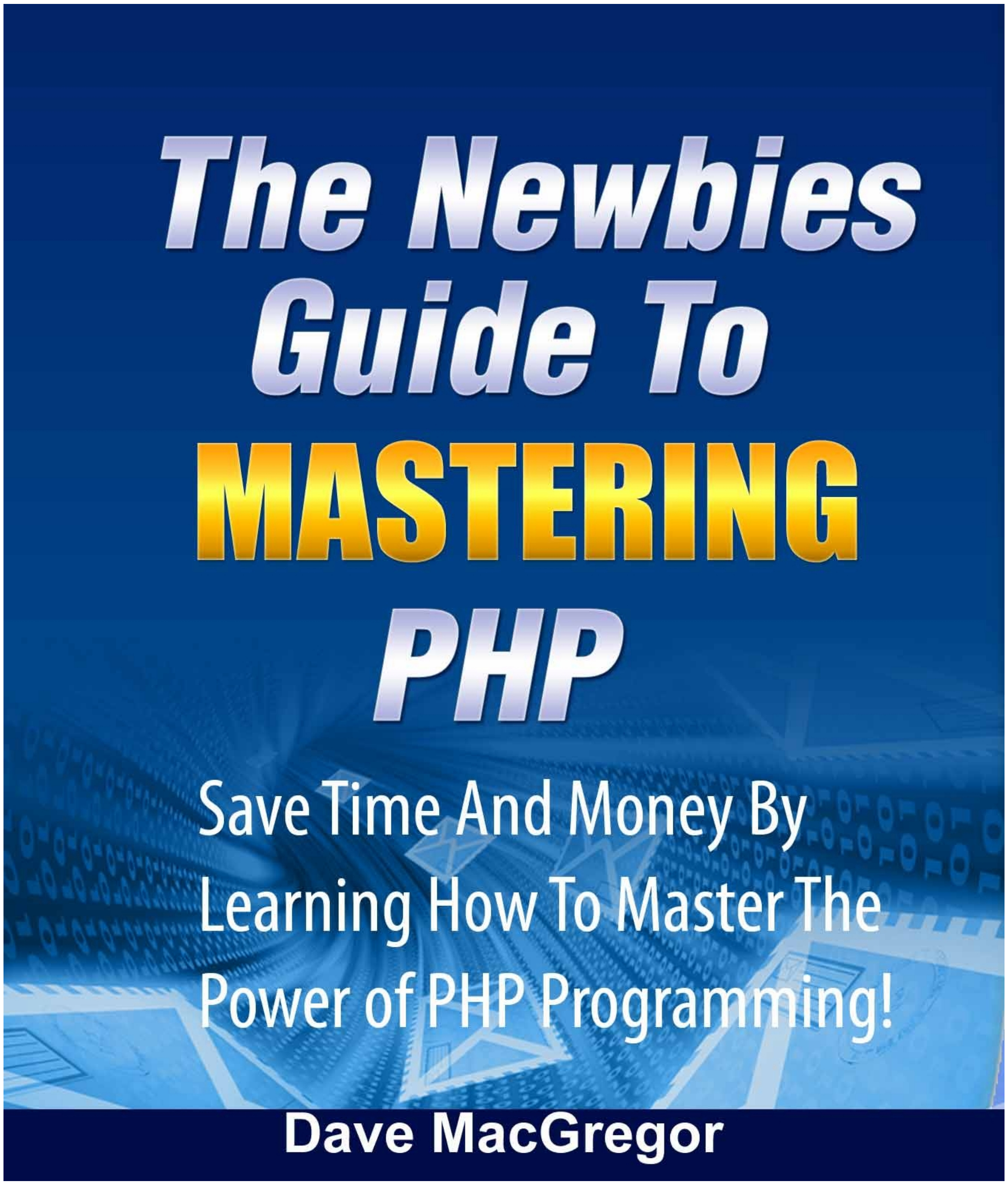

The Newbies Guide To
MASTERING
PHP
Save Time And Money By Learning How To Master The Power of PHP Programming!
Dave MacGregor

LEGAL NOTICE

First Kindle Original Edition, April 2008

Copyright 2008 by Dave MacGregor

Published by Avonside Publishing

Welcome to the world of programming!

First of all, I would like to congratulate you on your decision to learn one of the easiest yet powerful programming languages available – PHP.

Web Developers use PHP to create interactive, dynamic websites while making it easy to update and maintain, not to mention, prompt your visitors to purchase, join a list or even tell a friend.

What is PHP And Why Should I Learn It?

PHP has been around for over 12 years, first coming onto the scene in 1995 by a freelance software developer, Rasmus Lerdorf. He initially used PHP by creating a script that retained a log of all of the visitors that landed on his webpage.

His script would feature information on how many visits his site received and other detailed information about his website stats.

As you can imagine, It didn't take long before he started to receive emails and messages asking how he was able to do this, and because of the overwhelming interest in his scripts, he worked to develop PHP into what is now one of the most popular and widely used programming languages on the Internet.

Throughout the years, I have seen a number of webmasters ask the question as to why they should learn PHP. In truth, there are so many benefits to learning how to implement the power of PHP into your websites that a list of potential uses would take up this ebook alone. From managing pages and updating websites quickly, to creating commercial scripts for profit, the sky is truly the limit.

Think of all your favorite websites, WordPress, Paypal, eBay or perhaps FaceBook.

They all use the power of PHP to make their website run smoothly and efficiently.

