, this

PHP Notes for Professionals

400+ pages  
of professional hints and tricks

# Contents

<b>About</b>	1
<b>Chapter 1: Getting started with PHP</b>	2
Section 1.1: HTML output from web server	2
Section 1.2: Hello, World!	3
Section 1.3: Non-HTML output from web server	3
Section 1.4: PHP built-in server	5
Section 1.5: PHP CLI	5
Section 1.6: Instruction Separation	6
Section 1.7: PHP Tags	7
<b>Chapter 2: Variables</b>	9
Section 2.1: Accessing A Variable Dynamically By Name (Variable variables)	9
Section 2.2: Data Types	10
Section 2.3: Global variable best practices	13
Section 2.4: Default values of uninitialized variables	14
Section 2.5: Variable Value Truthiness and Identical Operator	15
<b>Chapter 3: Variable Scope</b>	18
Section 3.1: Superglobal variables	18
Section 3.2: Static properties and variables	18
Section 3.3: User-defined global variables	19
<b>Chapter 4: Superglobal Variables PHP</b>	21
Section 4.1: Suberglobals explained	21
Section 4.2: PHP5 SuperGlobals	28
<b>Chapter 5: Outputting the Value of a Variable</b>	32
Section 5.1: echo and print	32
Section 5.2: Outputting a structured view of arrays and objects	33
Section 5.3: String concatenation with echo	35
Section 5.4: printf vs sprintf	36
Section 5.5: Outputting large integers	36
Section 5.6: Output a Multidimensional Array with index and value and print into the table	37
<b>Chapter 6: Constants</b>	39
Section 6.1: Defining constants	39
Section 6.2: Class Constants	40
Section 6.3: Checking if constant is defined	40
Section 6.4: Using constants	42
Section 6.5: Constant arrays	42
<b>Chapter 7: Magic Constants</b>	43
Section 7.1: Difference between <code>FUNCTION</code> and <code>METHOD</code>	43
Section 7.2: Difference between <code>CLASS</code> , <code>get_class()</code> and <code>get_called_class()</code>	43
Section 7.3: File & Directory Constants	44
<b>Chapter 8: Comments</b>	45
Section 8.1: Single Line Comments	45
Section 8.2: Multi Line Comments	45
<b>Chapter 9: Types</b>	46
Section 9.1: Type Comparison	46
Section 9.2: Boolean	46
Section 9.3: Float	47

Section 9.4: Strings .....	48
Section 9.5: Callable .....	50
Section 9.6: Resources .....	50
Section 9.7: Type Casting .....	51
Section 9.8: Type Juggling .....	51
Section 9.9: Null .....	52
Section 9.10: Integers .....	52
<b>Chapter 10: Operators .....</b>	<b>54</b>
Section 10.1: Null Coalescing Operator (??) .....	54
Section 10.2: Spaceship Operator (<=>) .....	55
Section 10.3: Execution Operator (`) .....	55
Section 10.4: Incrementing (++) and Decrementing Operators (--)	55
Section 10.5: Ternary Operator (?) .....	56
Section 10.6: Logical Operators (&&/AND and   /OR) .....	57
Section 10.7: String Operators (. and .=) .....	57
Section 10.8: Object and Class Operators .....	57
Section 10.9: Combined Assignment (+= etc) .....	59
Section 10.10: Altering operator precedence (with parentheses)	59
Section 10.11: Basic Assignment (=) .....	60
Section 10.12: Association .....	60
Section 10.13: Comparison Operators .....	60
Section 10.14: Bitwise Operators .....	62
Section 10.15: instanceof (type operator) .....	64
<b>Chapter 11: References .....</b>	<b>67</b>
Section 11.1: Assign by Reference .....	67
Section 11.2: Return by Reference .....	67
Section 11.3: Pass by Reference .....	68
<b>Chapter 12: Arrays .....</b>	<b>71</b>
Section 12.1: Initializing an Array .....	71
Section 12.2: Check if key exists .....	73
Section 12.3: Validating the array type .....	74
Section 12.4: Creating an array of variables .....	74
Section 12.5: Checking if a value exists in array .....	74
Section 12.6: ArrayAccess and Iterator Interfaces .....	75
<b>Chapter 13: Array iteration .....</b>	<b>79</b>
Section 13.1: Iterating multiple arrays together .....	79
Section 13.2: Using an incremental index .....	80
Section 13.3: Using internal array pointers .....	80
Section 13.4: Using foreach .....	81
Section 13.5: Using ArrayObject Iterator .....	83
<b>Chapter 14: Executing Upon an Array .....</b>	<b>84</b>
Section 14.1: Applying a function to each element of an array	84
Section 14.2: Split array into chunks .....	85
Section 14.3: Imploding an array into string .....	86
Section 14.4: "Destructuring" arrays using list() .....	86
Section 14.5: array_reduce .....	86
Section 14.6: Push a Value on an Array .....	87
<b>Chapter 15: Manipulating an Array .....</b>	<b>89</b>
Section 15.1: Filtering an array .....	89
Section 15.2: Removing elements from an array .....	90

