



Lebanese University
Faculty of Information

Branch II

Dr. Mohamad Nagi

PhD in Data Mining - Computer Science

Advanced Web Design

Semester 5

Web Design

The graphical user interface to information stored on computers running web servers connected to the Internet.



Domain Name

What is a **Domain Name**?

Whenever a person types your Domain Name, the hosting company transforms all the files connected to your IP address and returns all the images, videos and words on your website. Each domain name is unique as like your register address. Whenever you open a new website, you will be able to select a name that is a reflection of your brand.

Domain Name

- TLD (top level domain name)
 - .com, .net, .biz, .us, .mobi and others
 - .org for non-profits
- There are many domain name registrars, including
 - <http://register.com>
 - <http://networksolutions.com>
 - <http://godaddy.com>
- Visit Registrar, choose name, pay with credit card or Paypal
- Private registration – additional fee but personal information is kept private and out of the WHOIS database

Domain Name

› ICANN - The Internet Corporation for Assigned Numbers & Names

- Non-profit organization
- Main function is to coordinate the assignment of:
 - › Internet domain names
 - › IP address numbers
 - › Protocol parameters
 - › Protocol port numbers.



Domain Name

Locates an organization or other entity on the Internet

Domain Name System

- › Divides the Internet into logical groups and understandable names
- › Associates unique computer IP Addresses with the text-based domain names you type into a web browser
Browser: `http://google.com`
IP Address: `74.125.225.78`

IP Address

- › Each device connected to the Internet has a unique numeric IP address.
- › These addresses consist of a set of four groups of numbers, called octets.

74.125.225.78 will get you Google!

- › An IP address may correspond to a domain name.

Web Hosting

Web Hosting is the service that allows others to view your website on the Internet. A web host offers space on their server to allow other devices worldwide to access your website via a network or modem.



Web Client

Browser Request



Server Response



Web Server

Web Hosting

› **Client/Server** can describe a relationship between two computer programs – the "**client**" and the "**server**".

› **Client**

requests some type of service (such as a file or database access) from the server.

› **Server**

fulfills the request and transmits the results to the client over a network

Web Hosting

Web Client

- › Connected to the Internet when needed
- › Usually runs web browser (client) software
(*such as Internet Explorer or Firefox*)
- › Uses HTTP (Hypertext Transfer Protocol)
- › Requests web pages from server
- › Receives web pages and files from server

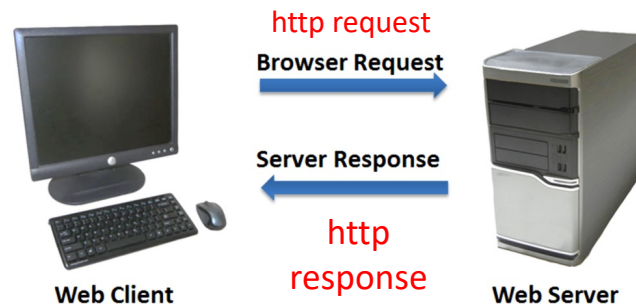
Web Hosting

Web Server

- › Continually connected to the Internet
- › Runs web server software
(*such as Apache or Internet Information Server*)
- › Uses HTTP (Hypertext Transfer Protocol)
- › Receives request for the web page
- › Responds to request and transmits status code, web page, and associated files

Web Hosting

- › A set of rules for exchanging files such as text, graphic images, sound, video, and other multimedia files on the Web.



- › Web browsers send HTTP requests for web pages and their associated files.
- › Web servers send HTTP responses back to the web browsers.

Web Hosting

Types of Web Hosting

Below are the different types of web hosting as follows:

- › Shared Hosting
- › VPS Hosting
- › Dedicated Server Hosting
- › Cloud Hosting
- › WordPress hosting

Web Hosting



Shared Hosting:

Hosted websites on the same server share all their memory, computer storage, disk storage, and other resources. there are various disadvantages such as many websites share the same server which leads to having the same resources and slowing down the websites.

If one website consumes more power due to increased traffic or faulty code, others have to face the problem. This phenomenon called the “bad neighbor effect”.

Web Hosting



VPS Hosting:

VPS stands for Virtual Private Server Hosting. However, you still share a server with other users, however on the server your web host gives you a separate partition. You will have dedicated server space, as well as reserved computer power and memory. VPS hosting can be excellent for medium-sized enterprises with fast increasing websites.

VPS provides more reliability and stability than shared hosting.

Web Hosting



Dedicated Server Hosting:

In a dedicated hosting type, the website owner has full control over the server on which the website is stored. i.e. the owner has complete admin and root access. Because of this, it allows faster executions and high-level security and all the resources are under the control of the owner.

One of the main advantages of dedicated hosting is that there is no problem of bad neighbor effect as no other websites are involved to use the website's resources.

Web Hosting



Word-press hosting:

Word-Press Hosting is a little bite like Shared hosting, but this was created for owners of WordPress sites. Your server is specifically configured to WordPress and there are plugins installed for important items such as caching and safety on your site. Because your site is extremely optimized, it loads more quickly and runs less troubleshoot the problem.

Web Hosting



Cloud Hosting:

Cloud hosting is currently, with no downtime, the most reliable solution on the market. It is the same as VPS hosting as in this type virtual machines are used. Your host offers a cluster of servers with cloud hosting. Each server is replicated for your files and resources.

If one of the cloud servers is busy or issues, your traffic is routed to another server in the cluster automatically. It is more scalable as it combines several computers into a virtual server that can then provide its resources based on their requirements.

Web Sites

Static Websites

A **Static Website** is displayed in a web browser exactly as it is stored. It contains web pages with fixed content coded in HTML and stored on a web server. It does not change, it stays the same, or "static" for every viewer of the site.

A static website does not require web programming or database design. Static is the most basic form of website and are the most simple to create and are perfect for small scale sites. Maintaining a large number of static pages can become a time consuming and impractical hassle very quickly. If you require a website with hundreds of pages and a ton of content, a dynamic website may be for you.

Web Sites

Static Websites

Advantages of static websites

- › Static websites take less time to build and deploy than dynamic websites.
- › You can duplicate the basic code of your static web pages to keep elements consistent while making slight changes to differentiate.
- › Static websites tend to be more secure.
- › It's easier for search engines to rank static websites because they usually load faster.
- › Building static websites doesn't require complex software.
- › You can change the layout and design of every page on a static website.
- › It's easier to restore a static website after a crash or [DDoS attack](#) by redeploying the codebase.

Web Sites

Static Websites

Disadvantages of static websites

- › Making updates to static websites can be challenging and time-consuming, especially if they're large.
- › Once you've built the basic structure, it can be hard to scale a website to add new content.
- › Because of the effort needed to update static websites, yours may end up with stagnant content that makes your company or brand feel outdated.
- › You can't interact with or create unique experiences for visitors.