Getting Started

First of all, you will need to have access to a web server that is set up to work with PHP. Don't let this scare you. Typically, most of the web hosting companies today will offer PHP support, so you shouldn't have a problem finding one. HostGator.com is an affordable solution, as is EagerHost.com. Both of these hosting companies fully support PHP.

Also, I want to make it clear that PHP is not a difficult language to learn, even if you have never tried to program before. In fact, if you are completely new to programming, PHP is often the easiest way to get started.

In the world of the wide web, there are two general types of coding languages: “Server side” and “Client side”. This means that one type is run on the web server itself, and the other is run in your web browser.

PHP happens to be a server side language. All of the processing is done on the web server itself, and the result is delivered to your web browser as HTML (which, by the way, is a Client side language).

To begin, php code looks like this:

<?php ← opens your code

```
Additional code here
```

?> ← closes your code

In fact, PHP always begins with <?php and always ends with a ?> If your code is not contained within these tags, the PHP engine on the web server will display errors or ignore it all together.

Another important point to make is that pages that contain php must end in .php in order to work correctly. This means if you create an index page that contains php code, it must end in .php such as index.php, not index.html. If you use .html, the webserver will once again ignore it. (There are a few exceptions to this but we will cover that in a later chapter).

You should also know that PHP can be embedded into HTML, and you don’t have to have raw PHP code in a PHP file, you can always switch between HTML and PHP, all on one page, as long as the page extension is .php (index.php, etc).

Here’s an example:

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

My First PHP Experience

<?php
//print output
echo ' My First PHP Experience’;
?>

</body>
</html>
```

If you look closely at the simple script above, you will see that the script starts off as any HTML page would, with an open HTML tag, a head tag, and a body tag. None of this is PHP, it's purely HTML, the language used to design web pages.

The PHP script starts where you see the line like this:

```
<?php
```

You can start to create a page in HTML as shown above and insert PHP script anywhere you like as long as you open it up with <?php

Another important aspect of PHP is that each line must end with a semi-colon (;). If it doesn't, your code might spit out errors.

Example: echo ' My First PHP Experience'; ← Note the semi-colon at the end of the line. This is very important! Without the semi-colon, this code would not work correctly.

As I mentioned above, in order to work with PHP you need a web server that allows it, which most do. The reason the web server needs to support PHP is because PHP code needs to be executed, and the PHP interpreter handles this.

Here's an example of how this works:

1) You load up your favorite browser and type in
www.somesite.com

2) The website, somesite.com receives your request to load the page.

3) The web server checks to determine what kind of document it is that is required to load. It detects that it is a .php page (index.php). The webserver calls on the PHP engine and transmits the contents of the index.php page.

4) The PHP Engine loads the document and everything between the open and close tags (Remember: <?php and ?>)and converts it into HTML output, so you can see it.

While this all sounds really complicated, it happens so quickly that all you see is the website load in your browser. The power behind it however is quite impressive.

Using the example above, if you were to load that page into your browser, you would not see the PHP code, all you would see is the output from the script.

Let’s Start Coding!

We’re going to tackle the basics to start with. Getting PHP to write things to the web browser, what variables are and how to use them...examples you can use in just about every web page you ever create.

Let’s start with the most basic one.

The PHP Echo Command Explained

We’re going to tell PHP to output something to the screen. Keep in mind that PHP can be used in conjunction with HTML as mentioned earlier, but we will keep it simple for this example and not use HTML at all.

EXAMPLE:

```
<?php echo “Warrior Forum Rocks”; ?>
```

It’s a lot better than the typical “Hello, World!” example used in countless php tutorial books, isn’t it? =)

Let’s look closer at the above command.

<?php – simply tells the web server to process this as PHP code. It’s the opening line to php.

Echo “Warrior Forum Rocks”; tells the server to write and display what is contained within the quotes and print it out on the screen.

The semi-colon means that the particular command has ended.

?> tells the webserver, “Allright! We’re done with PHP for now, let’s move back to regular old HTML”.

Also, if you wanted to display actual quotes on your webpage, you will need to do the following instead,

```
<?php echo “\” Warrior Forum Rocks\””; ?>
```

It’s a bit tricky, but once you get the hand of it, I promise, it will get a lot easier. Using the \” simply tells PHP that you want to display a quotation mark on your page.

When I started out, I created a swipe file of code, tips and even reminders to help me keep on track and avoid a ton of errors or problems. I suggest you do the same as you progress through your PHP training.

PHP Strings

A string is simply a sequence of characters (line of text) like “goodbye” or “hello” contained within quotation marks.   String values can be enclosed in either double quote (“ “) or single quotes (‘’). Strings could be considered the building blocks of PHP considering that most all of the data is going to come from what is contained within a string.

```
<?php
$double = “quotation marks.”;

$single = ‘quotation mark’.;

?>
```

When you use single quotes as the first example above indicates, you need to escape the apostrophe with a slash, like you would with double quotation marks, if you want it to display the quotation marks in the actual output text.

An example would be the following code:

```
<?php
echo ‘Don\’t you love being a warrior?’;
?>
```

Using the /’ will show the apostrophe on your page correctly.

There are also special commands that can be used within strings that allow you to manipulate and change the output text (the text that is displayed on the page). Here are a few common commands that are used:

\n; - this will make a new line

\r; - this is used for a carriage return

\t; - this is used to represent a tab

Single Quotes (‘’)

You use single quotes when you are displaying information that contains no variables in the code. For example:

```
<?php
Echo ‘Displaying a variable’;
?>
```

No Quotes

If you are displaying information by using a variable only, OR you are simply using the echo command, you are not required to use quotes at all. Example:

```
<?php
$variable = ‘displaying just a variable name’;
echo $variable;
?>
```

Double Quotes (“”)

If you would like to display (print out) a variable that is contained within a string or additional text/sentence, you then use double quotes like the following example:

```
<?php
$variable = ‘Jake Riley’;
echo “My name is $variable”;
?>
```

The (echo “My name is $variable”;) would display: My name is, and the variable - Jake Riley (My Name Is Jake Riley).

If a variable is to be displayed within words of a string, then the entire thing must be contained within double quotes, if you want it to display correctly.

PHP Variables

Variables are part of PHP programming and their purpose is to store a value, whether it be a number (integer), a sentence or phrase (string) or a letter (char). Every variable must have a name, and these names in PHP are case sensitive, meaning $you and $You would be considered different variables.

You simply assign a value to a specific variable using the assignment operator (the equal sign =)

Using variables in your code is very simple. This is how one looks:

```
$variable = 1;
```

A variable always begins with a dollar sign; followed by the name you decide to call it. Variable names can include letters, numbers and the underscore character, however you should never start variable names with a number.

Examples:

$var can be used correctly.
$var1, $var2, etc is just fine
$1var is not correct.

Declaring a variable is exceptionally easy. Just remember, you cannot have any spaces in the variable name, to avoid errors in your code.

Another example of a variable:

```
<?php
$Programming = “Programming Is Profitable”;
?>
```

Whatever you assign to your variable is what will be printed out on your page. For example, if you used $variable = 5 , your page would just show a “5”.

If you assign a sentence to a variable, the same thing happens.

Example: $variable = ‘Hello Warrior Friend.’;

The page would read “Hello Warrior Friend.”

You use the echo command to print out these variables onto your webpage (Example: echo $variable;)and the output would include whatever your variable contains.

Variables are a very important aspect of learning PHP, so you should become familiar with them relatively quickly.

Useful Variables

Here is an example of how you would use a variable, if you wanted to show the current date or time on your webpage:

```
<?php
$today = date("F j, Y");
echo "$today";
?>
```

The example above simply sets the date command as a variable that is called $today and uses the echo command to display the date on the screen.

Echo Multiple Variables

You can use the echo command to display more than one variable at a time. Combining multiple variables can be very useful and sometimes necessary to achieve specific results.

For example:

```
<?php
$phrase1 = “Apples are healthy.”;
$phrase2 = “Potato Chips Are Not”;
echo “$phrase1 $phrase2”;
```

```
?>
```

If you used this code, your webpage would show:

```
Apples are healthy. Potato Chips Are Not.
```

So, if you want to show different phrases on your page, you would use multiple variables to print out specific sentences or phrases.

You can also echo text and variables in the same statement by inserting periods around the variable like this:

```
<?php
$items = 3;
echo “You have purchased “.$items.” Items.”;
?>
```

This would show, “You have purchased 3 items”, since you have assigned the variable $items to the number 3.

PHP Includes & Problem Solving

A PHP include is used when you want to include the contents of one particular file inside of another.

```
<?php
include (“file.inc”);
?>
```

If you want to see a real example of how PHP Includes can save you time and help make managing and updating your website easier and faster, check out the following example.

Example of How PHP Includes Saves You Time

If you were developing a five page website that you plan to continue to add new pages to.

Your navigation HTML code looks like this:

```
<html>
<head>
<title>My Navigation Menu</title>
</head>
```

```
<body>
<a href= http://www.mysite.com/index.php>Home</a>
<a href= http://www.mysite.com/products.php>Products</a>
<a href= http://www.mysite.com/articles.php>Articles</a>
<a href= http://www.mysite.com/blog.php>Blog</a>
<a href= http://www.mysite.com/contact.php>Contact Us</a>
</body>
</html>
```

You want this navigation menu to appear on every single page on your website.

What happens when you want to add a new page to your site? You would typically have to change the menu on every single page to showcase the new link or item you wanted to add. The manual work involved would cost you a lot of time, and each time you went to update the links or add new ones, you would once again, have to update every other page that this menu appears on.

PHP Includes to the Rescue!

Simply cut and paste the navigation code into a plain text document using your favorite text editor. (Notepad, Text Edit, etc)

```
<a href= http://www.mysite.com/index.php>Home</a>
<a href= http://www.mysite.com/products.php>Products</a>
```

```
<a href= http://www.mysite.com/articles.php>Articles</a>
<a href= http://www.mysite.com/blog.php>Blog</a>
<a href= http://www.mysite.com/contact.php>Contact Us</a>
```

Now, save this text file as navigation.inc (or any other name you wish). You don't even have to use.inc as an extension for an include page, you can use almost any type of page extension including .php, or .html.

Now, go back to your webpage and remove the navigation code and replace it with this PHP code:

```
<?php
include "navigation.inc"l
?>
```

Then, save the HTML page as a .php page (index.php) and you're done! Anytime you want to change or update the menu items, all you have to do is edit one file – the navigation.inc text file!

Can you see the power of PHP, yet?

Tip When Creating Inc Pages:
Be sure to strip out all of the formatting code from the page you are including from (in this example, the text file.) Make sure that it does not contain things like <html>, <body> or anything like that.

Working Class Examples

We just discussed how you could save a ton of time updating your websites using simple PHP code. This time, we are going to show you how you can use PHP to make your job as a Webmaster even easier.

In our example, we are going to pretend we are developing a massive website with hundreds of content pages. We would like to showcase advertisements on individual pages of our site, perhaps Google Adsense. We would also like to test different color schemes to see what works best based on our current design. We also want to be able to change the ads periodically to determine what converts the best.

PHP will make it exceptionally easy to accomplish all of this quickly and effortlessly in comparison with editing every single page each time you want to change the color, or edit an ad.

Simply open up two blank text files. The first one is going to be our “settings file”, and the second one will contain our Google Adsense code.

In the settings file, we are going to want to set variables for the things that we are going to change from time to time, for example the main Adsense publisher code and the color scheme.

We will use the following variables to achieve this:

```
<?php
$ad_pub_number = “pub-012345678”;

$eb_linkcolor = “006699”;
?>
```

The variable, $ad_pub_number contains our AdSense publisher tracking code, and the $eb_linkcolor contains the color code that we want to use on this page (you can use any color code you like). In our example, the color 006699 is a navy blue.

Now, save this page as ‘settings.php’

It’s time to log into your Google account and copy the AdSense code snippet that appears when you create a publisher campaign.

Our example:

```
<script type="text/javascript"><!--
google_ad_client = "pub-0123456789";
google_ad_width = 120;
google_ad_height = 600;
google_ad_format = "120x600_as";
google_ad_type = "text_image";
google_ad_channel = "1";
google_color_border = "FFFFFF";
google_color_bg = "FFFFFF";
google_color_link = "006699";
google_color_text = "006699";
google_color_url = "006699";
//--></script>
  <script type="text/javascript"

src="  http://pagead2.googlesyndication.com/pagead/show_ads.js
">
</script>
```

Paste this into the blank text file.

Now, we have the code available, but in order for it to work, we will need to make a few small changes.

At the beginning of the page, add in the following line of PHP:

```
<?php echo “
```

and at the very end of the page, close the php with

```
?>
```

We also have to change the “google_ad_client = "pub-0123456789";” to our variable which was $ad_pub_number

We also want to change the colors of the links to include our previous variable, $eb_linkcolor

So now your code will look like this:

```
<?php echo “
<script type="text/javascript"><!--
google_ad_client = \"$ad_pub_number\”;
google_ad_width = 120;
google_ad_height = 600;
google_ad_format = "120x600_as";
google_ad_type = "text_image";
google_ad_channel = "1";
google_color_border = "FFFFFF";
```

```
google_color_bg = "FFFFFF";
google_color_link = \"$eb_linkcolor\";
google_color_text = \"$eb_linkcolor\";
google_color_url = \"$eb_linkcolor\";
//--></script>
<script type="text/javascript"
src="  http://pagead2.googlesyndication.com/pagead/show_ads.js
">
</script>
?>
```

Save this page as ads.php

If you look closely at the changes we made you will see that we told PHP to echo the code onto the screen and include the variables that we set in the settings.php file such as color scheme. We also used the quotation marks in the output with the slash \ since we wanted to display the quotation marks on the page. (\”$eb_linkcolor\”;)

We’re not finished either. We need to include a snippet of PHP on our content pages so that it calls the settings.php and uses the information we set up.

```
<?php
require “settings.php”;
?>
```

This tells the page that it requires the contents of the settings.php file to be included. You have to copy and paste this code at the very top of each content page that you want to use the settings.php page with.

Now, to finish off this task, we need to also call the ads.php page that we created that actually contains the ad. To do this, we simply need to include the following code wherever we want the ad to appear:

```
<?php
include “ads.php”;
?>
```

This will pull the code from the ads.php page that we created and show these ads wherever we place this code. Be sure to use .php as the extension of each content page (article.php, content.php etc) to ensure that your code works correctly.

Whenever you want to change the publisher code or the colors used, you just have to edit the settings.php file to instantly update the entire site (or every page that includes the code telling it to include the settings file).

Redirecting Using PHP

PHP is also very useful if you are looking to redirect your visitors to other pages or websites. For example, if you are updating your website and do not want people to see the index page, you could use PHP to direct them to a temporary page until you have completed the updates. You could also use PHP to redirect visitors to an external product page using your affiliate link.

The PHP code required for a successful redirection is really short and sweet. It looks like this:

```
<?php
header (“Location: http://www.whatever-site.com”);
?>
```

Whenever a visitor lands on your page they will be instantly directed to the URL that is contained within the brackets.

By now, you have a good understanding of what PHP is and how useful it can be to web developers. You also know how to add basic PHP functionality to your webpages.

Commenting Your Code

One important aspect of PHP programming is including comments within your code. Comments are lines of text that help explain what certain parts of the code do, but this text is not visible to anyone but those who have access to the source code itself.

The reason commenting is important is to be able to quickly go back through your code, if ever you want to update it, or share it with other developers and be able to remember exactly what each part of your code does. The most professional coders learn to comment early on, as a way to stay organized and to save time when you are required to go back into your code months from now and make changes.

Here is an example of code that includes comments. Remember, comments are only visible to those who have access to the source, they will not be displayed on your website. (I've seen some pretty funny comments left for me by fellow coders in the past. It can be quite entertaining! =)

```
<?php
 •
 // This is a single comment line.

 # This is also a single comment line.

 /* This is used for block comments, if you are working on multi-line
 comments, or are leaving lengthy notes in your code.

