8. PHP Tutorial I

웹프로그래밍

2020년 2학기

충남대학교 컴퓨터공학과

PHP Tutorial

- PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages
- PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.
- PHP 7 is the latest stable release.
- *PHP Exercises*

PHP Introduction

- **What is PHP?**
  - PHP is an acronym for “PHP:Hypertext Preprocessor”
  - PHP is a widely-used, open source scripting language
  - PHP scripts are executed on the server
  - PHP is free to download and use

- **What is a PHP File?**
  - PHP files can contain text, HTML, CSS, JavaScript, and PHP code
  - PHP code are executed on the server, and the result is returned to the browser as plain HTML
  - PHP files have extension “**.php**”

PHP Introduction (cont'd)

- What Can PHP Do?
  - Can generate dynamic page content
  - Can create, open, read, write, delete, and close files on the server
  - Can collect form data
  - Can send and receive cookies
  - Can add, delete, modify data in your database
  - Can be used to control user-access
  - Can encrypt data

PHP Introduction (cont'd)

- **Why PHP?**
  - Runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
  - Is compatible with almost all servers used today (Apache, IIS, etc.)
  - Supports a wide range of databases
  - Is free. Download it from the official PHP resource: www.php.net
  - Is easy to learn and runs efficiently on the server side

- **What's new in PHP7**
  - Is much faster than the previous popular stable release (PHP 5.6)
  - Has improved Error Handling
  - Supports stricter Type Declarations for function arguments
  - Supports new operators (like the spaceship operator : **< = >**)

PHP Installation

- **To start using PHP, you can**
  - Find a web host with PHP and MySQL support
  - Install a web server on your own PC, and then install PHP and MySQL

- **Use a Web Host With PHP Support**
  - If your server has activated support for PHP you do not need to do anything.
  - Just create some **.php** files, place them in your web directory, and the server will automatically parse them for you

- **Set Up PHP on Your Own PC**
  - If your server does not support PHP, you must
    - Install a web server
    - Install PHP (http://php.net/manual/en/install.php)
    - Install a database, such as MySQL

PHP Syntax

- **Basic PHP Syntax** *Try it!*
  - A PHP script starts with **<?php** and ends with **?>**
  - The default file extension for PHP files is "**.php**".
  - A PHP file normally contains HTML tags, and some PHP scripting code
  - Note : PHP statements end with a semicolon(**;**)

```
<?php
// PHP code goes here
?>
```

- **PHP Case Sensitivity**
  - No keywords(e.g **if**, **else**, **while**, **echo**, etc), classes, functions, and user-defined functions are case-sensitive. *Try it!*
  - However; all variable names are case-sensitive *Try it!*

- *PHP Exercises*

PHP Comments

- **Comments in PHP** *Try it!* *Try it!* *Try it!*
  - Comments can be used to :
    - Let others understand your code
    - Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

PHP Variables

- **Creating (Declaring) PHP Variables** *Try it!*
  - A variable starts with the **$** sign, followed by the name of the variable.
  - **Note** : When you assign a text value to a variable, put quotes around the value.
  - **Note**: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

- **PHP Variables**
  - Rules for PHP variables ;
    - A variable starts with the **$** sign, followed by the name of the variable
    - A variable name must start with a letter or the underscore character
    - A variable name cannot start with a number
    - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
    - Variable names are case-sensitive (**$age** and **$AGE** are two different variables)

PHP Variables (cont'd)

- **Output Variables** *Try it!* *Try it!* *Try it!*
  - The PHP **echo** statement is often used to output data to the screen.

- **PHP is a Loosely Typed Language**
  - PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.
  - In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch
  - You will learn more about the **strict**, and **non-strict** requirements, and the data type declarations in the PHP Functions chapter.

PHP Variables (cont'd)

- **PHP Variables Scope**
  - The scope of a variable is the part of the script where the variable can be referenced/used
  - PHP has three different variable scopes
    - local, global, static

- **Global and Local Scope**
  - A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function *Try it!*
  - A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function *Try it!*

PHP Variables (cont'd)

- **PHP The global Keyword** *Try it!*
  - The **global** keyword is used to access a global variable from within a function
  - PHP also stores all global variables in an array called **$GlOBALS[*index*].** The ***index*** holds the name of the variable. The array is also accessible from within functions and can be used to update global variables directly. *Try it!*

```
<?php
$x = 5;
$y = 10;

function myTest() {
  global $x, $y;
  $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```

```
<?php
$x = 5;
$y = 10;

function myTest() {
  $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

PHP Variables (cont'd)

- **PHP The static Keyword** *Try it!*
  - Sometimes we want a local variable NOT to be deleted
  - To do this, use the **static** keyword when you first declare the variable
  - Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.
  - **Note**: The variable is still local to the function.

```
<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
myTest();
myTest();
?>
```

- *PHP Exercise*

PHP echo and print Statements

- **PHP echo and print Statements**
  - Both
    - Used to output data to the screen
  - The differences
    - Echo
      - Has no return value
      - Can take multiple parameters (although such usage is rare)
      - Is marginally faster than print
    - Print
      - Has a return value of 1 so it can be used in expressions.
      - Can take one argument

PHP echo and print Statements (cont'd)

- **The PHP echo Statement**
  - Can be used with or without parentheses : **echo** or **echo ( )**
  - **Display text** *Try it!*
  - **Display Variables** *Try it!*

- **The PHP print Statement**
  - Can be used with or without parentheses : **print** or **print( )**
  - **Display text** *Try it!*
  - **Display Variables** *Try it!*

PHP Data Types

- **PHP Data Types**
  - PHP supports the following data types
    - String, Integer, Float(floating point numbers-also called double), Boolean, Array, Object, NULL, Resource

- **PHP String** *Try it!*
  - Can be any text inside quotes.
  - You can use single or double quotes.

PHP Data Types (cont'd)

- **PHP Integer** *Try it!*
  - An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647
  - Rules for integers
    - An integer must have at least one digit
    - An integer must not have a decimal point
    - An integer can be either positive or negative
    - Integers can be specified in three formats: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation
- **PHP Float** *Try it!*
  - A float (floating point number) is a number with a decimal point or a number in exponential form

PHP Data Types (cont'd)

- **PHP Boolean**
  - Represents two possible states

```
$x = true;
$y = false;
```

- **PHP Array** *Try it!*
  - An array stores multiple values in one single variable

- **PHP Object** *Try it!*
  - If you create a __construct( ) function, PHP will automatically call this function when you create an object from a class

PHP Data Types (cont'd)

- **PHP NULL Value** *Try it!*
  - A variable of data type NULL is a variable that has no value assigned to it
  - **Tip** : If a variable is created without a value, it is automatically assigned a value of NULL
  - Variables can also be emptied by setting the value to NULL

- **PHP Resource**
  - The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP
  - A common example of using the resource data type is a database call.

PHP Strings

- **PHP String Functions**
  - In this chapter we will look at some commonly used functions to manipulate strings.

- **strlen() - Return The Length of a String** *Try it!*
  - Returns the length of a string

- **str_word_count() - Count Words in a String** *Try it!*
  - Counts the number of words in a string

- **strrev() - Reverse a String** *Try it!*
  - Reverse a string

PHP Strings (cont'd)

- **strpos() - Search For a Text Within a String** *Try it!*
  - Searches for a specific text within a string
  - If a match is found, the function returns the character position of the first match.
  - If no match is found, it will return FALSE
  - **Tip** : The first character position in a string is 0 (not 1)

- **str-_replace() - Replace Text Within a String** *Try it!*
  - Replaces some characters with some other characters in a string.

- *PHP String Reference*
- *PHP Exercises*

PHP Numbers

- **PHP Integers** *Try it!*
  - A non-decimal number between -2147483648 and 2147483647
  - To check if the type of a variable is integer
    - is_int(), is_integer()-alias of is_int(), is_long()-alias of is_int()
- **PHP Floats** *Try it!*
  - Can commonly store a value up to 1.7976931348623E+308 (platform dependent), and have a maximum precision of 14 digits
  - To check if the type of a variable is float
    - is_float(), is_double()-alias of is_float()

PHP Numbers (cont'd)

- **PHP Infinity** *Try it!*
  - To check if a numeric value is finite or infinite
    - is_finite(), is_infinite()

- **PHP NaN** *Try it!*
  - Not a Number
  - To check if a value is not a number
    - is_nan()

- **PHP Numerical Strings** *Try it!*
  - The PHP is_numeric() function can be used to find whether a variable is numeric.