Web Sites

Dynamic Websites

A **Dynamic Website** requires web programming and database design. A dynamic website contains information and content that changes, depending on factors such as the viewer of the site, the time of the day, the time zone, or the native language of the country the viewer).

The content of your site (text/images) is stored on a database. When the information is updated or changed within the database, it changes on the site.

When a dynamic site is accessed, pages are generated on-the-fly (or in *real-time*) to the user based on this data from the visitor.

Web Sites

Dynamic Websites



Front End

- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation



Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

Web Sites

Dynamic Websites

Advantages of dynamic websites

- › They provide more website functionality and enable user interaction.
- › Dynamic sites let you request and store information in an organized way.
- › They display content based on the user's needs.
- › These type of websites enable additional website flexibility by allowing connection to a CMS.
- › They allow multiple users to adjust the content.
- › It is less costly to make adjustments and changes versus a static website.
- › Dynamic sites are more likely to attract recurring customers and visitors.

Web Sites

Dynamic Websites

Disadvantages of dynamic websites

- › Functions like building the site foundation, creating database connections, and adding other features can make a dynamic website more expensive than a static website.
- › There can be design and layout limitations when it comes to displaying content on your dynamic website.



Top 5 WordPress Alternatives:

➤ 1. Joomla

Joomla is also another Content Management system which works on model view controller architecture. It is one of the biggest competitions for WordPress.

➤ 2. Drupal

Drupal is another alternative to WordPress which is more digital. Being preferred for digital solutions Drupal is not suitable for beginners to use. The various advantages of Drupal include creating blogs, personal websites, forums, and even social networking sites.



Top 5 WordPress Alternatives:

› 3. **Craft CMS**

Although Craft CMS is not that famous, it is extensively being used by Netflix and Salesforce websites. It is developer friendly due to its extensive back end features for your application.

› 4. **Shopify**

Unlike other sites, e-commerce sites also require alterations and demand to have something easy. Among all these WordPress alternatives, Shopify is the best alternative to WordPress in this regard. It is simple and helps in growing your business the easy way.

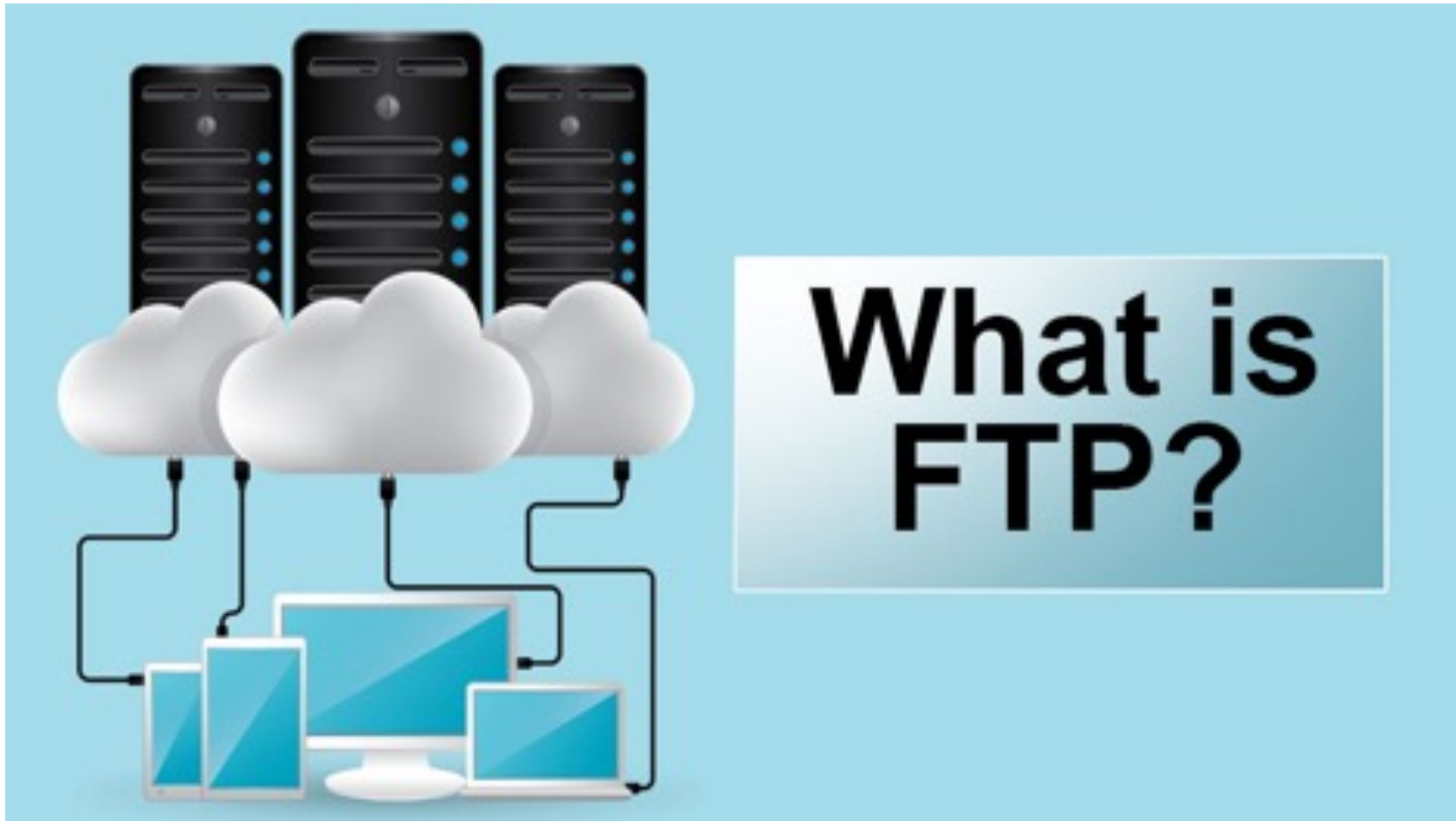


Top 5 WordPress Alternatives:

› 5. Wix

Wix is one of the oldest and widely used websites which is used to build sites. It was launched in 2006 and has been in the industry since then. One of the best catches of Wix is that it provides animation features that animate texts and other elements. It is one of the most intriguing factors of Wix.

> FTP



FTP

Introduction to FTP

FTP stands for File Transfer Protocol. It is a protocol that is used for transferring a file from one location to another i.e. from one host to another host. It is a standard mechanism that is provided by TCP/IP.

How does it work?