 ?>
```

You do not have to use semi-colons after each line, since the PHP server will ignore all of your comments in the first place.

The IF Statement

The “IF” Statement in PHP is very similar to using the word IF in every day terms. For example, If you do not finish reading this eBook you will not know everything you need to know in order to use the power of PHP =)

In PHP, “IF” and “Else” are known as conditionals. Let’s take a look at how PHP compares these values for conditionals. You will also see what is called “operators” in any IF statement.

==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

So, a valid IF statement could consist of the following:

```
<?php
if ($variable == “some value”) {
echo “Correct”;}
?>
```

You will most likely want to use IF with ELSE. Else gives you the option of doing something else with your PHP script in the event IF does not calculate one way or another.

For example, you can have your website display specific text if a condition is met (like a password is entered correctly), or something ELSE if not (like a password entered incorrectly).

An example of this sort of code would look like this:

```
<?php
$five = “5”;
if ($five == “5”) {
echo “Correct Password!”;

} else {

echo “Incorrect Password!”;
}
?>
```

Using this, you could create a simple password protected area using the IF and ELSE command. A PHP page with a conditional statement could be set up to process using a HTML login form. Then, a variable $password , could be used to detect success or failure.

```
<form action="login.php" method="post">
<INPUT type="password" name="password">
<input type="submit">
</form>
```

This HTML form will set the variable in the “action” page (named login.php in this example) with the $_POST command (more on this later) and do one of two things: If the password is correct, it will show the desired content. If it is NOT correct, it will redirect to another URL (or page). Since the action is in PHP, viewing the source in the web *browser isn't going to reveal the password.*

And your PHP Action page would include an IF/Else combination, with perhaps a redirection command if the password was entered incorrectly.

```
<?php
if($_POST['password'] == 'some_password'){

echo "

<!---Put your protected HTML content here...-->
" ;
} else {
header ("location: some_error_page.html");
}
?>
```

This is a very simple way to password protect a specific page, and while it isn't bullet proof secure, it could easily protect non-sensitive information.

Multiple Conditions

The IF / Else statement is very useful if you need to check for only one specific condition, however a third party comes into play when you are required to use multiple conditions.

Example: If you need to check to see if a truck is either a Dodge, Ford or Chevy, you would need to use what is called an "ElseIf" statement.

```
<php

$truck = "Chevy";
if($truck == "Dodge"){
        echo "It’s Ram Tough!";
} elseif {$truck == “Ford”}
        echo “Built Ford Tough!”;
} else {
        echo "We’ll Be There!";
}

?>
```

ElseIf cannot be used without the condition IF however, so what happens if you have multiple instances of “ElseIfs’?

Often times, we have to evaluate multiple conditions, in this example we have to list many different truck models, and in these instances using “ElseIf” can be a tad cumbersome (writing out a dozen “ElseIf’ conditions is quite tedious). In this case, we can use a short-cut known as a “Switch Command”.

Let’s add a handful of other truck models to our inventory using a “Switch” so we can show you how it works:

```
<?php
$truck = "Chevy";
echo "Drive a $truck, <br/>";
switch ($truck){
        case "Dodge":
                echo "Ram Tough!";
                break;
        case "Ford":
                echo "Built Ford Tough!";
                break;
        case "Toyota":
                echo "Got The Guts?";
                break;
        case "Nissan":
                echo "Shift_power";
                break;
        case "GMC":
                echo "Professional Grade";
                break;
} ?>
```

Using the Switch Command makes the code shorter and a lot cleaner, and minimizes the number of “If/Else” statements used. However, when you do use the Switch command, be sure to include the “break” statement, as it terminates loops and ends the process. Failure to use a break will result in the information continually being processed.

In order for a switch to work correctly, you need to also add a ‘default’ to the code in the event that no match can be found.

```
<?php
$truck = "Chevy";
echo "Drive a $truck, <br/>";
switch ($truck){
        case "Dodge":
                echo "Ram Tough!";
                break;
        case "Ford":
                echo "Built Ford Tough!";
                break;
        case "Toyota":
                echo "Got The Guts?";
                break;
        case "Nissan":
                echo "Shift_power";
                break;
        case "GMC":
                echo "Professional Grade";
                break;
        default:
                echo "We'll Be There!";
                break;
} ?>
```

So, if there are no matches, our default will be used in its place.

PHP Arrays

An array can be thought of as a single variable that stores more than a single value. An array uses a “key” to determine what value to reference.

```
<?php

echo $array[0];

?>
```

An array is accessed by the name of the array, followed by square brackets containing the key of the array item we want to access.

```
$array[key]
```

An array contains an array of values, each of which are referenced by this “key”.

We use the keys to access each individual item within the array.

We would assign an array like this:

```
<?php

$array = array(
    'first value',
    'second value',
    'third value',
    'fourth value' );

?>
```

All that information would be stuck into the $array array. We could then access each item:

```
<?php

echo $array[0];   // prints "first value"
echo $array[1];   // prints "second value"
echo $array[2];   // prints "third value"
echo $array[3];   // prints "fourth value"
```

```
?>
```

The key index consists of numbers, the first value in the array is accessed with 0, NOT 1. The second accessed by 1, the third by 2, and so forth.

```
$array[key] = value;
```

Key values typically start at 0, as PHP likes to number things starting at Zero, instead of 1.

Using our truck example from earlier, let’s assign them using an array.

```
<?php
$truck_array[0] = "Toyota";
$truck_array[1] = "Dodge";
$truck_array[2] = "Chevy";
$truck_array[3] = "Ford";
?>
```

And here is how we would output the information from this array:

```
<?php
echo "Two great truck dealers are "
. $truck_array[0] . " and " . $truck_array[1];
echo "<br />Two more great truck makers are "
. $truck_array[2] . " and" . $truck_array[3];
?>
```

Finally, here is the output result from the above array:

```
Two great truck dealers are Toyota and Dodge.
Two more great truck dealers are Chevy and Ford.
```

Associative Arrays

An associative array is an array in which the keys are associated with values.