- **PHP Casting Strings and Floats to Integers** *Try it!*

PHP Math

- PHP pi( ) Function Try it!
- PHP min( ) and max( ) Functions Try it!
- PHP abs( ) Function Try it!
- PHP sqrt( ) Function Try it!
- PHP round( ) Function Try it!
- Random Numbers Try it! Try it!

PHP Constants

- **PHP Constants**
  - A constant is an identifier (name) for a simple value. The value cannot be changed during the script
  - A valid constant name starts with a letter or underscore (no $ sign before the constant name)
  - **Note:** Unlike variables, constants are automatically global across the entire script

PHP Constants (cont'd)

- **Create a PHP Constant** *Try it!* *Try it!*
  - To create a constant, use the **define( )** function
  - Syntax

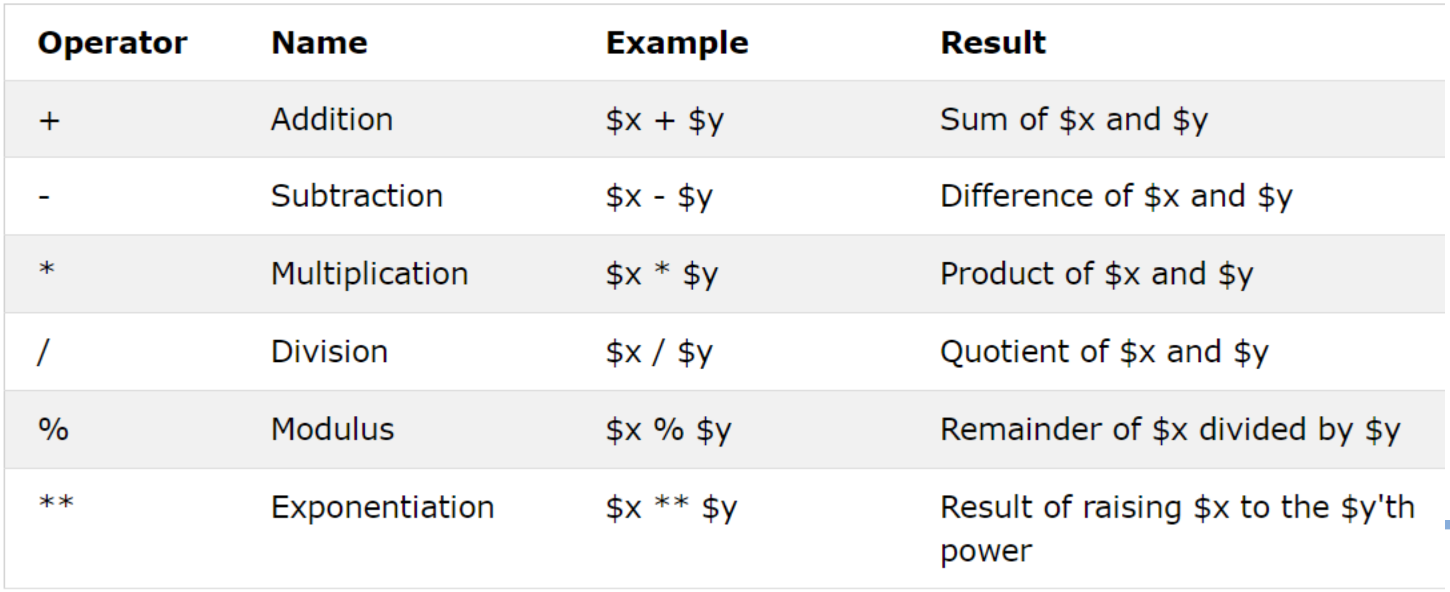
```
define(name, value, case-insensitive)
```

    - name : Specifies the name of the constant
    - *value*: Specifies the value of the constant
    - *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false
- **PHP Constant Arrays** *Try it!*
  - In PHP7, you can create Array constant using the **define( )** function
- **Constants are Global** *Try it!*
  - Constants are automatically global and can be used across the entire script

PHP Operators

- **PHP Operators**
  - PHP divides the operators in the following groups
    - Arithmetic operators, Assignment operators, Comparison operators, Increment/Decrement operators, Logical operators, String operators, Array operators, Conditional assignment operators

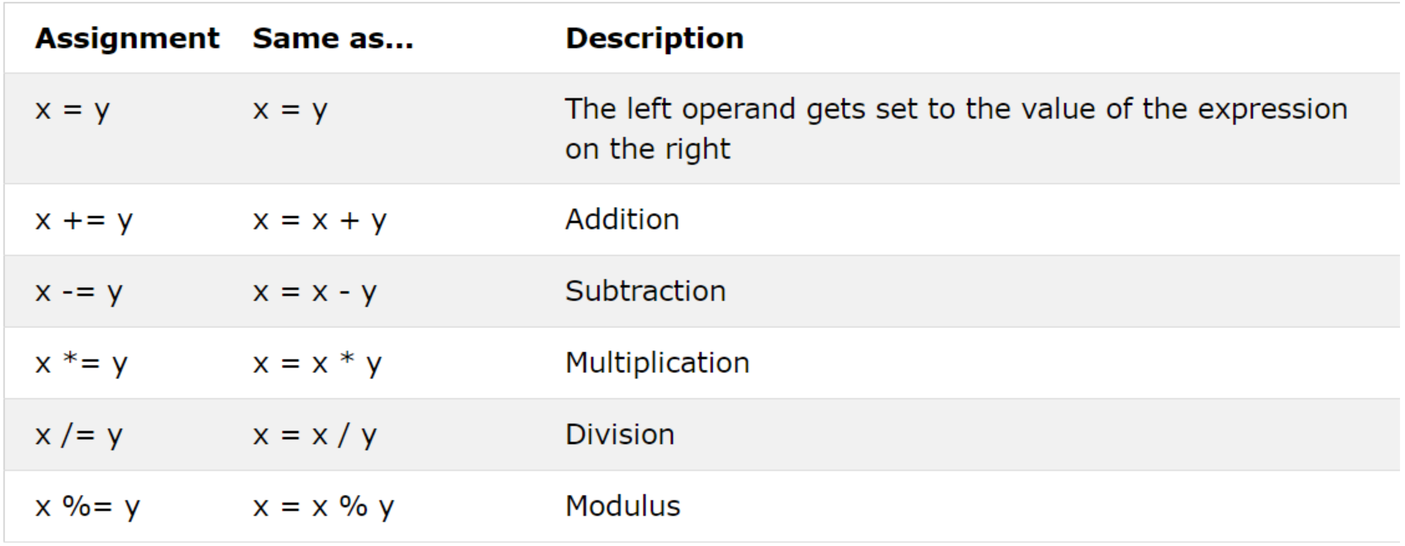
- **PHP Arithmetic Operators**



Operator	Name	Example	Result	
+	Addition	$x + $y	Sum of $x and $y	*Try it!*
-	Subtraction	$x - $y	Difference of $x and $y	*Try it!*
*	Multiplication	$x * $y	Product of $x and $y	*Try it!*
/	Division	$x / $y	Quotient of $x and $y	*Try it!*
%	Modulus	$x % $y	Remainder of $x divided by $y	*Try it!*
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power	*Try it!*

PHP Operators (cont'd)