It uses TCP services. It needs two TCP connections. One is Control connection and another is Data connection. For control connection, it uses port 21 and for data connection, it uses port 20

➤ FTP

- A set of rules that allow files to be exchanged between computers on the Internet.
- Web developers commonly use FTP to transfer web page files from their computers to web servers.
- FTP is also used to download programs and files from other servers to individual computers.

E-Mail

-  ► Sending E-mail
 - SMTP Simple Mail Transfer Protocol
-  ► Receiving E-mail
 - POP (POP3) Post Office Protocol
 - IMAP Internet Mail Access Protocol

> PHP

- 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.





To Test your **PhP** code you can install locally on your pc a **PhP** compiler or by using the online compiler on the following link:
https://www.w3schools.com/php/phptryit.asp?filename=tryphp_compiler

PHP Syntax

Basic PHP Syntax

- › A PHP script can be placed anywhere in the document.
- › A PHP script starts with `<?php` and ends with `?>`:

```
<?php  
// PHP code goes here  
?>
```
- › The default file extension for PHP files is ".php".
- › A PHP file normally contains HTML tags, and some PHP scripting code.

PHP Syntax

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

PHP Syntax

PHP Case Sensitivity

In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.

In the example below, all three echo statements below are equal and legal:

```
<?php  
ECHO "Hello World!<br>";  
echo "Hello World!<br>";  
ECHO "Hello World!<br>";  
?>
```

PHP Syntax

PHP Case Sensitivity

Note: However; all variable names are case-sensitive!

Look at the example below; only the first statement will display the value of the **\$color** variable! This is because \$color, \$COLOR, and \$coLOR are treated as three different variables:

```
<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

PHP Comments

A comment in PHP code is a line that is not executed as a part of the program.

PHP supports several ways of commenting:

› Syntax for single-line comments:

```
<?php
```

```
// This is a single-line comment
```

```
# This is also a single-line comment
```

```
?>
```

PHP Comments

- › Syntax for multiple-line comments:

```
<?php
```

```
/*
```

```
This is a multiple-lines comment block  
that spans over multiple  
lines
```

```
*/
```

```
?>
```

PHP Comments

- › Using comments to leave out parts of the code:

```
<?php
```

```
// You can also use comments to leave out parts of a code line
```

```
$x = 5 /* + 15 */ + 5;
```

```
echo $x;
```

```
?>
```

PHP Variables

Variables are "containers" for storing information.

In PHP, a variable starts with the **\$** sign, followed by the name of the variable:

```
<?php  
$txt = "Hello world!";  
$x = 5;  
$y = 10.5;  
?>
```

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

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

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)

Remember that PHP variable names are case-sensitive!

PHP Variables

Output Variables

The PHP `echo` statement is often used to output data to the screen.

- › The following example will show how to output text and a variable:

```
<?php  
$txt = "W3Schools.com";  
echo "I love " . $txt . "!";  
?>
```

- › The following example will output the sum of two variables:

```
<?php  
$x = 5;  
$y = 4;  
echo $x + $y;  
?>
```

PHP echo and print Statements

With PHP, there are two basic ways to get output: `echo` and `print`.

PHP echo and print Statements

`echo` and `print` are more or less the same. They are both used to output data to the screen.

The differences are small:

- › `echo` has no return value while `print` has a return value of 1 so it can be used in expressions.
- › `echo` can take multiple parameters (although such usage is rare) while `print` can take one argument.
- › `echo` is marginally faster than `print`.

PHP echo and print Statements

The PHP echo Statement

The `echo` statement can be used with or without parentheses:
`echo` or `echo()`.

Display Text

The following example shows how to output text with the `echo` command (notice that the text can contain HTML markup):

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

PHP echo and print Statements

Display Variables

The following example shows how to output text and variables with the `echo` statement:

```
<?php
```

```
$txt1 = "Learn PHP";
```

```
$txt2 = "W3Schools.com";
```

```
$x = 5;
```

```
$y = 4;
```

```
echo "<h2>" . $txt1 . "</h2>";
```

```
echo "Study PHP at " . $txt2 . "<br>";
```

```
echo $x + $y;
```

```
?>
```

PHP echo and print Statements

The PHP print Statement

The `print` statement can be used with or without parentheses:
`print` or `print()`.

Display Text

The following example shows how to output text with the `print` command (notice that the text can contain HTML markup):

```
<?php  
print "<h2>PHP is Fun!</h2>";  
print "Hello world!<br>";  
print "I'm about to learn PHP!<br>";  
?>
```

PHP echo and print Statements

Display Variables

The following example shows how to output text and variables with the `print` statement:

```
<?php
```

```
$txt1 = "Learn PHP";
```

```
$txt2 = "W3Schools.com";
```

```
$x = 5;
```

```
$y = 4;
```

```
print "<h2>" . $txt1 . "</h2>";
```

```
print "Study PHP at " . $txt2 . "<br>";
```

```
print $x + $y;
```

```
?>
```

PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- › String
- › Integer
- › Float (floating point numbers - also called double)
- › Boolean
- › Array
- › Object
- › NULL
- › Resource

PHP Data Types

PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

```
<?php
```

```
$x = "Hello world!";
```

```
$y = 'Hello world!';
```

```
echo $x;
```

```
echo "<br>";
```

```
echo $y;
```

```
?>
```

PHP Data Types

PHP Integer

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: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example \$x is an integer. The PHP var_dump() function returns the data type and value:

```
<?php
```

```
$x = 5985;
```

```
var_dump($x);
```

```
?>
```

PHP Data Types

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example \$x is a float. The PHP var_dump() function returns the data type and value:

```
<?php  
$x = 10.365;  
var_dump($x);  
?>
```

PHP Data Types

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

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

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Data Types

PHP Array

An array stores multiple values in one single variable.

In the following example \$cars is an array. The PHP var_dump() function returns the data type and value:

```
<?php  
$cars = array("Volvo","BMW","Toyota");  
var_dump($cars);  
?>
```

You will learn a lot more about arrays in later chapters of this tutorial.

PHP Data Types

PHP Object

Classes and objects are the two main aspects of object-oriented programming.

A class is a template for objects, and an object is an instance of a class. When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Let's assume we have a class named Car. A Car can have properties like model, color, etc. We can define variables like \$model, \$color, and so on, to hold the values of these properties.

When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

If you create a `__construct()` function, PHP will automatically call this function when you create an object from a class.

PHP Data Types

PHP Object

```
<?php
class Car {
    public $color;
    public $model;
    public function __construct($color, $model) {
        $this->color = $color;
        $this->model = $model;
    }
    public function message() {
        return "My car is a " . $this->color . " " . $this->model . "!";
    }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
```



PHP Data Types

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. We will not talk about the resource type here, since it is an advanced topic.

PHP Strings

A string is a sequence of characters, like "Hello world!".

PHP String Functions

In this chapter we will look at some commonly used functions to manipulate strings.

`strlen()` - Return the Length of a String

The PHP `strlen()` function returns the length of a string.

Return the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

PHP Strings

`str_word_count()` - Count Words in a String

The PHP `str_word_count()` function counts the number of words in a string.

Example

Count the number of word in the string "Hello world!":

