

PHP

Introduction - How web works - Setting up the environment (LAMP server) - Programming basics Print/echo - Variables and constants – Strings and Arrays – Operators, Control structures and looping structures – Functions – Reading Data in Web Pages - Embedding PHP within HTML - Establishing connectivity with MySQL database.

Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Introduction

- PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.
 - PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
 - PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
 - It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
 - PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
 - PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
 - PHP is forgiving: PHP language tries to be as forgiving as possible.
 - PHP Syntax is C-Like.

Characteristics of PHP

- Five important characteristics make PHP's practical nature possible –
 - Simplicity
 - Efficiency
 - Security
 - Flexibility
 - Familiarity

basic structure of PHP program

- PHP script always starts with `<?php` and ends with `?>`.
- PHP script can be placed anywhere in the document.
- In some server you can use the above script as `<?` and ends with `?>` but it does not support by all servers.
- Normally PHP file contains HTML tags and PHP scripts.
- PHP file must be stored in .php file extension
- Each PHP code must end with semicolon(;

```
<?php
function primeCheck($number){
    if ($number == 1)
        return 0;
    for ($i = 2; $i <= $number/2; $i++){
        if ($number % $i == 0)
            return 0;
    }
    return 1;
}
$number = 31;
$flag = primeCheck($number);
if ($flag == 1)
    echo "Prime";
else
    echo "Not Prime"
?>
```

variables

- All variables in PHP are denoted with a leading dollar sign (\$).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.
- PHP variables are Perl-like.

uses of PHP

- Server-Side Scripting. In PHP server-side scripting is the main area of operation of PHP scripts
- Command-Line Scripting
- Desktop Application Development
- Open Source
- Cross Platform
- PHP is a general-purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server.
- Any PHP code in a requested file is executed by the PHP runtime, usually to create **dynamic** web page content or **dynamic** images used on websites or elsewhere.

Variable Types

- **Integers** – are whole numbers, without a decimal point, like 4195.
- **Doubles** – are floating-point numbers, like 3.14159 or 49.1.
- **Booleans** – have only two possible values either true or false.
- **NULL** – is a special type that only has one value: NULL.
- **Strings** – are sequences of characters, like 'PHP supports string operations.'
- **Arrays** – are named and indexed collections of other values.
- **Objects** – are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** – are special variables that hold references to resources external to PHP (such as database connections).

Strings

- Example :

```
<?php
$variable = "name";
$literally = 'My $variable will not print!';
print($literally);
print "<br>";
$literally = "My $variable will print!";
print($literally);
?>
```
- This will produce following result –
- My \$variable will not print! My name will print There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.
- Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP –
- Certain character sequences beginning with backslash (\) are replaced with special characters

Strings

- they are sequences of characters, like "PHP supports string operations". Following are valid examples of string
- \$string_1 = "This is a string in double quotes";
- \$string_2 = 'This is a somewhat longer, singly quoted string';
- \$string_39 = "This string has thirty-nine characters";
- \$string_0 = ""; // a string with zero characters
- Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

Strings

- Variable names (starting with \$) are replaced with string representations of their values.
- The escape-sequence replacements are –
 - \n is replaced by the newline character
 - \r is replaced by the carriage-return character
 - \t is replaced by the tab character
 - \\$ is replaced by the dollar sign itself (\$)
 - \" is replaced by a single double-quote (")
 - \\ is replaced by a single backslash (\)

Constant

- A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default, a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. If you have defined a constant, it can never be changed or undefined.
- To define a constant you have to use `define()` function and to retrieve the value of a constant, you have to simply specifying its name. Unlike with variables, you do not need to have a constant with a `$`. You can also use the function `constant()` to read a constant's value if you wish to obtain the constant's name dynamically.