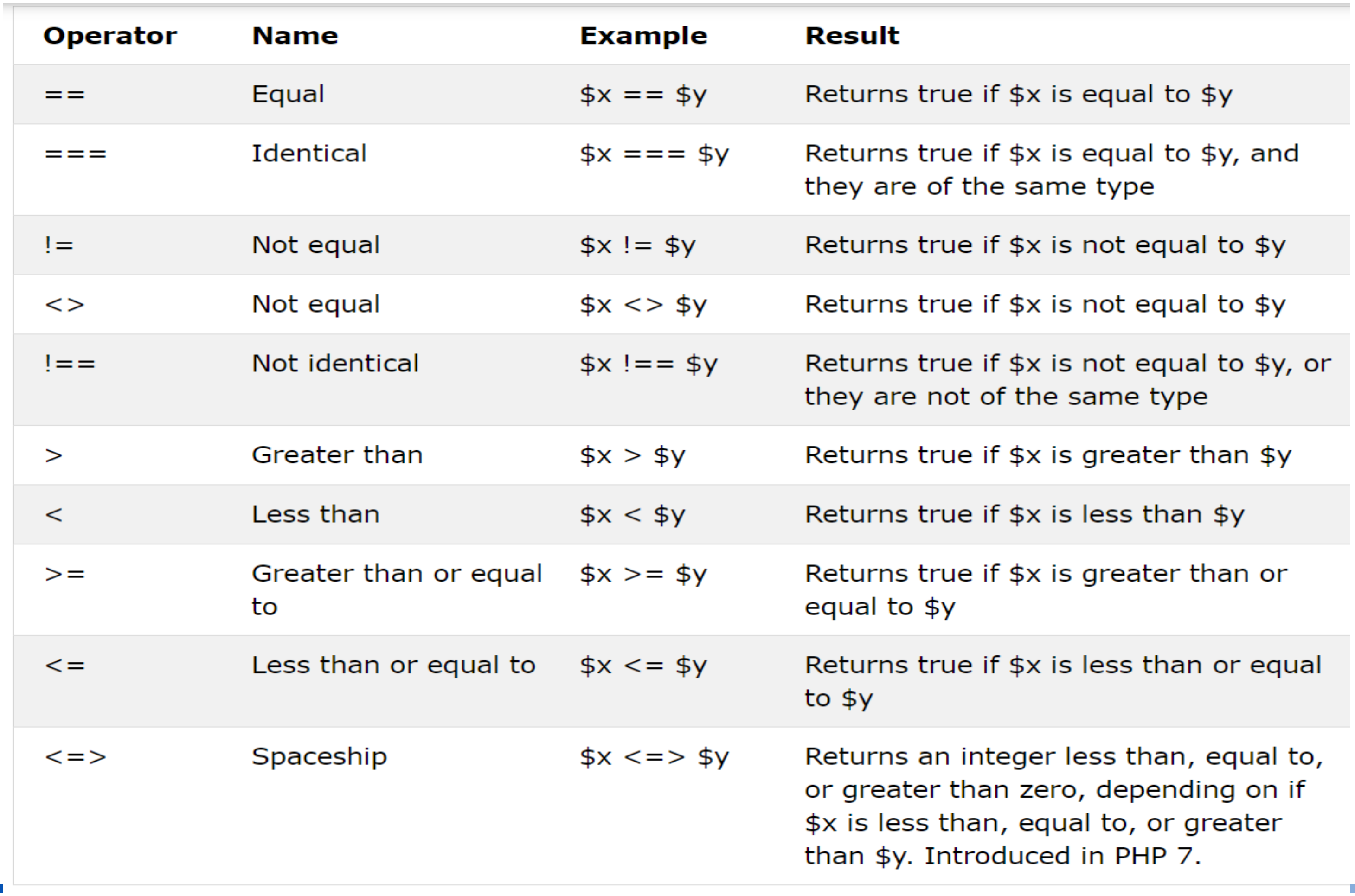
- PHP Assignment Operators



Assignment	Same as...	Description	
x = y	x = y	The left operand gets set to the value of the expression on the right	Try it!
x += y	x = x + y	Addition	Try it!
x -= y	x = x - y	Subtraction	Try it!
x *= y	x = x * y	Multiplication	Try it!
x /= y	x = x / y	Division	Try it!
x %= y	x = x % y	Modulus	Try it!

PHP Operators (cont'd)

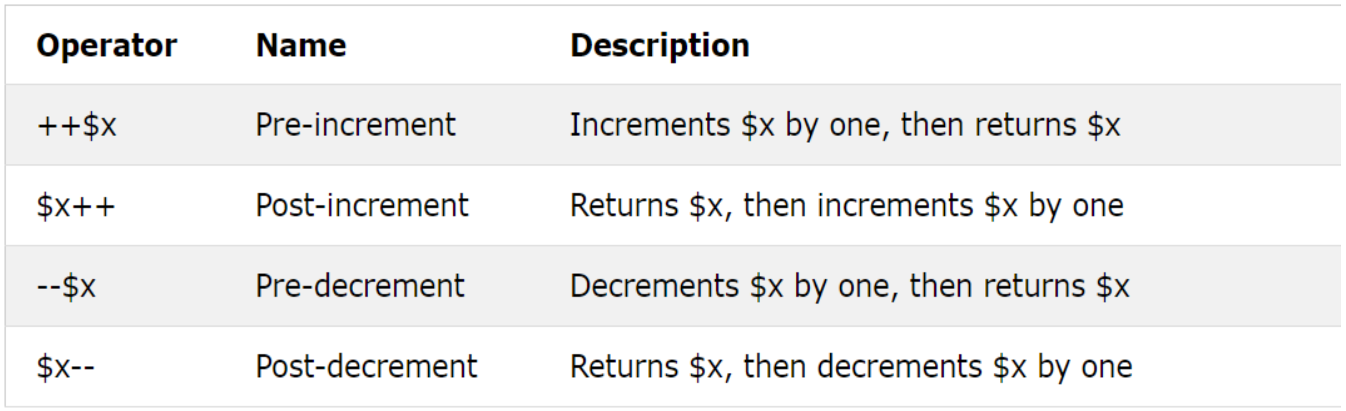
PHP Comparison Operators



Operator	Name	Example	Result	
==	Equal	$x == $y	Returns true if $x is equal to $y	*Try it!*
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type	*Try it!*
!=	Not equal	$x != $y	Returns true if $x is not equal to $y	*Try it!*
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y	*Try it!*
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type	*Try it!*
>	Greater than	$x > $y	Returns true if $x is greater than $y	*Try it!*
<	Less than	$x < $y	Returns true if $x is less than $y	*Try it!*
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y	*Try it!*
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y	*Try it!*
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.	*Try it!*

PHP Operators (cont’d)

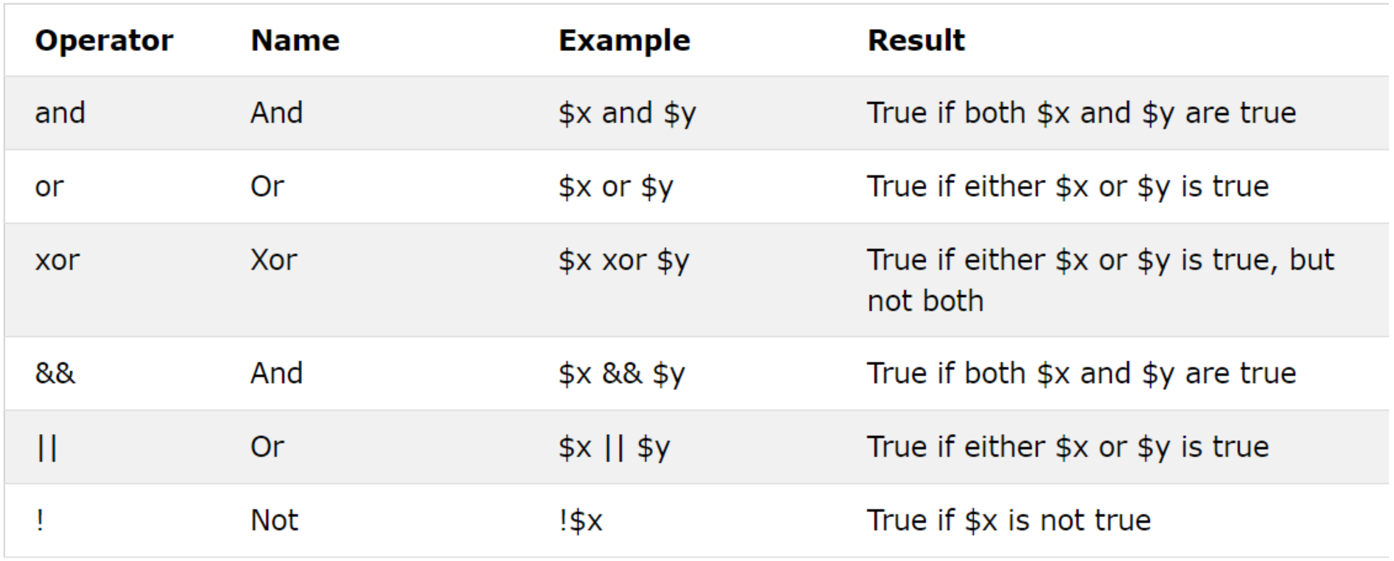
- **PHP Increment / Decrement Operators**