```
<?php
```

```
echo str_word_count("Hello world!"); // outputs 2
```

```
?>
```

PHP Strings

`strrev()` - Reverse a String

The PHP `strrev()` function reverses a string.

Example

Reverse the string "Hello world!":

```
<?php
```

```
echo strrev("Hello world!"); // outputs !dlrow olleH
```

```
?>
```



PHP Strings

`strpos()` - Search For a Text Within a String

The PHP `strpos()` function 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`.

Example

Search for the text "world" in the string "Hello world!":

```
<?php
```

```
echo strpos("Hello world!", "world"); // outputs 6
```

```
?>
```



PHP Strings

`str_replace()` - Replace Text Within a String

The PHP `str_replace()` function replaces some characters with some other characters in a string.

Example

Replace the text "world" with "Dolly":

```
<?php
```

```
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
```

```
?>
```

PHP Strings

Complete PHP String Reference

For a complete reference of all string functions, go to our complete [PHP String Reference](#).

The PHP string reference contains description and example of use, for each function!



PHP Numbers

in this chapter we will look in depth into Integers, Floats, and Number Strings.

PHP Numbers

One thing to notice about PHP is that it provides automatic data type conversion.

So, if you assign an integer value to a variable, the type of that variable will automatically be an integer. Then, if you assign a string to the same variable, the type will change to a string.

This automatic conversion can sometimes break your code.

PHP Numbers

PHP Integers

An integer is a number without any decimal part.

2, 256, -256, 10358, -179567 are all integers. While 7.56, 10.0, 150.67 are floats.

So, an integer data type is a non-decimal number between -2147483648 and 2147483647. A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.

Another important thing to know is that even if $4 * 2.5$ is 10, the result is stored as float, because one of the operands is a float (2.5).

PHP Numbers

Here are some 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 (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

PHP has the following functions to check if the type of a variable is integer:

`is_int()`

`is_integer()` - alias of `is_int()`

`is_long()` - alias of `is_int()`

Example

Check if the type of a variable is integer:

```
<?php
```

```
$x = 5985;
```

```
var_dump(is_int($x));
```

```
$x = 59.85;
```

```
var_dump(is_int($x));
```

```
?>
```

PHP Numbers

PHP Floats

A float is a number with a decimal point or a number in exponential form. 2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 are all floats.

The float data type can commonly store a value up to 1.7976931348623E+308 (platform dependent), and have a maximum precision of 14 digits.

PHP has the following functions to check if the type of a variable is float:

`is_float()`

`is_double()` - alias of `is_float()`

Example

Check if the type of a variable is float:

```
<?php
```

```
$x = 10.365;
```

```
var_dump(is_float($x));
```

```
?>
```

PHP Numbers

PHP Infinity

A numeric value that is larger than `PHP_FLOAT_MAX` is considered infinite. PHP has the following functions to check if a numeric value is finite or infinite:

[`is\_finite\(\)`](#)

[`is\_infinite\(\)`](#)

However, the PHP `var_dump()` function returns the data type and value:

Example

Check if a numeric value is finite or infinite:

```
<?php
```

```
$x = 1.9e411;
```

```
var_dump($x);
```

```
?>
```

PHP Numbers

PHP NaN

NaN stands for Not a Number.

NaN is used for impossible mathematical operations.

PHP has the following functions to check if a value is not a number:

[is_nan\(\)](#)

However, the PHP `var_dump()` function returns the data type and value:

Example

Invalid calculation will return a NaN value:

```
<?php
$x = acos(8);
var_dump($x);
?>
```

PHP Numbers

PHP Numerical Strings

The PHP `is_numeric()` function can be used to find whether a variable is numeric. The function returns true if the variable is a number or a numeric string, false otherwise.

Check if the variable is numeric:

```
<?php
$x = 5985;
var_dump(is_numeric($x));
$x = "5985";
var_dump(is_numeric($x));
$x = "59.85" + 100;
var_dump(is_numeric($x));
$x = "Hello";
var_dump(is_numeric($x));
?>
```

PHP Numbers

PHP Casting Strings and Floats to Integers

Sometimes you need to cast a numerical value into another data type.

The (int), (integer), or intval() function are often used to convert a value to an integer.

Cast float and string to integer:

```
<?php
// Cast float to int
$x = 23465.768;
$int_cast = (int)$x;
echo $int_cast;
echo "<br>";
// Cast string to int
$x = "23465.768";
$int_cast = (int)$x;
echo $int_cast;
?>
```

PHP Constants

Constants are like variables except that once they are defined they cannot be changed or undefined.

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

Create a PHP Constant

To create a constant, use the `define()` function.

Syntax

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

Parameters:

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

Example

Create a constant with a **case-sensitive** name:

```
<?php
```

```
define("GREETING", "Welcome to W3Schools.com!");
```

```
echo GREETING;
```

```
?>
```



PHP Constants

PHP Constant Arrays

In PHP7, you can create an Array constant using the define() function.

Create an Array constant:

```
<?php  
define("cars", [  
    "Alfa Romeo",  
    "BMW",  
    "Toyota"  
]);  
echo cars[0];  
?>
```

PHP Constants

Constants are Global

Constants are automatically global and can be used across the entire script.

This example uses a constant inside a function, even if it is defined outside the function:

```
<?php  
define("GREETING", "Welcome to W3Schools.com!");  
function myTest() {  
    echo GREETING;  
}  
myTest();  
?>
```

PHP if...else... elseif Statements

Conditional statements are used to perform different actions based on different conditions.

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following 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 Statements

PHP - The if Statement

The if statement executes some code if one condition is true.

Syntax

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

Example

Output "Have a good day!" if the current time (HOUR) is less than 20:

```
<?php
```

```
$t = date("H");
```

```
if ($t < "20") {  
    echo "Have a good day!";  
}  
?>
```

PHP if...else... elseif Statements

PHP - The if...else Statement

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;  
}
```

Example

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```
<?php  
$t = date("H");
```

```
if ($t < "20") {  
    echo "Have a good day!";  
} else {
```

PHP if...else... elseif Statements

PHP - The if...elseif...else Statement

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;  
}
```

Example

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```
<?php  
$t = date("H");
```

```
if ($t < "10") {  
    echo "Have a good morning!";  
}
```

PHP Switch Statement

The switch statement is used to perform different actions based on different conditions.

The PHP switch Statement

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 Switch Statement

This is how it works: First we have a single expression n (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use `break` to prevent the code from running into the next case automatically. The default statement is used if no match is found.

Example