Section 15.3: Sorting an Array .....	91
Section 15.4: Whitelist only some array keys .....	96
Section 15.5: Adding element to start of array .....	96
Section 15.6: Exchange values with keys .....	97
Section 15.7: Merge two arrays into one array .....	97
<b>Chapter 16: Processing Multiple Arrays Together .....</b>	<b>99</b>
Section 16.1: Array intersection .....	99
Section 16.2: Merge or concatenate arrays .....	99
Section 16.3: Changing a multidimensional array to associative array .....	100
Section 16.4: Combining two arrays (keys from one, values from another) .....	100
<b>Chapter 17: Datetime Class .....</b>	<b>102</b>
Section 17.1: Create Immutable version of DateTime from Mutable prior PHP 5.6 .....	102
Section 17.2: Add or Subtract Date Intervals .....	102
Section 17.3: getTimestamp .....	102
Section 17.4: setDate .....	103
Section 17.5: Create DateTime from custom format .....	103
Section 17.6: Printing DateTimes .....	103
<b>Chapter 18: Working with Dates and Time .....</b>	<b>105</b>
Section 18.1: Getting the difference between two dates / times .....	105
Section 18.2: Convert a date into another format .....	105
Section 18.3: Parse English date descriptions into a Date format .....	107
Section 18.4: Using Predefined Constants for Date Format .....	107
<b>Chapter 19: Control Structures .....</b>	<b>109</b>
Section 19.1: if else .....	109
Section 19.2: Alternative syntax for control structures .....	109
Section 19.3: while .....	109
Section 19.4: do-while .....	110
Section 19.5: goto .....	110
Section 19.6: declare .....	110
Section 19.7: include & require .....	111
Section 19.8: return .....	112
Section 19.9: for .....	112
Section 19.10: foreach .....	113
Section 19.11: if elseif else .....	113
Section 19.12: if .....	114
Section 19.13: switch .....	114
<b>Chapter 20: Loops .....</b>	<b>116</b>
Section 20.1: continue .....	116
Section 20.2: break .....	117
Section 20.3: foreach .....	118
Section 20.4: do...while .....	118
Section 20.5: for .....	119
Section 20.6: while .....	120
<b>Chapter 21: Functions .....</b>	<b>121</b>
Section 21.1: Variable-length argument lists .....	121
Section 21.2: Optional Parameters .....	122
Section 21.3: Passing Arguments by Reference .....	123
Section 21.4: Basic Function Usage .....	124
Section 21.5: Function Scope .....	124

<b>Chapter 22: Functional Programming</b>	125
Section 22.1: Closures	125
Section 22.2: Assignment to variables	126
Section 22.3: Objects as a function	126
Section 22.4: Using outside variables	127
Section 22.5: Anonymous function	127
Section 22.6: Pure functions	128
Section 22.7: Common functional methods in PHP	128
Section 22.8: Using built-in functions as callbacks	129
Section 22.9: Scope	129
Section 22.10: Passing a callback function as a parameter	129
<b>Chapter 23: Alternative Syntax for Control Structures</b>	131
Section 23.1: Alternative if/else statement	131
Section 23.2: Alternative for statement	131
Section 23.3: Alternative while statement	131
Section 23.4: Alternative foreach statement	131
Section 23.5: Alternative switch statement	132
<b>Chapter 24: String formatting</b>	133
Section 24.1: String interpolation	133
Section 24.2: Extracting/replacing substrings	134
<b>Chapter 25: String Parsing</b>	136
Section 25.1: Splitting a string by separators	136
Section 25.2: Substring	136
Section 25.3: Searching a substring with strpos	138
Section 25.4: Parsing string using regular expressions	139
<b>Chapter 26: Classes and Objects</b>	140
Section 26.1: Class Constants	140
Section 26.2: Abstract Classes	142
Section 26.3: Late static binding	144
Section 26.4: Namespacing and Autoloading	145
Section 26.5: Method and Property Visibility	147
Section 26.6: Interfaces	149
Section 26.7: Final Keyword	152
Section 26.8: Autoloading	153
Section 26.9: Calling a parent constructor when instantiating a child	154
Section 26.10: Dynamic Binding	155
Section 26.11: \$this, self and static plus the singleton	156
Section 26.12: Defining a Basic Class	159
Section 26.13: Anonymous Classes	160
<b>Chapter 27: Namespaces</b>	162
Section 27.1: Declaring namespaces	162
Section 27.2: Referencing a class or function in a namespace	162
Section 27.3: Declaring sub-namespaces	163
Section 27.4: What are Namespaces?	164
<b>Chapter 28: Sessions</b>	165
Section 28.1: session_start() Options	165
Section 28.2: Session Locking	165
Section 28.3: Manipulating session data	166
Section 28.4: Destroy an entire session	166

[Click here to download full PDF material](#)