Operator	Name	Description	
++$x	Pre-increment	Increments $x by one, then returns $x	*Try it!*
$x++	Post-increment	Returns $x, then increments $x by one	*Try it!*
--$x	Pre-decrement	Decrements $x by one, then returns $x	*Try it!*
$x--	Post-decrement	Returns $x, then decrements $x by one	*Try it!*

PHP Operators (cont'd)

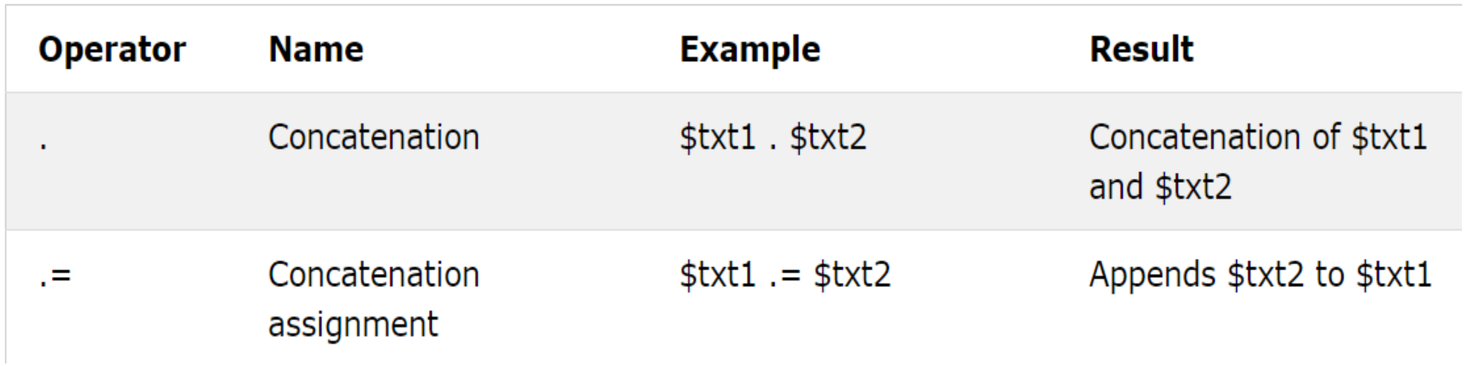
- **PHP Logical Operators**



Operator	Name	Example	Result	
and	And	$x and $y	True if both $x and $y are true	*Try it!*
or	Or	$x or $y	True if either $x or $y is true	*Try it!*
xor	Xor	$x xor $y	True if either $x or $y is true, but not both	*Try it!*
&&	And	$x && $y	True if both $x and $y are true	*Try it!*
||	Or	$x || $y	True if either $x or $y is true	*Try it!*
!	Not	!$x	True if $x is not true	*Try it!*

PHP Operators (cont'd)

- **PHP String Operators**



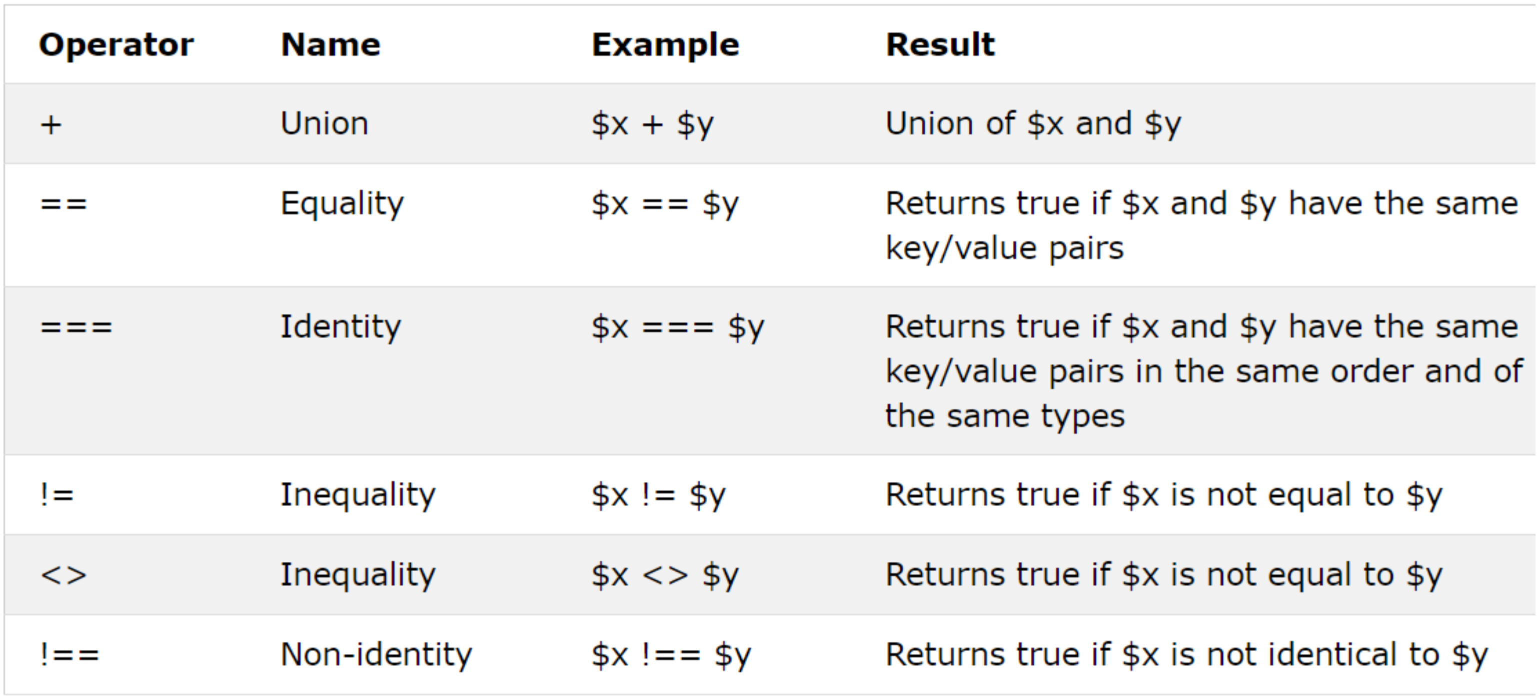
Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

Try it!

Try it!