Operators

- Simple answer can be given using expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator.
- PHP language supports following type of operators.
 - Arithmetic Operators
 - Comparison Operators
 - Logical (or Relational) Operators
 - Assignment Operators
 - Conditional (or ternary) Operators

Constant

- `constant()` function
 - As indicated by the name, this function will return the value of the constant.
 - This is useful when you want to retrieve value of a constant, but you do not know its name, i.e. It is stored in a variable or returned by a function.
- Differences between constants and variables are
 - There is no need to write a dollar sign (\$) before a constant, where as in Variable one has to write a dollar sign.
 - Constants cannot be defined by simple assignment, they may only be defined using the `define()` function.
 - Constants may be defined and accessed anywhere without regard to variable scoping rules.
 - Once the Constants have been set, may not be redefined or undefined.

Control Structure

- **if...else statement** – use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** – is used with the if...else statement to execute a set of code if **one** of the several condition is true
- **switch statement** – is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

Looping

- Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.
 - **for** – loops through a block of code a specified number of times.
 - **while** – loops through a block of code if and as long as a specified condition is true.
 - **do...while** – loops through a block of code once, and then repeats the loop as long as a special condition is true.
 - **foreach** – loops through a block of code for each element in an array.

PHP Superglobals

4) \$_FILES["FormElementName"]

It can be used to upload files from a client computer/system to a server.

\$_FILES["FormElementName"]["ArrayIndex"]

Such as File Name, File Type, File Size, File temporary name.

5) \$_SESSION["VariableName"]

A session variable is used to store information about a single user, and are available to all pages within one application.

6) \$_COOKIE["VariableName"]

A cookie is used to identify a user. cookie is a small file that the server embedded on user computer.

7) \$_SERVER["ConstantName"]

\$_SERVER holds information about headers, paths, and script locations.

eg.

\$_SERVER["SERVER_PORT"], \$_SERVER["SERVER_NAME"], \$_SERVER["REQUEST_URI"]

Reading Data in Web Pages - PHP Superglobals

- PHP super global variable is used to access global variables from anywhere in the PHP script.
- PHP Super global variables is accessible inside the same page that defines it, as well as outside the page.
- while local variable's scope is within the page that defines it.
- The PHP super global variables are :
 - 1) \$_GET["FormElementName"]

It is used to collect value from a form(HTML script) sent with method='get' information sent from a form with the method='get' is visible to everyone(it display on the browser URL bar).
 - 2) \$_POST["FormElementName"]

It is used to collect value in a form with method="post". Information sent from a form is invisible to others.(can check on address bar)
 - 3) \$_REQUEST["FormElementName"]

This can be used to collect data with both post and get method.

PHP Superglobals

- Several predefined variables in PHP 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.
- The PHP superglobal variables are:
 - \$GLOBALS
 - \$_SERVER
 - \$_REQUEST
 - \$_POST
 - \$_GET
 - \$_FILES
 - \$_ENV
 - \$_COOKIE
 - \$_SESSION

Example for \$GLOBALS

```
<?php
    $x = 75;
    $y = 25;
    function addition() {
        $GLOBALS['z'] = $GLOBALS['x']
        + $GLOBALS['y'];
    }
    addition();
    echo $z;
?>
```

Example for \$_SERVER

```
<?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'];
?>
```

Example for \$_REQUEST

```
<html>
<body>
<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;
    }
}
?>
</body>
</html>
```

Example for \$_POST

```
<html>
<body>
<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;
    }
}
?>
</body>
</html>
```

Example for \$_GET

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

use of die()

it is used to display a message and exit the script. It may be used to print alternate message . Instead of showing error it will show the user friendly message. die() function is only used to print string messages .value of variables cannot print with die() function.

```
<?php
    die("this is Php die example");
    //this will print the message
    $x="hello test";
    die($x)
    //this will show blank screen.
?>
```

Arrays in PHP

- An array stores multiple values in one single variable.