```
<?php
$truck[“Toyota”] = Tundra;
$truck[“Nissan”] = Titan;
$truck[“Dodge”] = Ram;
?>
```

A syntax example using the Associative Array above would look like this:

```
echo "Toyota makes the " . $truck[“Toyota”] . "<br />";
echo "Nissan makes the " . $truck[“Nissan”] . "<br />";
echo "Dodge makes the " . $truck[“Dodge”];
```

When viewed in a browser, it would look like this:

```
Toyota makes the Tundra
Nissan makes the Titan
Dodge makes the Ram
```

You don't have to access array items with number values starting at 0, you can simply define your own keys.

You may not see the usefulness of the Array and Associative Array right now, however you will discover just how handy they are in upcoming lessons.

Associative arrays allow you to associate different words with each array item, it is easier to remember which word accesses what item in large array, your mind remembers words easier than having to count, starting from zero, in your head.

Remember the single quotes/double quotes thing? This applies when accessing array items with keys also. An integer value or a variable needs not be quoted, a string value needs single quotes, a string value containing a variable needs double quotes.

Special Arrays

$_SERVER

This array contains lots of values for you to use in your scripts. It contains the client's IP address, the browser version, the URI of the script being access, various HTTP headers and much more, here are some examples.

$_SERVER['REMOTE_ADDR']; // client's IP
$_SERVER['HTTP_USER_AGENT']; // client's user-agent string
$_SERVER['REQUEST_URI']; // requested URI
$_SERVER['HTTP_X_FORWARDED_FOR']; // HTTP-x-forwarded-for header

$_POST

This array contains anything that has been submitted through the POST method. We will be looking at this in the next chapter.

$_GET

This array contains anything that has been submitted through the GET method. We will be looking at this in the next chapter.

$_COOKIE

Contains cookie information for the client. Say you set a cookie on their machine named “mycookie” with a value of “Jester”.

$_COOKIE['mycookie'] would contain 'Jester'.

$_REQUEST

This associative array consists of everything in $_POST, $_GET, $_COOKIE and $_FILE.

PHP Loops

Using loops in conjunction with arrays is a great way to improve your PHP scripts, and eases the workload of repetitive tasks.

The first type of Loop we will discuss is a “While Loop”. This is one of the most useful loop functions in PHP, and looks like this:

```
<?php

while(this_condition_is_true) {
   // perform this task
}

?>
```

The structure of a while loop looks very similar to an if statement. PHP checks to see if what is inside of the parenthesis evaluates to be true.

If so, everything that is nested inside of the curly braces is performed. PHP continues to perform those tasks until the condition in the parenthesis evaluates to false.

```
<?php

// Let's count to 10!

$i = 1;
while($i <= 10) {
    echo "$i<br />\n";
    $i++;
}

?>
```

What is happening here is very simple. First off, we declare a variable, $i, that holds the value of 1. This variable is what we'll use for the conditional of our while loop as well as to echo out our numbers from 1 to 10.

We then tell PHP that we want it to perform a loop. Our loop will continue so long as the value of $i is less than or equal to 10 ($i <= 10). Each time PHP iterates through the loop we tell it to echo out our variable and then increment it by one.

Real World Usage for WHILE Loops

The Internet Marketing Company wants to show a pricing matrix on their website for their current product packages. Since their products are so popular, the price fluctuates constantly to reflect the current demand for their services.

Using a “While Loops” along with a bit of HTML can make the process of changing the pricing structure, really easy. In fact, you’ll only have to change one value every time the pricing changes!

```
<?php
$product_price = 2.15; //This is the price of one eBook
$counter = 1;