PHP Operators (cont'd)

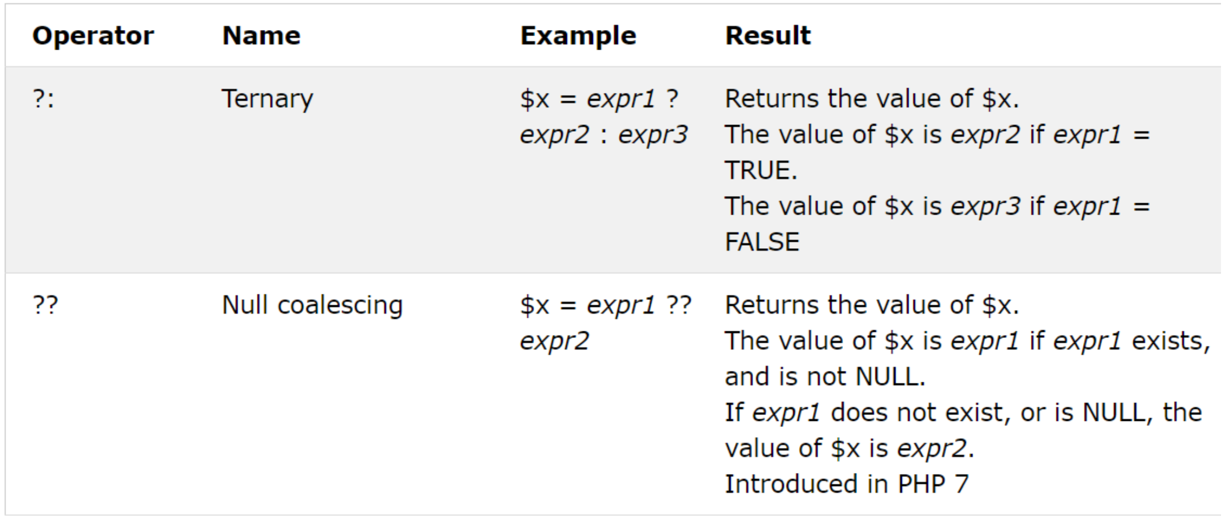
- PHP Array Operators



Operator	Name	Example	Result	
+	Union	$x + $y	Union of $x and $y	Try it!
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs	Try it!
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types	Try it!
!=	Inequality	$x != $y	Returns true if $x is not equal to $y	Try it!
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y	Try it!
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y	Try it!

PHP Operators (cont'd)

- PHP Conditional Assignment Operators



Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

Try it!

Try it!

- *PHP Exercises*

PHP if .. else... elseif Statement

- **PHP Conditional Statements**
  - **if** statement - executes some code if one condition is true
  - **if ... else** statement - executes some code if a condition is true and another code if that condition is false
  - **if ... elseif...else** statement - executes different codes for more than two conditions
  - **switch** statement - selects one of many blocks of code to be executed

PHP if .. else... elseif Statement (cont'd)

- **PHP – The if Statement** *Try it!*
  - The **if** statement executes some code if one condition is true.
  - Syntax

```
if (condition) {
    code to be executed if condition is true;
}
```

- **PHP – The if... else Statement** *Try it!*
  - The **if...else** statement executes some code if a condition is true and another code if that condition is false
  - Syntax

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

PHP if .. else… elseif Statement (cont'd)

- PHP – The if...elseif...else Statement *Try it!*
  - **The if...elseif...else** statement executes different codes for more than two conditions.
  - Syntax

```
if (condition) {
    code to be executed if this 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;
}
```

- *PHP Exercises*

PHP switch Statement

- **The PHP switch Statement** *Try it!*
  - Use the **switch** statement to **select one of many blocks of code to be executed**.
  - Syntax

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

- *PHP Exercises*

PHP Loops

- PHP Loops
  - **while** - loops through a block of code as long as the specified condition is true
  - **do**...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
  - **for** - loops through a block of code a specified number of times
  - **foreach** - loops through a block of code for each element in an array

PHP while Loop

- **The PHP while Loop** *Try it!* *Try it!*
  - The **while** loop executes a block of code as long as the specified condition is true.
  - Syntax

```
while (condition is true) {
    code to be executed;
}
```

- *PHP Exercises*

PHP do while Loops (cont'd)

- **The PHP do...while Loop** *Try it!* *Try it!*
  - The **do...while** loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true
  - Syntax

```
do {
    code to be executed;
} while (condition is true);
```

PHP for Loop

- **The PHP for Loop** *Try it!* *Try it!*
  - The **for** loop is used when you know in advance how many times the script should run.
  - Syntax

```
for (init counter; test counter; increment counter) {
    code to be executed;
}
```

- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

- *PHP Exercises*

PHP foreach Loop

- **The PHP foreach Loop** *Try it!* *Try it!*
  - The **foreach** loop works only on arrays, and is used to loop through each key/value pair in an array.
  - Syntax

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

    - For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element

PHP Break and Continue

- **PHP Break** *Try it!*
- **PHP Continue** *Try it!*
  - Breaks one iteration (in the loop)
- **Break and Continue in While Loop** *Try it!* *Try it!*

PHP Functions

- **PHP Built-in Functions** *Try it!*
  - PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

- **Create a User Defined Function in PHP** *Try it!*
  - A user-defined function declaration starts with the word **function**
  - Syntax

```
function functionName() {
    code to be executed;
}
```

  - **Note** : A function name can start with a letter or underscore. Function names are NOT case-sensitive
  - **Tip:** Give the function a name that reflects what the function does!

PHP Functions (cont'd)

- **PHP Function Arguments** *Try it!* *Try it!*
  - Information can be passed to functions through arguments. An argument is just like a variable
  - Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma

PHP Functions (cont'd)

- **PHP is a Loosely Typed Language**
  - PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing and error.
  - In PHP 7, type declarations were added. This gives us an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch *Try it!*
  - To specify **strict** we need to set **declare(strict_types = 1);** This must be the on the very first line of the PHP file. *Try it!*
  - The **strict** declaration forces things to be used in the intended way

PHP Functions (cont'd)

- **PHP Default Argument Value** *Try it!*

- **PHP Functions – Returning values** *Try it!*

- **PHP Return Type Declarations** *Try it!* *Try it!*
  - PHP 7 also supports Type Declarations for the **return** statement. Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.
  - To declare a type for the function return, add a colon (**:**) and the type right before the opening curly (**{**) bracket when declaring the function

PHP Functions (cont'd)

- **Passing Arguments by Reference** *Try it!*
  - When a function argument is passed by reference, changes to the argument also change the variable that was passed in. To turn a function argument into a reference, the & operator is used

- *PHP Exercises*

PHP Arrays *Try it!*

What is an Array?

- An array is a special variable, which can hold more than one value at a time.
- An array can hold many values under a single name, and you can access the values by referring to an index number.

Create an Array in PHP

- In PHP, the **array( )** function is used to create an array

```
array();
```

- In PHP, there are three types of arrays
  - **Indexed arrays** - Arrays with a numeric index
  - **Associative arrays** - Arrays with named keys
  - **Multidimensional arrays** - Arrays containing one or more arrays

PHP Arrays

- Get The Length of an Array – The count( ) Function *Try it!*
- *Complete PHP Array Reference*
- *PHP Exercises*

PHP Indexed Arrays

- **PHP Indexed Arrays** *Try it!*
  - Two ways to create indexed arrays
    - The index can be assigned automatically (index always starts at 0), like this

```
$cars = array("Volvo", "BMW", "Toyota");
```

  - or the index can be assigned manually:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

PHP Indexed Arrays (cont'd)

- **Loop Through an Indexed Array** *Try it!*
  - To loop through and print all the values of an indexed array, you could use a **for** loop

- *PHP Exercises*

PHP Associative Arrays

- **PHP Associative Arrays** *Try it!*
  - Associative arrays are arrays that use named keys that you assign to them
  - Two ways to create an associative array

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

or:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

PHP Associative Arrays (cont'd)

- **Loop Through an Associative Array** *Try it!*
  - To loop through and print all the values of an associative array, you could use a **foreach** loop