```
<?php
```

```
$favcolor = "red";
```

```
switch ($favcolor) {
```

```
    case "red":
```

```
        echo "Your favorite color is red!";
```

```
        break;
```

```
    case "blue":
```

```
        echo "Your favorite color is blue!";
```

```
        break;
```

```
    case "green":
```

```
        echo "Your favorite color is green!";
```

PHP Loops

In the following chapters you will learn how to repeat code by using loops in PHP.

PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

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

The following chapters will explain and give examples of each loop type.

PHP Loops

The PHP while Loop

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;  
}
```

The example below displays the numbers from 1 to 5:

```
<?php  
$x = 1;  
while($x <= 5) {  
    echo "The number is: $x <br>";  
    $x++;  
}  
?>
```

PHP Loops

The PHP while Loop

This example counts to 100 by tens:

Example

```
<?php
$x = 0;
while($x <= 100) {
    echo "The number is: $x <br>";
    $x+=10;
}
?>
```

Example Explained

`$x = 0;` - Initialize the loop counter (`$x`), and set the start value to 0

`$x <= 100` - Continue the loop as long as `$x` is less than or equal to 100

`$x+=10;` - Increase the loop counter value by 10 for each iteration

PHP Loops

The PHP do...while Loop

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 Loops

The PHP do...while Loop

The example below first sets a variable \$x to 1 (\$x = 1). Then, the do while loop will write some output, and then increment the variable \$x with 1. Then the condition is checked (is \$x less than, or equal to 5?), and the loop will continue to run as long as \$x is less than, or equal to 5:

```
<?php  
$x = 1;  
do {  
    echo "The number is: $x <br>";  
    $x++;  
} while ($x <= 5);  
?>
```

Note: In a do...while loop the condition is tested AFTER executing the statements within the loop. This means that the do...while loop will execute its statements at least once, even if the condition is false.

PHP Loops

The PHP do...while Loop

This example sets the \$x variable to 6, then it runs the loop, **and then the condition is checked:**

```
<?php
```

```
$x = 6;
```

```
do {
```

```
    echo "The number is: $x <br>";
```

```
    $x++;
```

```
} while ($x <= 5);
```

```
?>
```

PHP Loops

The PHP for Loop

The **for** loop - Loops through a block of code a specified number of times. 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 for each iteration;  
}
```

Parameters:

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 Loops

The PHP for Loop

The example below displays the numbers from 0 to 10:

Example

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

Example Explained

`$x = 0;` - Initialize the loop counter (`$x`), and set the start value to 0

`$x <= 10;` - Continue the loop as long as `$x` is less than or equal to 10

`$x++` - Increase the loop counter value by 1 for each iteration

PHP Loops

The PHP for Loop

This example counts to 100 by tens:

```
<?php
for ($x = 0; $x <= 100; $x+=10) {
    echo "The number is: $x <br>";
}
?>
```

Example Explained

`$x = 0;` - Initialize the loop counter (`$x`), and set the start value to 0

`$x <= 100;` - Continue the loop as long as `$x` is less than or equal to 100

`$x+=10` - Increase the loop counter value by 10 for each iteration

PHP Loops

The PHP foreach Loop

The **foreach** loop - Loops through a block of code for each element in an array.

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 Loops

The PHP foreach Loop

The following example will output both the keys and the values of the given array (\$age):

Example

```
<?php
```

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

```
foreach($age as $x => $val) {
```

```
    echo "$x = $val<br>";
```

```
}
```

```
?>
```

You will learn more about arrays in the PHP Arrays chapter.

PHP Functions

The real power of PHP comes from its functions.
PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

Please check out our PHP reference for a complete overview of the [PHP built-in functions](#).

PHP Functions

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

- › A function is a block of statements that can be used repeatedly in a program.
- › A function will not execute automatically when a page loads.
- › A function will be executed by a call to the function.

Create a User Defined Function in PHP

A user-defined function declaration starts with the word function:

Syntax

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

PHP Functions

PHP User Defined Functions

Note: A function name must start with a letter or an underscore. Function names are NOT case-sensitive.

Tip: Give the function a name that reflects what the function does! In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code, and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name followed by brackets ():

```
<?php
function writeMsg() {
    echo "Hello world!";
}
writeMsg(); // call the function
?>
```



PHP Functions

PHP Function Arguments

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.

The following example has a function with one argument (\$fname). When the `familyName()` function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:



PHP Functions

PHP Function Arguments

```
<?php
```

```
function familyName($fname) {  
    echo "$fname Refsnes.<br>";  
}
```

```
familyName("Jani");  
familyName("Hege");  
familyName("Stale");  
familyName("Kai Jim");  
familyName("Borge");  
?>
```



PHP Functions

PHP Function Arguments

The following example has a function with two arguments (\$fname and \$year):

```
<?php  
function familyName($fname, $year) {  
    echo "$fname Refsnes. Born in $year <br>";  
}
```

```
familyName("Hege", "1975");  
familyName("Stale", "1978");  
familyName("Kai Jim", "1983");  
?>
```



PHP Functions

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

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 us an option to specify the expected data type when declaring a function, and by adding the strict declaration, it will throw a "Fatal Error" if the data type mismatches.

PHP Functions

PHP is a Loosely Typed Language

In the following example we try to send both a number and a string to the function without using strict:

```
<?php
function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it will
return 10
?>
```

PHP Functions

To specify **strict** we need to set **declare(strict_types=1);**. This must be on the very first line of the PHP file.

In the following example we try to send both a number and a string to the function, but here we have added the **strict** declaration:

```
<?php declare(strict_types=1); // strict requirement
```

```
function addNumbers(int $a, int $b) {  
    return $a + $b;  
}
```

```
echo addNumbers(5, "5 days");
```

```
// since strict is enabled and "5 days" is not an integer, an error will be  
thrown
```

```
?>
```

The **strict** declaration forces things to be used in the intended way.

PHP Functions

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function `setHeight()` without arguments it takes the default value as argument:

```
<?php declare(strict_types=1); // strict requirement
function setHeight(int $minheight = 50) {
    echo "The height is : $minheight <br>";
}
```

```
setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

PHP Functions

PHP Functions - Returning values

To let a function return a value, use the **return** statement:

```
<?php declare(strict_types=1); // strict requirement
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}
```

```
echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

PHP Functions

PHP Return Type Declarations

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. In the following example we specify the return type for the function:

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : float {
    return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

PHP Functions

PHP Return Type Declarations

You can specify a different return type, than the argument types, but make sure the return is the correct type:

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : int {
    return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
?>
```