echo "<table border=\"1\" align=\"center\">"; //note the escapes for
quotes!
echo "<tr><th>Quantity</th>";
echo "<th>Price</th></tr>";
while ( $counter <= 20 ) { //we just set the counter limit at 20
        echo "<tr><td>";
        echo $counter;
        echo "</td><td>";
        echo $product_price * $counter; //here we multiply
        echo "</td></tr>";
        $counter = $counter + 1; //here we add 1 to the counter and
start again
}
echo "</table>";
?>
```

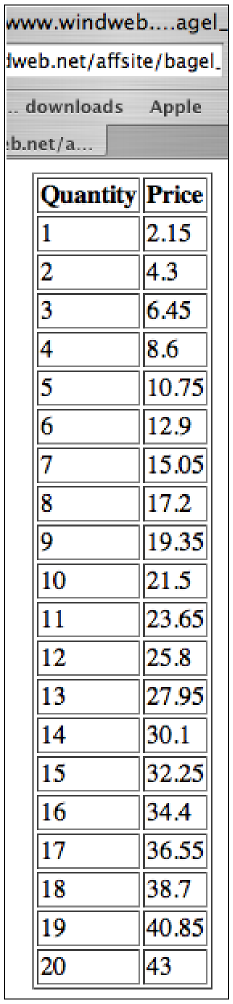
Our actual Loop code is highlighted in blue. You will see that we've set the variables $product_price and $counter. The $counter variable allows us to create a math function to increment the unit price by 1 every time we see a loop (see the $counter = $counter +1 part of the code). So, while the $counter equals less than or equal to 20, this loop will function and remain as is.

Once we hit 21, that's it – the loop stops!

Note on PHP Math Functions:

+ addition (You can use ++ to increment a value by 1)
- subtraction
* multiplication
/ division

Here is what it looks like in a browser. You can quickly see just how much money 20 eBooks would cost you at the current price.



Quantity	Price
1	2.15
2	4.3
3	6.45
4	8.6
5	10.75
6	12.9
7	15.05
8	17.2
9	19.35
10	21.5
11	23.65
12	25.8
13	27.95
14	30.1
15	32.25
16	34.4
17	36.55
18	38.7
19	40.85
20	43

The FOR Loop

The For Loop is very similar to a WHILE Loops, the only different being a little more code is contained within the loop. The FOR Loop can be a little more compact than a While Loop.

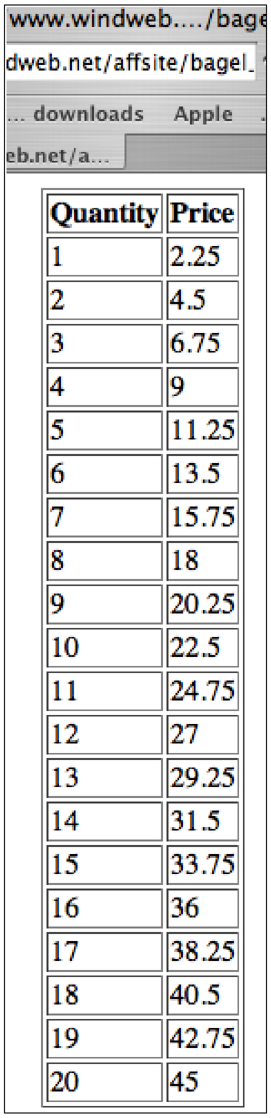
Here's the logic;

```
for ( create a counter; conditional statement; increment the counter){
        do this;
}
```

Let's use our fluctuating ebook cost from our earlier example and write this out. The products just went up by $.10!

```
<?php
$product_price = 2.25;

echo "<table border=\"1\" align=\"center\">";
echo "<tr><th>Quantity</th>";
echo "<th>Price</th></tr>";
for ( $counter = 1; $counter <= 20; $counter += 1) {
        echo "<tr><td>";
        echo $counter;
        echo "</td><td>";
        echo $product_price * $counter;
        echo "</td></tr>";
}
echo "</table>";
?>
```



Quantity	Price
1	2.25
2	4.5
3	6.75
4	9
5	11.25
6	13.5
7	15.75
8	18
9	20.25
10	22.5
11	24.75
12	27
13	29.25
14	31.5
15	33.75
16	36
17	38.25
18	40.5
19	42.75
20	45

Once again, the loop is highlighted in blue above. You will notice that the counter is defined inside of the loop, as opposed to a variable located outside.

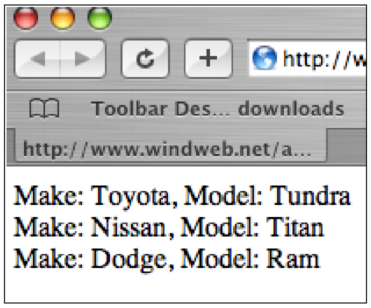
There is one more loop we are going to cover, and it's called the "For Each Loop".

For Each Loop

If you wanted to loop through an associative Array, you would use the For Each to complete this task. Where the WHILE and FOR loops run until an error is encountered, the FOR EACH loop will run through every single element in the array.

Let's go back to the associative array that we set up for the truck inventory earlier:

```
<?php
$truck[“Toyota”] = Tundra;
$truck[“Nissan”] = Titan;
$truck[“Dodge”] = Ram;
?>
```



```
Make: Toyota, Model: Tundra
Make: Nissan, Model: Titan
Make: Dodge, Model: Ram
```

To loop through the models, we would use this PHP code:

```
<?php
foreach( $truck as $make => $model){
        echo "Make: $make, Model: $model <br />";
}
?>
```

Functions

A function in PHP is really quite easy to understand. It's a piece of code that can be named and re-used at any time. Functions can help us reduce our coding time significantly, by simply writing the code once and then defining it as a function that can be called on at any time to appear.

This is how it works:

```
<?php
Function MyFunctionName () {
// define your function here
}
?>
```

Let's pretend that you want every page on your website to feature your company name. You would create a function to do this, and call it "MyCompanyName".

```
<?php
function MyCompanyName(){
}
?>
```

Now, let's add a bit of code in between the curly brackets, that we plan to execute.

```
<?php
function MyCompanyName(){
echo "Welcome to The Creamy Bagel Company! ";
}
?>
```

Whenever you want your company name to appear on a page, you simply call the function within a PHP tag and it’s done!