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}
?>
```

- *PHP Exercises*

PHP Multidimensional Arrays

- An array containing one or more arrays

PHP - Two-dimensional Arrays *Try it!* *Try it!*

- An array of arrays (a three-dimensional array is an array of arrays of arrays).

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

PHP Sorting Arrays

- **PHP – Sort Functions For Arrays**
  - **sort ( )** - sort arrays in ascending order
  - **rsort ( )** – sort arrays in descending order
  - **asort( )** – sort associative arrays in ascending order, according to the value
  - **ksort( )** –sort associative arrays in ascending order, according to the key
  - **arsort( )** –sort associative arrays in descending order, according to the value
  - **krsort( )** –sort associative arrays in descending order, according to the key

- **Sort Array in Ascending Order – sort( )** *Try it!* *Try it!*

- **Sort Array in Descending Order – rsort( )** *Try it!* *Try it!*

PHP Sorting Arrays (cont'd)

- Sort Array (Ascending Order), According to Value – asort( ) *Try it!*
- Sort Array (Ascending Order), According to Key – ksort( ) *Try it!*
- Sort Array (Descending Order), According to Value – arsort( ) *Try it!*
- Sort Array (Descending Order), According to Key – krsort( ) *Try it!*
- *PHP Exercises*

PHP Global Variables - Superglobals

- Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes

- **PHP Global Variables – Superglobals**
  - "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special
    - $GLOBALS, $_SERVER, $_REQUEST, $_POST, $_GET, $_FILES, $_ENV, $_COOKIE, $_SESSION

PHP Global Variables – Superglobals (cont'd)

- **PHP $GLOBALS** *Try it!*
  - A PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods)
  - PHP stores all global variables in an array called $GLOBALS[*index*].
  - The *index* holds the name of the variable

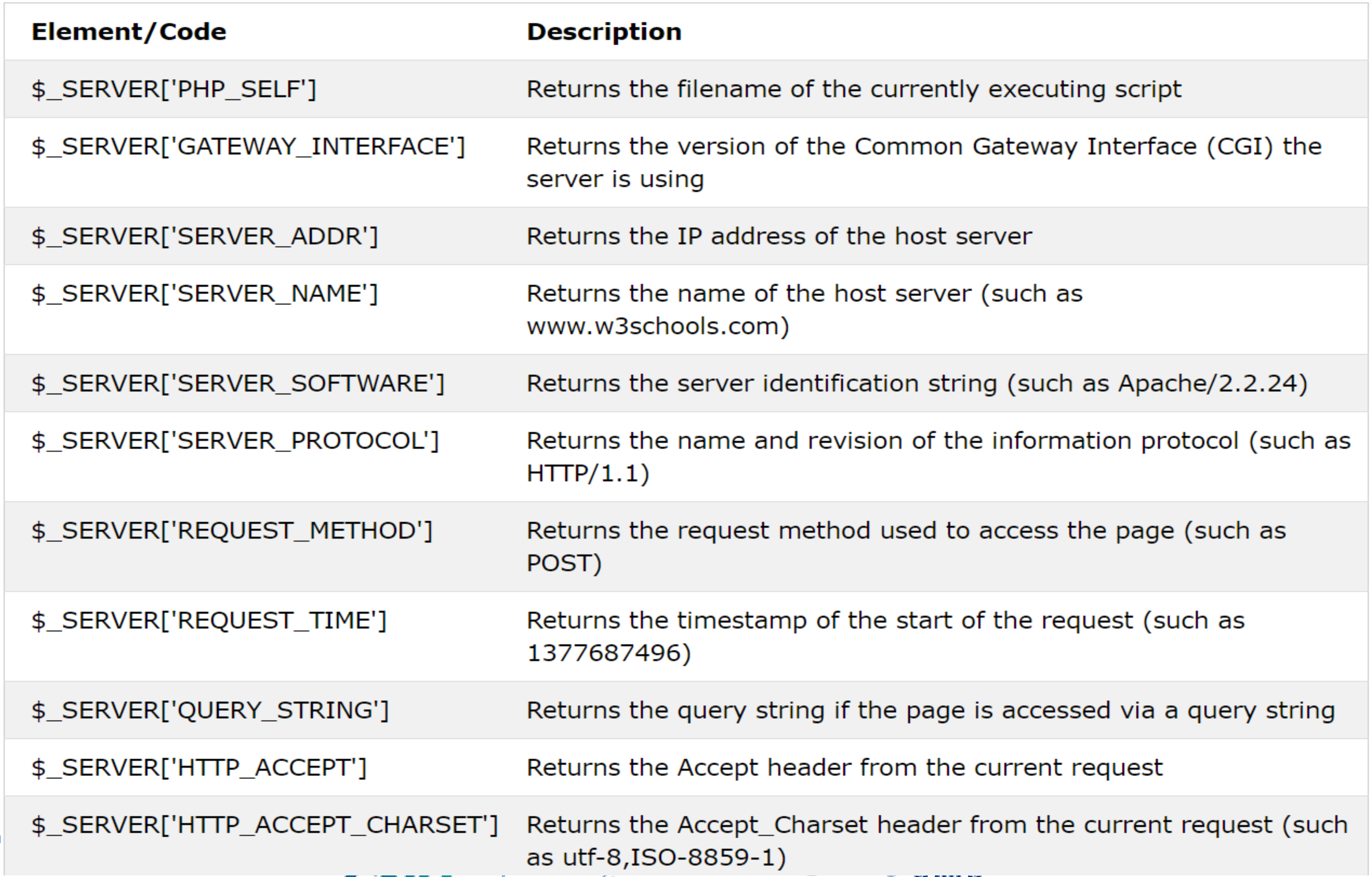
```php
<?php
$x = 75;
$y = 25;

function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

PHP Global Variables – Superglobals (cont'd)

- **PHP $_SERVER** *Try it!*
  - $_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

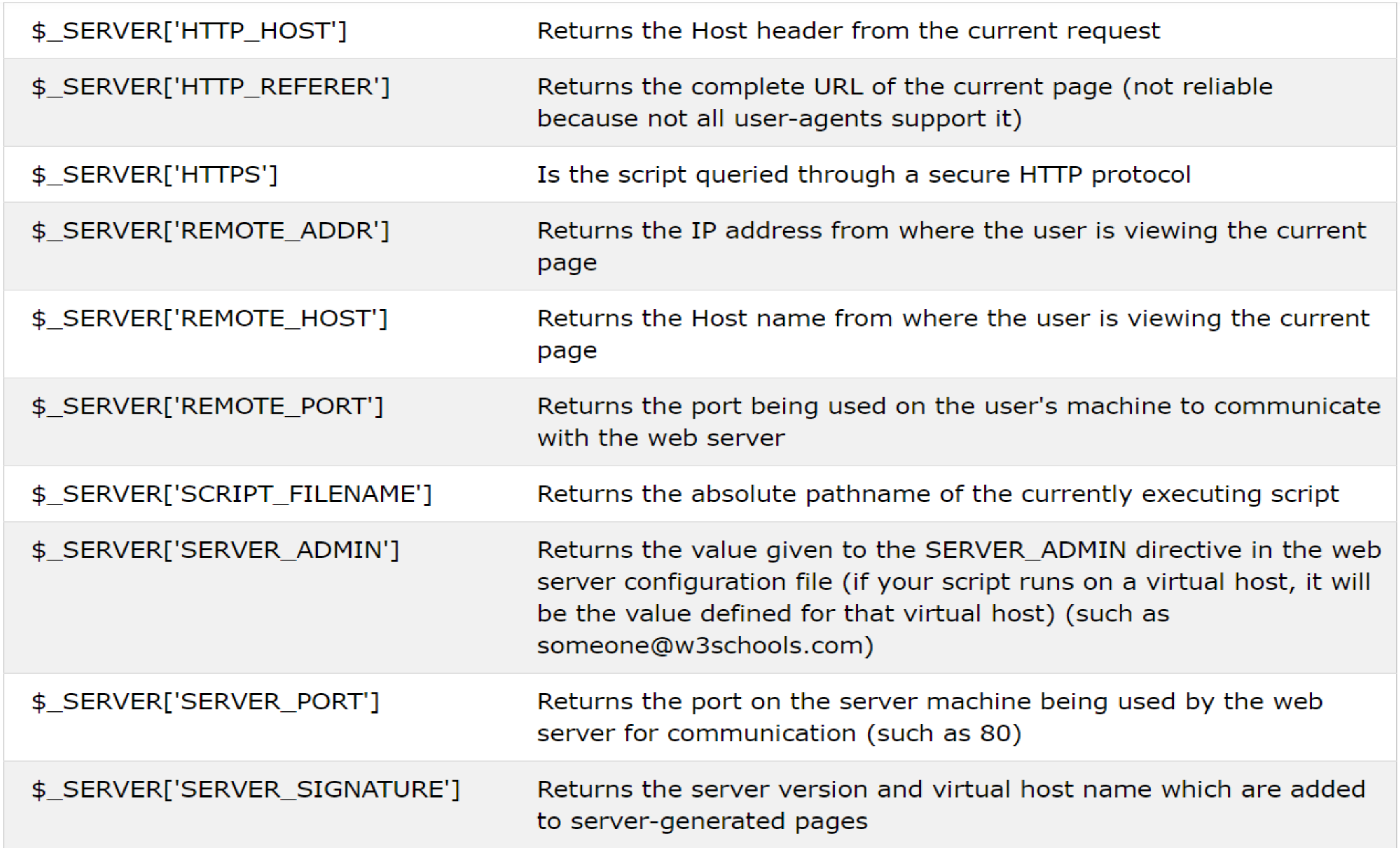
```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

PHP Global Variables – Superglobals (cont'd)



Element/Code	Description
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)

PHP Global Variables – Superglobals (cont'd)



$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@w3schools.com)
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80)
$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages

PHP Global Variables – Superglobals (cont’d)

$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page

- **PHP $_REQUEST** *Try it!*
  - Is used to collect data after submitting an HTML form.

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_REQUEST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>
```