PHP Functions

Passing Arguments by Reference

In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed. 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: Use a pass-by-reference argument to update a variable:

```
<?php
function add_five(&$value) {
    $value += 5;
}
$num = 2;
add_five($num);
echo $num;
?>
```

PHP Arrays

An array stores multiple values in one single variable:

What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
```

```
$cars2 = "BMW";
```

```
$cars3 = "Toyota";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is to create an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

PHP Arrays

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

PHP Indexed Arrays

There are 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";
```

The following example creates an indexed array named \$cars, assigns three elements to it, and then prints a text containing the array values:

```
<?php
```

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

```
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
```

```
?>
```

PHP Arrays

PHP Indexed Arrays

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

PHP Arrays

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are 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";
```

The named keys can then be used in a script:

```
<?php
```

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

```
echo "Peter is " . $age['Peter'] . " years old.";
```

```
?>
```

PHP Arrays

PHP Associative Arrays

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

```
<?php
```

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

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

PHP Arrays

PHP - Multidimensional Arrays

A multidimensional array is an array containing one or more arrays. PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

The dimension of an array indicates the number of indices you need to select an element.

- › For a two-dimensional array you need two indices to select an element
- › For a three-dimensional array you need three indices to select an element

PHP Arrays

PHP - Two-dimensional Arrays

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

First, take a look at the following table:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

PHP Arrays

PHP - Two-dimensional Arrays

We can store the data from the table above in a two-dimensional array, like this:

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

Now the two-dimensional \$cars array contains four arrays, and it has two indices: row and column.

PHP Arrays

PHP - Two-dimensional Arrays

To get access to the elements of the \$cars array we must point to the two indices (row and column):

Example

```
<?php
```

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

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

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

```
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2]."<br>;
```

```
?>
```

PHP Arrays

PHP - Two-dimensional Arrays

We can also put a for loop inside another for loop to get the elements of the \$cars array (we still have to point to the two indices):

```
<?php
for ($row = 0; $row < 4; $row++) {
    echo "<p><b>Row number $row</b></p>";
    echo "<ul>";
    for ($col = 0; $col < 3; $col++) {
        echo "<li>".$cars[$row][$col]."</li>";
    }
    echo "</ul>";
}
?>
```

PHP Arrays

Complete PHP Array Reference

For a complete reference of all array functions, go to our complete [PHP Array Reference](#).

The reference contains a brief description, and examples of use, for each function!

PHP RegEx

What is a **Regular Expression**?

A regular expression is 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.

A regular expression can be a single character, or a more complicated pattern.

Regular expressions can be used to perform all types of text search and text replace operations.

Syntax

In PHP, regular expressions are strings composed of delimiters, a pattern and optional modifiers.

PHP RegEx

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

In the example above, `/` is the **delimiter**, `w3schools` is the **pattern** that is being searched for, and `i` is a **modifier** that makes the search case-insensitive.

The delimiter can be any character that is not a letter, number, backslash or space. The most common delimiter is the forward slash (`/`), but when your pattern contains forward slashes it is convenient to choose other delimiters such as `#` or `~`.

PHP RegEx

Regular Expression Functions

PHP provides a variety of functions that allow you to use regular expressions.

The `preg_match()`, `preg_match_all()` and `preg_replace()` functions are some of the most commonly used ones:

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

PHP RegEx

Using preg_match()

The `preg_match()` function will tell you whether a string contains matches of a pattern.

Example

Use a regular expression to do a case-insensitive search for "w3schools" in a string:

```
<?php
$str = "Visit W3Schools";
$pattern = "/w3schools/i";
echo preg_match($pattern, $str);
?>
```

PHP RegEx

Using `preg_match_all()`

The `preg_match_all()` function will tell you how many matches were found for a pattern in a string.

Example

Use a regular expression to do a case-insensitive count of the number of occurrences of "ain" in a string:

```
<?php
```

```
$str = "The rain in SPAIN falls mainly on the plains.";
```

```
$pattern = "/ain/i";
```

```
echo preg_match_all($pattern, $str);
```

```
?>
```

PHP RegEx

Using preg_replace()

The `preg_replace()` function will replace all of the matches of the pattern in a string with another string.

Example

Use a case-insensitive regular expression to replace Microsoft with W3Schools in a string:

```
<?php
```

```
$str = "Visit Microsoft!";
```

```
$pattern = "/microsoft/i";
```

```
echo preg_replace($pattern, "W3Schools", $str);
```

```
?>
```

PHP RegEx

Regular Expression Modifiers

Modifiers can change how a search is performed.

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

PHP RegEx

Regular Expression Patterns

Brackets are used to find a range of characters:

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

Metacharacters

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

Quantifiers

Quantifiers define quantities:

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

PHP Forms

The PHP superglobals \$_GET and \$_POST are used to collect form-data.

PHP - A Simple HTML Form

The example below displays a simple HTML form with two input fields and a submit button:

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

PHP Forms

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
</body>
</html>
```

The output could be something like this:

Welcome John

Your email address is john.doe@example.com

PHP Forms

The same result could also be achieved using the HTTP GET method:

Example

```
<html>
```

```
<body>
```

```
<form action="welcome_get.php" method="get">
```

```
Name: <input type="text" name="name"><br>
```

```
E-mail: <input type="text" name="email"><br>
```

```
<input type="submit">
```

```
</form>
```

```
</body>
```

```
</html>
```

PHP Forms

and "welcome_get.php" looks like this:

```
<html>
<body>
```

```
Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>
```

```
</body>
</html>
```

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

PHP Forms

GET vs. POST

Both **GET** and **POST** create an array (e.g. `array( key1 => value1, key2 => value2, key3 => value3, ...)`). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both **GET** and **POST** are treated as `$_GET` and `$_POST`. These are 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.

`$_GET` is an array of variables passed to the current script via the URL parameters.

`$_POST` is an array of variables passed to the current script via the HTTP POST method.



PHP Forms

GET vs. POST

When to use **GET**?

Information sent from a form with the **GET** method is **visible to everyone** (all variable names and values are displayed in the URL). **GET** also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

PHP Forms

GET vs. POST

When to use **POST**?

Information sent from a form with the **POST** method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send. Moreover **POST** supports advanced functionality such as support for multi-part binary input while uploading files to server. However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Developers prefer POST for sending form data.

PHP Forms

PHP Form Validation

Think **SECURITY** when processing PHP forms!

These pages will show how to process PHP forms with security in mind. Proper validation of form data is important to protect your form from hackers and spammers!

The HTML form we will be working at in these chapters, contains various input fields: required and optional text fields, radio buttons, and a submit button:

PHP Form Validation Example

* required field

Name: *

E-mail: *

Website:

Comment:

Gender: ☐ Female ☐ Male ☐ Other *

Your Input:

PHP Forms

PHP Form Validation

The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

PHP Forms

PHP Form Validation

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

Name: `<input type="text" name="name">`

E-mail: `<input type="text" name="email">`

Website: `<input type="text" name="website">`

Comment: `<textarea name="comment" rows="5" cols="40"></textarea>`

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

Gender:

`<input type="radio" name="gender" value="female">`Female

`<input type="radio" name="gender" value="male">`Male

`<input type="radio" name="gender" value="other">`Other

PHP Forms

PHP Form Validation

The Form Element

The HTML code of the form looks like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

When the form is submitted, the form data is sent with method="post".

PHP Forms

PHP Form Validation

What is the `$_SERVER["PHP_SELF"]` variable?

The `$_SERVER["PHP_SELF"]` is a super global variable that returns the filename of the currently executing script.

So, the `$_SERVER["PHP_SELF"]` sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the `htmlspecialchars()` function?

The `htmlspecialchars()` function converts special characters to HTML entities. This means that it will replace HTML characters like `<` and `>` with `<` and `>`. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

PHP Forms

PHP Form Validation

Big Note on PHP Form Security

The `$_SERVER["PHP_SELF"]` variable can be used by hackers!
If `PHP_SELF` is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.



PHP Forms

PHP Form Validation

The next stValidate Form Data With PHP

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

```
<script>location.href('http://www.hacked.com')</script>
```

- this would not be executed, because it would be saved as HTML escaped code, like this:

```
&lt;script&gt;location.href('http://www.hacked.com')&lt;/script&gt;
```

The code is now safe to be displayed on a page or inside an e-mail.



PHP Forms

PHP Form Validation

We will also do two more things when the user submits the form:
Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP `trim()` function)

Remove backslashes (`\`) from the user input data (with the PHP `stripslashes()` function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function `test_input()`.

PHP Forms

Now, we can check each `$_POST` variable with the `test_input()` function:s:

```
<?php
```

```
// define variables and set to empty values
```

```
$name = $email = $gender = $comment = $website = "";
```

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

```
    $name = test_input($_POST["name"]);
```

```
    $email = test_input($_POST["email"]);
```

```
    $website = test_input($_POST["website"]);
```

```
    $comment = test_input($_POST["comment"]);
```

```
    $gender = test_input($_POST["gender"]);
```

```
}
```

```
function test_input($data) {
```

```
    $data = trim($data);
```

```
    $data = stripslashes($data);
```

```
    $data = htmlspecialchars($data);
```

```
    return $data;
```

```
}
```

```
?>
```

PHP Forms

PHP - Required Fields

From the validation rules table on the previous page, we see that the "Name", "E-mail", and "Gender" fields are required. These fields cannot be empty and must be filled out in the HTML form.

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

PHP Forms

PHP - Required Fields

In the previous chapter, all input fields were optional.

In the following code we have added some new variables: `$nameErr`, `$emailErr`, `$genderErr`, and `$websiteErr`. These error variables will hold error messages for the required fields. We have also added an if else statement for each `$_POST` variable. This checks if the `$_POST` variable is empty (with the PHP `empty()` function). If it is empty, an error message is stored in the different error variables, and if it is not empty, it sends the user input data through the `test_input()` function:

PHP Forms

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = test_input($_POST["name"]);
    }
}
```

```
if (empty($_POST["email"])) {
    $emailErr = "Email is required";
} else {
    $email = test_input($_POST["email"]);
}
```

PHP Forms

```
if (empty($_POST["website"])) {  
    $website = "";  
} else {  
    $website = test_input($_POST["website"]);  
}  
if (empty($_POST["comment"])) {  
    $comment = "";  
} else {  
    $comment = test_input($_POST["comment"]);  
}  
if (empty($_POST["gender"])) {  
    $genderErr = "Gender is required";  
} else {  
    $gender = test_input($_POST["gender"]);  
}  
}  
?>
```

PHP Forms

PHP - Display The Error Messages

Then in the HTML form, we add a little script after each required field, which generates the correct error message if needed (that is if the user tries to submit the form without filling out the required fields):

To See the example: [Click here](#)

PHP Forms

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters, dashes, apostrophes and whitespaces. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);  
if (!preg_match("/^[a-zA-Z-' ]*$/", $name)) {  
    $nameErr = "Only letters and white space allowed";  
}
```

The [preg_match\(\)](#) function searches a string for pattern, returning true if the pattern exists, and false otherwise.

PHP Forms

PHP - Validate E-mail

The easiest and safest way to check whether an email address is well-formed is to use PHP's `filter_var()` function.

In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);  
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {  
    $emailErr = "Invalid email format";  
}
```

PHP Forms

PHP - Validate URL

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

```
$website = test_input($_POST["website"]);  
if (!preg_match("/\b(?:(:https?|ftp):\/\/|www\.|)[-a-z0-9+&@#\%?=~_|!:,.;]*[-a-z0-9+&@#\%?=~_|]/i",$website)) {  
    $websiteErr = "Invalid URL";  
}
```

The script will look like this: [Click here](#)

MySQL Database

With PHP, you can connect to and manipulate databases.
MySQL is the most popular database system used with PHP.

What is MySQL?

MySQL is a database system used on the web

MySQL is a database system that runs on a server

MySQL is ideal for both small and large applications

MySQL is very fast, reliable, and easy to use

MySQL uses standard SQL

MySQL compiles on a number of platforms

MySQL is free to download and use

MySQL is developed, distributed, and supported by Oracle Corporation

MySQL is named after co-founder Monty Widenius's daughter: My

MySQL Database

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Databases are useful for storing information categorically. A company may have a database with the following tables:

- Employees
- Products
- Customers
- Orders

PHP + MySQL Database System

PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Database Queries: A query is a question or a request.

We can query a database for specific information and have a recordset returned.

MySQL Database

Database Queries

A query is a question or a request.

We can query a database for specific information and have a recordset returned.

Look at the following query (using standard SQL):

```
SELECT LastName FROM Employees
```

The query above selects all the data in the "LastName" column from the "Employees" table.

Download MySQL Database

If you don't have a PHP server with a MySQL Database, you can download it for free here: <http://www.mysql.com>

Facts About MySQL Database

MySQL is the de-facto standard database system for web sites with HUGE volumes of both data and end-users (like Facebook, Twitter, and Wikipedia).

MySQL Database

PHP Connect to MySQL

PHP 5 and later can work with a MySQL database using:

MySQLi extension (the "i" stands for improved)

PDO (PHP Data Objects)

Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

MySQL Database

Should I Use MySQLi or PDO?

If you need a short answer, it would be "Whatever you like".