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . " , " . $cars[1] . " and " . $cars[2] . " .";
?>
```
- In PHP, the array() function is used to create an array: 3 types of array
 - **Indexed arrays** - Arrays with a numeric index
 - **Associative arrays** - Arrays with named keys
 - **Multidimensional arrays** - Arrays containing one or more arrays

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";

Example:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . " , " . $cars[1] . " and " . $cars[2]
. " .";
?>
```

Indexed arrays

- Get The Length of an Array - The count() Function
 - The count() function is used to return the length (the number of elements) of an array:

```
<?php
$scars = array("Volvo", "BMW", "Toyota");
echo count($scars);
?>
```
- 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
$scars = array("Volvo", "BMW", "Toyota");
$scarslength = count($scars);
for($x = 0; $x < $scarslength; $x++) {
    echo $scars[$x];
    echo "<br>";
}
?>
```

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>";
}
?>
```

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");
 - \$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.";
?>

Multidimensional arrays

- A multidimensional array is an array containing one or more arrays.
- PHP understands 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 - Two-dimensional Arrays : A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).
 - \$scars = array
(
 array("Volvo",22,18),
 array("BMW",15,13),
 array("Saab",5,2),
 array("Land Rover",17,15)
);

Multidimensional arrays

- Now the two-dimensional \$cars array contains four arrays, and it has two indices: row and column.
- To get access to the elements of the \$cars array we must point to the two indices (row and column):

```
<?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>";
?>
```
- 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>";
}
?>
```

Example

```
<?php
/* Defining a PHP Function */
function writeMessage() {
    echo "You are really a nice person, Have a nice time!";
}

/* Calling a PHP Function */
writeMessage();
?>
```

Functions

- PHP functions are similar to other programming languages.
- A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.
- There are two parts which should be clear to you –
 - Creating a PHP Function
 - Calling a PHP Function

Adding 2 Numbers

```
<?php
function addFunction($num1, $num2) {
    $sum = $num1 + $num2;
    echo "Sum of the two numbers is : $sum";
}

addFunction(10, 20);
?>
```

PHP Functions returning value

```
<?php
    function addFunction($num1, $num2) { $sum
= $num1 + $num2;
    return $sum;
}
$return_value = addFunction(10, 20);
echo "Returned value from the function :
$return_value";
?>
```

Factorial of the number 'n' using recursive function

```
<?php
// PHP code to get the factorial of a number
// function to get factorial in iterative way
function Factorial($number){
    if($number <= 1){
        return 1;
    }
    else{
        return $number * Factorial($number - 1);
    }
}
// Driver Code
$number = 10;
$fact = Factorial($number);
echo "Factorial = $fact";
?>
```

Fibonacci series of 'n' terms

```
<?php
// PHP code to get the Fibonacci series
// Recursive function for fibonacci series.
function Fibonacci($number){
    // if and else if to generate first two numbers
    if ($number == 0)
        return 0;
    else if ($number == 1)
        return 1;
    // Recursive Call to get the upcoming numbers
    else
        return (Fibonacci($number-1) + Fibonacci($number-2));
}
// Driver Code
$number = 10;
for ($counter = 0; $counter < $number; $counter++){
    echo Fibonacci($counter), ' ';
}
}
```

Palindrome on a given number

```
<?php
// PHP code to check for Palindrome number in PHP
// Function to check for Palindrome
function Palindrome($number){
    $temp = $number;
    $new = 0;
    while (floor($temp)) {
        $d = $temp % 10;
        $new = $new * 10 + $d;
        $temp = $temp/10;
    }
    if ($new == $number){
        return 1;
    }
    else{
        return 0;
    }
}
// Driver Code
$original = 1441;
if (Palindrome($original)){
    echo "Palindrome";
}
else {
    echo "Not a Palindrome";
}
?>
```

Palindrome on a given string

```
<?php
// PHP code to check for Palindrome string in PHP
// Using strrev()
function Palindrome($string){
    if (strrev($string) == $string){
        return 1;
    }
    else{
        return 0;
    }
}
// Driver Code
$original = "DAD";
if(Palindrome($original)){
    echo "Palindrome";
}
else {
    echo "Not a Palindrome";
}
?>
```

Create connection

It is very important when you want to start a dynamic website to create connection with your SQL (we are talking about mysql) by using `mysql_connect()` function. In `mysql_connect()` arguments should be string and it's important to use `die()` function with `mysql_error()` function to check if there is a problem with the connection or not.

Connectivity with MySQL database

- Create connection
- Select database
- Perform database query
- Use return data
- Close connection

Select database

Now we have to select the database which we are creating and saving our data by using `mysql_select_db()` function which the arguments are the name of database and connection we made earlier.

Perform database query

In this step we have to select out data from database and bring it into our web page. The best decision to use `mysql_query()` function which it will Send a MySQL query with 2 arguments as displayed in the code.

Use return data

To display the result on your web page, you have to use `while()` loop function in addition to `mysql_fetch_array()` function which fetch a result row as an associative array, a numeric array, or both.

Close connection

To close connection, it is better to use `mysql_close()` function to close MySQL connection. This is a very important function as it closes the connection to the database server. Your script will still run if you do not include this function. And too many open MySQL connections can cause problems for your account. This it is a good practice to close the MySQL connection once all the queries are executed.

Five steps to PHP database connections:

```
<?php
// 1. Create a database connection
// $connection allows us to keep referring to this connection after it is established
$connection = mysql_connect("localhost","root","myPassword");
if (!$connection) {
    die("Database connection failed: " . mysql_error());
}
$db_select = mysql_select_db("widget_corp",$connection); // 2. Select a database to use
if (!$db_select) {
    die("Database selection failed: " . mysql_error());
}
$result = mysql_query("SELECT * FROM subjects", $connection); // 3. Perform database query
if (!$result) {
    die("Database query failed: " . mysql_error());
}
while ($row = mysql_fetch_array($result)) { // 4. Use returned data
    echo $row["menu_name"]." ".$row["position"]."<br />";
}
connectionmysql_close($connection); // 5. Close
?>
```

HOW WEB WORKS

- In order to develop and run PHP Web pages three vital components need to be installed on your computer system.
- **Web Server** – PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server. Download Apache for free here
– <https://httpd.apache.org/download.cgi>
- **Database** – PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database. Download MySQL for free here
– <https://www.mysql.com/downloads/>
- **PHP Parser** – In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.

HOW WEB WORKS

- **Apache Configuration**
 - If you are using Apache as a Web Server then this section will guide you to edit Apache Configuration Files.
- **PHP.INI File Configuration**
 - The PHP configuration file, php.ini, is the final and most immediate way to affect PHP's functionality.
- **Windows IIS Configuration**
 - To configure IIS on your Windows machine you can refer your IIS Reference Manual shipped along with IIS.

HOW WEB WORKS

PHP Parser Installation

- Before you proceed it is important to make sure that you have proper environment setup on your machine to develop your web programs using PHP.
- Type the following address into your browser's address box.
 - <http://127.0.0.1/info.php>
If this displays a page showing your PHP installation related information then it means you have PHP and Webserver installed properly. Otherwise you have to follow given procedure to install PHP on your computer.
- This section will guide you to install and configure PHP over the following four platforms –
 - [PHP Installation on Linux or Unix with Apache](#)
 - [PHP Installation on Mac OS X with Apache](#)
 - [PHP Installation on Windows NT/2000/XP with IIS](#)
 - [PHP Installation on Windows NT/2000/XP with Apache](#)

Setting up the environment (LAMP server)

- **Step 1: Install Apache**
 - open terminal and type in these commands:
 - `sudo apt-get update`
 - `sudo apt-get install apache2`
 - That's it. To check if Apache is installed, direct your browser to your server's IP address (eg. `http://12.34.56.789`).
 - **How to Find your Server's IP address**
 - You can run the following command to reveal your server's IP address.
 - `ifconfig eth0 | grep inet | awk '{ print $2 }'`
- **Step 2: Install MySQL**
 - MySQL is a powerful database management system used for organizing and retrieving data. To install MySQL, open terminal and type in these commands:
 - `sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql`
 - During the installation, MySQL will ask you to set a root password. If you miss the chance to set the password while the program is installing, it is very easy to set the password later from within the MySQL shell. Once you have installed MySQL, we should activate it with this command:
 - `sudo mysql_install_db`

Setting up the environment (LAMP server)

Finish up by running the MySQL set up script:

- `sudo /usr/bin/mysql_secure_installation`
- The prompt will ask you for your current root password.
- Type it in.
- Enter current password for root (enter for none):
 - OK, successfully used password, moving on...
- Then the prompt will ask you if you want to change the root password. Go ahead and choose and move on to the next steps.

Setting up the environment (LAMP server)

- Remove test database and access to it? [Y/n] y
 - Dropping test database...
... Success!
- - Removing privileges on test database...
... Success!
- Reloading the privilege tables will ensure that all changes made so far will take effect immediately.
Reload privilege tables now? [Y/n] y
... Success!
- Cleaning up...

Setting up the environment (LAMP server)

- It's easiest just to say yes to all the options. At the end, MySQL will reload and implement the new changes.
- By default, a MySQL installation has an anonymous user, allowing anyone to log into MySQL without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.
- Remove anonymous users? [Y/n] y
 - ... Success!
- Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.
- Disallow root login remotely? [Y/n] y
 - ... Success!
- By default, MySQL comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.
- Once you're done with that you can finish up by installing PHP.

Setting up the environment (LAMP server)

• **Step 3: Install PHP**

- To install PHP, open terminal and type in this command.
 - `sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt`
- After you answer yes to the prompt twice, PHP will install itself. It may also be useful to add php to the directory index, to serve the relevant php index files:
 - `sudo nano /etc/apache2/mods-enabled/dir.conf`
- Add index.php to the beginning of index files. The page should now look like this:

```
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl
    index.php index.xhtml index.htm
</IfModule>
```

Setting up the environment (LAMP server)

• PHP Modules

- PHP also has a variety of useful libraries and modules that you can add onto your virtual server. You can see the libraries that are available.
- apt-cache search php5-
- Terminal will then display the list of possible modules. The beginning looks like this:
 - php5-cgi - server-side, HTML-embedded scripting language (CGI binary)
 - php5-cli - command-line interpreter for the php5 scripting language
 - php5-common - Common files for packages built from the php5 source
 - php5-curl - CURL module for php5
 - php5-dbg - Debug symbols for PHP5
 - php5-dev - Files for PHP5 module development
 - php5-gd - GD module for php5
 - php5-gmp - GMP module for php5
 - php5-ldap - LDAP module for php5
 - php5-mysql - MySQL module for php5
 - php5-odbc - ODBC module for php5
 - php5-pgsql - PostgreSQL module for php5

Setting up the environment (LAMP server)

• Step 4: RESULTS — See PHP on your Server

- Although LAMP is installed, we can still take a look and see the components online by creating a quick php info page
- To set this up, first create a new file:
 - `sudo nano /var/www/info.php`
- Add in the following line:

```
<?php
phpinfo();
?>
```


Then Save and Exit.
- Restart apache so that all of the changes take effect:
 - `sudo service apache2 restart`
- Finish up by visiting your php info page (make sure you replace the example ip address with your correct one):
`http://12.34.56.789/info.php`

Setting up the environment (LAMP server)

php5-pspell - pspell module for php5
php5-recode - recode module for php5
php5-snmp - SNMP module for php5
php5-sqlite - SQLite module for php5
php5-tidy - tidy module for php5
php5-xmllrpc - XML-RPC module for php5
php5-xsl - XSL module for php5
php5-adodb - Extension optimising the ADOdb database abstraction library
php5-auth-pam - A PHP5 extension for PAM authentication
[...]

- Once you decide to install the module, type:
 - `sudo apt-get install name of the module`
 - You can install multiple libraries at once by separating the name of each module with a space.
- Congratulations! You now have LAMP stack on your droplet!

Setting up the environment (LAMP server)

PHP Version 5.3.10-1ubuntu3.2	
	
System	Linux x3.2.0-24-virtual #37-Ubuntu SMP Wed Apr 25 12:51:49 UTC 2012 i686
Build Date	Jun 13 2012 17:02:41
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d/pdo_mysql.ini
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626.NTS
PHP Extension Build	API220090626.NTS
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sshv3, tls
Registered Stream Filters	zlib*, bzip2*, convertionv*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert*, consumed, dechunk
This server is protected with the Suhosin Patch 0.9.10 Copyright (c) 2006-2007 Hardened-PHP Project Copyright (c) 2007-2009 Section8 GmbH	
This program makes use of the Zend Scripting Language Engine: Zend Engine v2.3.0, Copyright (c) 1998-2012 Zend Technologies	