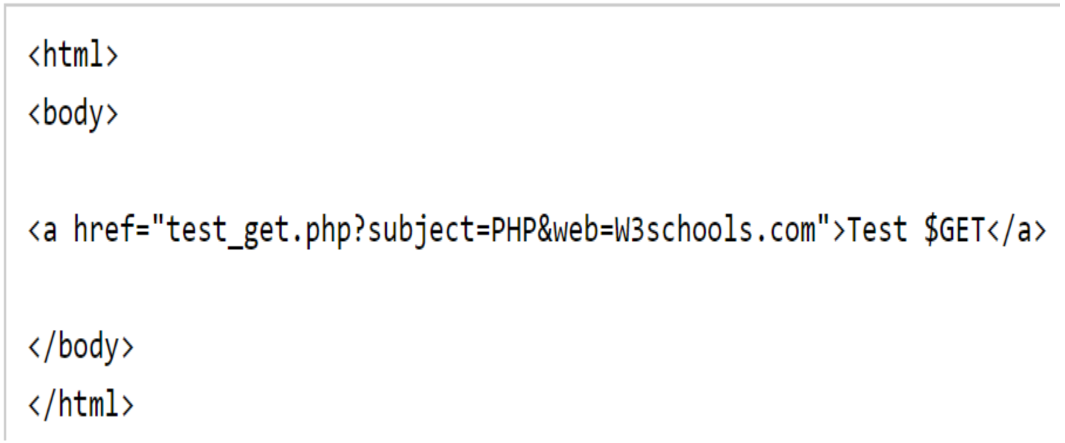
PHP Global Variables – Superglobals (cont'd)

- **PHP $_POST** *Try it!*
  - Is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

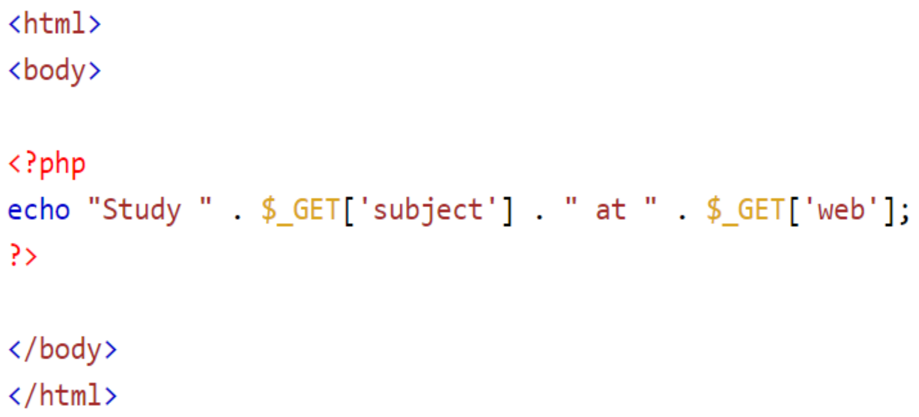
```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_POST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>
```

PHP Global Variables – Superglobals (cont'd)

- PHP $_GET *Try it!*
  - Can also be used to collect form data after submitting an HTML form with method="get". Can also collect data sent in the URL
  - Assume we have an HTML page that contains a hyperlink with parameters:



```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```



```
<html>
<body>

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>

</body>
</html>
```

- When a user clicks on the link "Test $GET", the parameters "subject" and "web" are sent to "test_get.php", and you can then access their values in "test_get.php" with $_GET.

PHP RegEx

- **What is a Regular Expression?**
  - A sequence of characters that forms a search pattern.
  - When you search for data in a text, you can use this search pattern to describe what you are searching for.

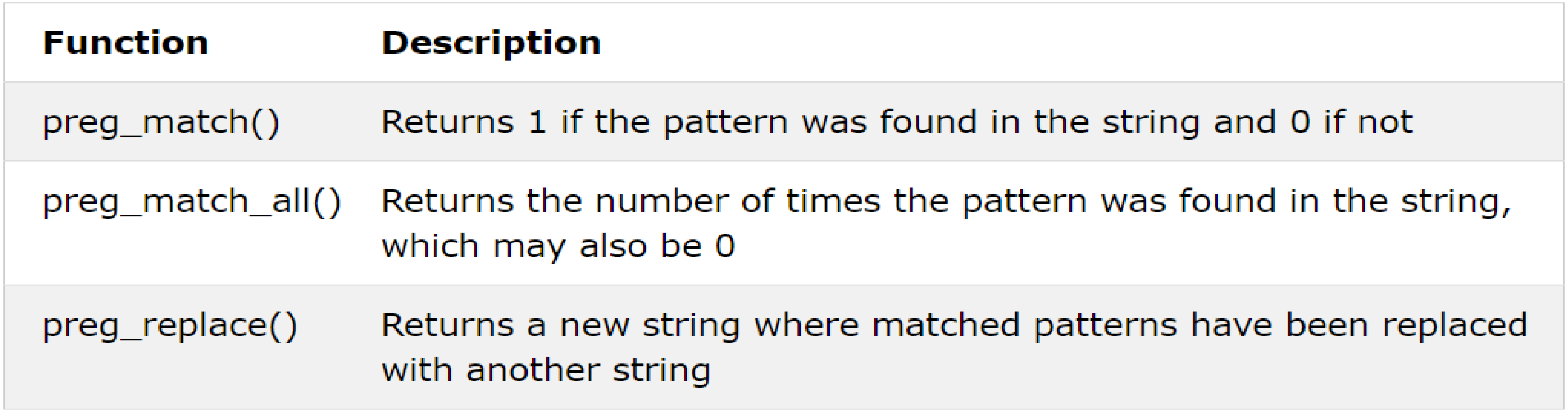
- **Syntax**