Both MySQLi and PDO have their advantages:

PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.

So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.

Both are object-oriented, but MySQLi also offers a procedural API.

Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

MySQL Database

MySQLi Installation

For Linux and Windows: The MySQLi extension is automatically installed in most cases, when php5 mysql package is installed.

For installation details, go

to: <http://php.net/manual/en/mysqli.installation.php>

PDO Installation

For installation details, go

to: <http://php.net/manual/en/pdo.installation.php>

MySQL Database

Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

Example (MySQLi Object-Oriented)

```
<?php
```

```
$servername = "localhost";
```

```
$username = "username";
```

```
$password = "password";
```

```
// Create connection
```

```
$conn = new mysqli($servername, $username, $password);
```

```
// Check connection
```

```
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
```

```
}
```

MySQL Database

Note on the object-oriented example above:

`$connect_error` was broken until PHP 5.2.9 and 5.3.0. If you need to ensure compatibility with PHP versions prior to 5.2.9 and 5.3.0, use the following code instead:

```
// Check connection
if (mysqli_connect_error()) {
    die("Database connection failed: " . mysqli_connect_error());
}
```

MySQL Database

Example (MySQLi Procedural)

```
<?php
```

```
$servername = "localhost";
```

```
$username = "username";
```

```
$password = "password";
```

```
// Create connection
```

```
$conn = mysqli_connect($servername, $username, $password);
```

```
// Check connection
```

```
if (!$conn) {
```

```
    die("Connection failed: " . mysqli_connect_error());
```

```
}
```

```
echo "Connected successfully";
```

```
?>
```

MySQL Database

Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB",
$username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
} catch(PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>
```

MySQL Database

Note: In the PDO example above we have also **specified a database (myDB)**. PDO require a valid database to connect to. If no database is specified, an exception is thrown.

Tip: A great benefit of PDO is that it has an exception class to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block.

MySQL Database

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

MySQLi Object-Oriented:

```
$conn->close();
```

MySQLi Procedural:

```
mysqli_close($conn);
```

PDO:

```
$conn = null;
```

PHP XML

What is XML?

The XML language is a way to structure data for sharing across websites. Several web technologies like RSS Feeds and Podcasts are written in XML. XML is easy to create. It looks a lot like HTML, except that you make up your own tags.

If you want to learn more about XML, please visit our [XML tutorial](#). To read and update, create and manipulate an XML document, you will need an XML parser.

In PHP there are two major types of XML parsers:

- › Tree-Based Parsers
- › Event-Based Parsers

PHP XML

Tree-Based Parsers

Tree-based parsers holds the entire document in Memory and transforms the XML document into a Tree structure. It analyzes the whole document, and provides access to the Tree elements (DOM).

This type of parser is a better option for smaller XML documents, but not for large XML document as it causes major performance issues.

Example of tree-based parsers:

- › SimpleXML
- › DOM

PHP XML

Event-Based Parsers

Event-based parsers do not hold the entire document in Memory, instead, they read in one node at a time and allow you to interact with in real time. Once you move onto the next node, the old one is thrown away.

This type of parser is well suited for large XML documents. It parses faster and consumes less memory.

Example of event-based parsers:

- XMLReader
- XML Expat Parser

PHP XML

The SimpleXML Parser

SimpleXML is a tree-based parser.

SimpleXML provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout.

SimpleXML turns an XML document into a data structure you can iterate through like a collection of arrays and objects.

Compared to DOM or the Expat parser, SimpleXML takes a fewer lines of code to read text data from an element.

Installation

From PHP 5, the SimpleXML functions are part of the PHP core. No installation is required to use these functions.

PHP XML

PHP SimpleXML - Read From String

The PHP `simplexml_load_string()` function is used to read XML data from a string.

Assume we have a variable that contains XML data, like this:

```
$myXMLData =  
"<?xml version='1.0' encoding='UTF-8'?>  
<note>  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend!</body>  
</note>";
```

PHP XML

The example below shows how to use the `simplexml_load_string()` function to read XML data from a string:

```
<?php
$xmlData =
"<?xml version='1.0' encoding='UTF-8'?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>";
$xml=simplexml_load_string($xmlData) or die("Error: Cannot create
object");
print_r($xml);
?>
```

[Run The example](#)

PHP XML

PHP SimpleXML - Get Node/Attribute Values

SimpleXML is a PHP extension that allows us to easily manipulate and get XML data.

PHP SimpleXML - Get Node Values

Get the node values from the "[note.xml](#)" file:

Example

```
<?php
```

```
$xml=simplexml_load_file("note.xml") or die("Error: Cannot create object");
```

```
echo $xml->to . "<br>";
```

```
echo $xml->from . "<br>";
```

```
echo $xml->heading . "<br>";
```

```
echo $xml->body;
```

```
?>
```

Run The example

PHP XML

PHP SimpleXML - Get Node Values of Specific Elements

The following example gets the node value of the <title> element in the first and second <book> elements in the ["books.xml"](#) file:

Example

```
<?php
$xml=simplexml_load_file("books.xml") or die("Error: Cannot create
object");
echo $xml->book[0]->title . "<br>";
echo $xml->book[1]->title;
?>
```

Run The example

PHP XML

PHP SimpleXML - Get Node Values - Loop

The following example loops through all the <book> elements in the "books.xml" file, and gets the node values of the <title>, <author>, <year>, and <price> elements:

Example

```
<?php
$xml=simplexml_load_file("books.xml") or die("Error: Cannot create
object");
foreach($xml->children() as $books) {
    echo $books->title . ", ";
    echo $books->author . ", ";
    echo $books->year . ", ";
    echo $books->price . "<br>";
}
?>
```

Run The example

PHP XML

PHP SimpleXML - Get Attribute Values

The following example gets the attribute value of the "category" attribute of the first <book> element and the attribute value of the "lang" attribute of the <title> element in the second <book> element:

Example

```
<?php
$xml=simplexml_load_file("books.xml") or die("Error: Cannot create
object");
echo $xml->book[0]['category'] . "<br>";
echo $xml->book[1]->title['lang'];
?>
```

Run The example

PHP XML

PHP SimpleXML - Get Attribute Values – Loop

The following example gets the attribute values of the <title> elements in the "books.xml" file:

```
<?php
$xml=simplexml_load_file("books.xml") or die("Error: Cannot create
object");
foreach($xml->children() as $books) {
    echo $books->title['lang'];
    echo "<br>";
}
?>
```

Run The example

More PHP SimpleXML

For more information about the PHP SimpleXML functions, visit our [PHP SimpleXML Reference](#).



Lebanese University
Faculty of Information

Brunch II

Dr. Mohamad **Nagi**

PhD in Data Mining - Computer Science

Thank you for Listening