```
$exp = "/w3schools/i";
```

  - / : the delimiter
  - w3schools : the **pattern** that is being searched for
  - i : a **modifier** that makes the search case-insensitive.

PHP RegEx (cont'd)

- **Regular Expression Functions**



Function	Description
preg_match()	Returns 1 if the pattern was found in the string and 0 if not
preg_match_all()	Returns the number of times the pattern was found in the string, which may also be 0
preg_replace()	Returns a new string where matched patterns have been replaced with another string

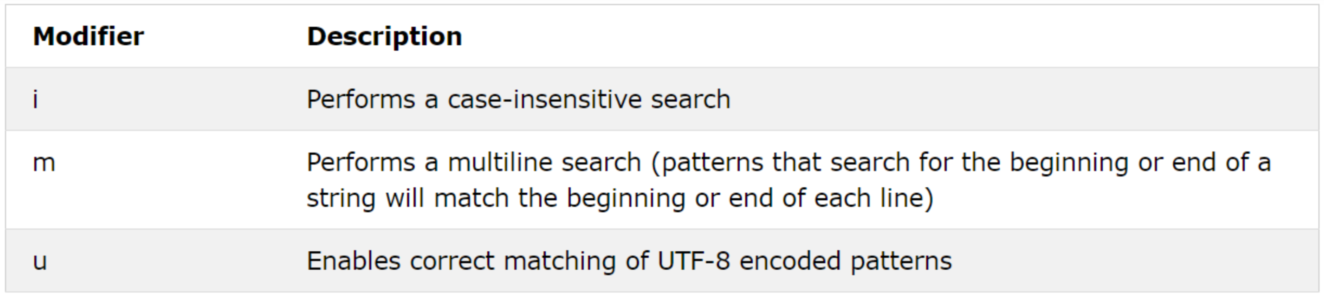
Try it!

Try it!

Try it!

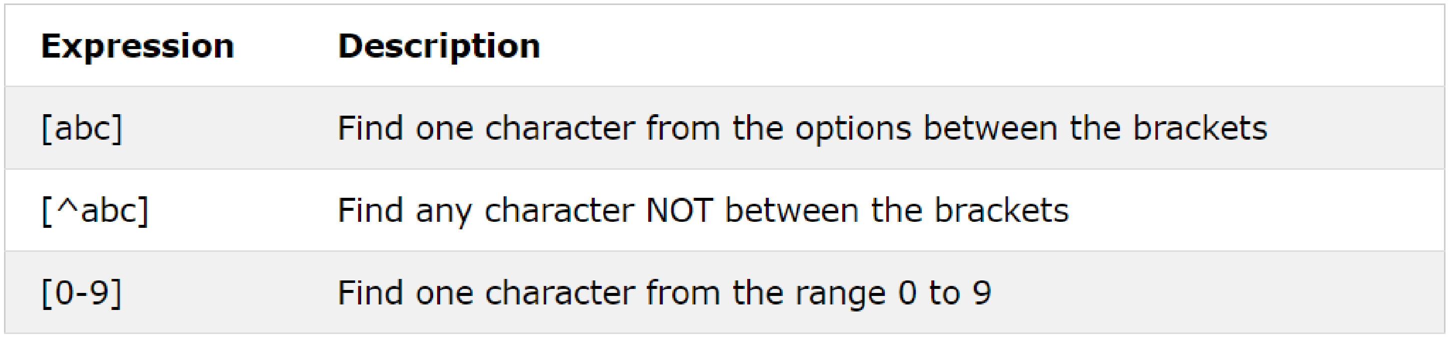
PHP RegEx (cont'd)

Regular Expression Modifiers



Modifier	Description
i	Performs a case-insensitive search
m	Performs a multiline search (patterns that search for the beginning or end of a string will match the beginning or end of each line)
u	Enables correct matching of UTF-8 encoded patterns

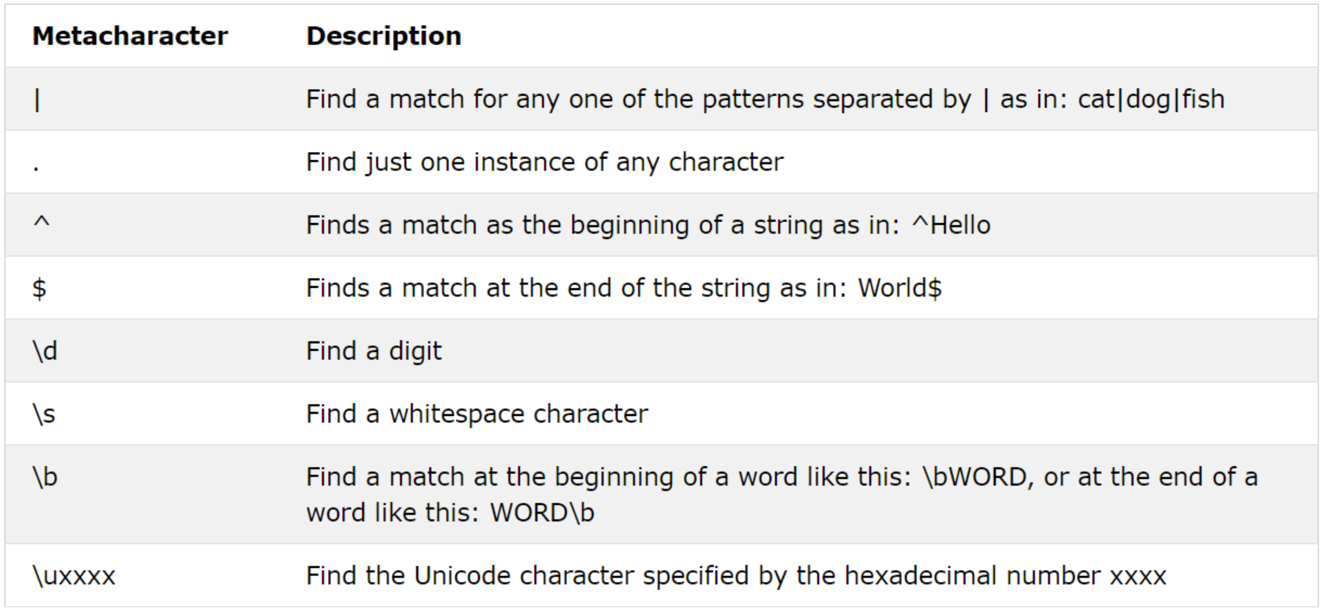
Regular Expression Patterns



Expression	Description
[abc]	Find one character from the options between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find one character from the range 0 to 9

PHP RegEx (cont'd)

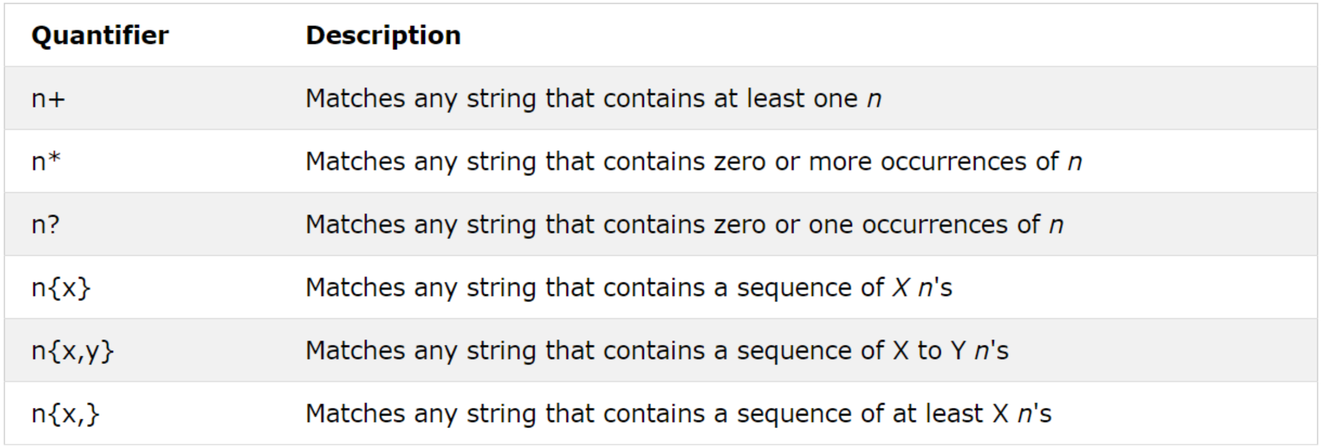
- **Metacharacters**
  - Are characters with a special meaning



Metacharacter	Description
|	Find a match for any one of the patterns separated by | as in: cat|dog|fish
.	Find just one instance of any character
^	Finds a match as the beginning of a string as in: ^Hello
$	Finds a match at the end of the string as in: World$
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

PHP RegEx (cont'd)

- **Quantifiers**



Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{x}	Matches any string that contains a sequence of *X* *n*'s
n{x,y}	Matches any string that contains a sequence of X to Y *n*'s
n{x,}	Matches any string that contains a sequence of at least X *n*'s

- **Grouping** *Try it!*
  - Can use parentheses **( )** to apply quantifiers to entire patterns