```
<?php
function myCompanyName(){
 echo "Welcome to The Super Smooth Programming Corp!<br
/>";
}
echo “Isn’t it about time you experienced a better coding
company?<br />”;
myCompanyName();

?>
```

A function can contain just bout any type of PHP code imaginable, so you should quickly see just how useful this can be, as well as how much time you can save when you begin coding.

PHP Sessions

When you dive deeper into PHP, and come to the point where you are interested in displaying specific user/visitor data throughout different areas and pages on your website, it's time to learn PHP sessions!

(Think Shopping Carts, etc)

A session is simply, a way to store information (in the form of variables) that will be used throughout multiple pages. Unlike a cookie that can be placed or stored on your computer, PHP sessions do not store information this way.

It is also different than other variables you might use, in the sense that you are not passing them individually to each separate page, but instead simply retrieving them from the initial session that is opened at the start of each page.

Here's an example of a session:

```
<?php
// this starts the session
session_start();

// this sets variables in the session
$_SESSION['color']='blue';
```

```
$_SESSION['size']='large;
$_SESSION['shape']='round';
print  "Done";
?>
```

The first thing we need to do is to open a session using the command: session_start();

After you open up a session, you can then set variables, for instance, colors, sizes, etc. In our example, we have set it as color blue, size large and shape round, in that order.

One thing you should know is that in order for this code to work, you must place the session_start() code in the header of your page, and you can not send anything to the browser before that code, therefore it's best to just place it directly after the <?php to avoid any problems or errors.

Most sessions store a cookie on your computer to use as a "key".

The web server will automatically add a lengthy, random 'session ID" that represents a unique session. It may look similar to this:

a8486d5484a654255csa55ca54ddg5d5g478

When a session is opened on another page, it quickly scans your computer for a key. If one is found and it matches, it allows access to that session. If no key is found, it will start a new session for you.

Sessions are used to store data, and to do this we can use an associative array, based on the $_SESSION variable.

For example, look at the following code used for login form sessions:

```
<?php
session_start(); //  Starts a PHP session
echo "<form method=POST action=index.php>
  User Name: <input type=text name=\"username\">
  Password: <input type=text name=\"password\">
  <input type=submit>
  </form>";  // This is the HTML form
$_SESSION['username']=$_POST["username"]; // Enters the
username into the array
$_SESSION['password']=$_POST["password"]; // Enters the
password into the array
?>
```

Using this code, your username and password are now stored in an array that will last until the season has ended or un-set. An ended session occurs when either a viewer closes their browser or you tell PHP to run a command and end or terminate the session.

You can change this in the php.ini file on your server by changing the 0 in session.cookie_lifetime = 0 to the number of seconds that you would like the session to last for, or by using session_set_cookie_params().

```
<?php
session destroy();
?>
```

Or, we can edit or remove individual session variables, as well as the entire session. To change a session variable, we just reset it to something different.

```
<?php
// you have to open the session to be able to modify or remove it
session_start();

// to change a variable, just overwrite it
$_SESSION['size']='large';

//you can remove a single variable in the session
unset($_SESSION['shape']);

// or this would remove all the variables in the session, but not the
session itself
session_unset();

// this would destroy the session variables
```

```
session_destroy();
?>
```

This will abruptly end the session and clear out any associated data that was stored from the current session.

You can use the unset() variableto remove a single variable or session_unset() to remove all variables for a specific session.

Of course, you can also use session_destroy() to destroy the session entirely.

If you want to remove specific data from a session without deleting the entire thing, you can call on the “IF” statement and combine it with two commands, the first is called the “ISSET” command and the second, the “Unset” command.

Here is how it might look:

```
<?php
if(isset($_SESSION['items'])){
unset($_SESSION['items']);}
?>
```

Looking at this code, if the specific key is set in this session, it will remove the keys’ data.

Cookies

You have probably heard about cookies, in fact, you most likely have dozens stored on your computer from visits to different websites that you visit regularly or log into. As mentioned earlier, cookies store information on your computer from these visits. This allows websites to remember you when you return, and can make things easier for you by remembering your login details, etc.

You can use PHP to create and set cookies easily.

```
<?php
setcookie(name, value, expiration);
?>
```

There are however, three required factors when setting a cookie on a users computer.

1) Name : This is where you simply set the name of your cookie so you can retrieve it at a later date. You can call the cookie anything you want.

2) Value: The value that you wish to store inside of the cookie. Some common values are usernames, or perhaps the date of your last visit.

3) Expiration: You set the expiration date for your cookie in regards to when it is to be deleted from the users computer. If you do not set an expiration date on your cookie, the cookie will

be deleted when the visitor closers their browser, so if you want it be retained, you need to ensure you set a specific expiration period.

Obtaining information from a cookie is as easy as initially setting on. If you recall the “ISSET” command from earlier, PHP extracts cookie data in a very similar way, by creating an associative array to store the cookie data using the $_Cookie variable. The array key is what you called the variable when it was set. This means you can set as many cookies as you wish, with no limits.

Cookies are stored in small text files on your vistors computer, so they don’t take up much room at all.

Here is an example of code that sets a cookie:

```
<?php
if(isset($_COOKIE['username'])){ // If there’s a cookie...
$username = $_COOKIE['username']; // set the variable...
echo "Welcome back, $username."; // and show the data in the
cookie...
} else {
echo "No username was found. Sorry!";} // or show this if there
wasn’t one.
?>
```

Processing Forms

If you own a website, odds are that you will find it very handy to learn how to use PHP to process forms, such as Contact Forms, or even question based survey forms. To do this, we use the PHP mail() command.

Simply create a HTML form that includes the following fields:

Name, Email and Message

Here is an example of a simple HTML form that can be used to collect data:

```
<html>
<head><title>Test Page</title></head>
<body>
<h2>Data Collection</h2><p>
<form action="process.php" method="post">
<table>
<tr><td>Name:</td><td><input type="text" name="Name"
/></td></tr>
<tr><td>Age:</td><td><input type="text" name="Age"
/></td></tr>
<tr><td colspan="2" align="center"><input type="submit"
/></td></tr>
</table>
</form>
</body>
</html>
```

This page contacts <form action=”process.php” method=”post”> which means that the page will send the Name and Age data to the page, “process.php”.

Now, we have to create the process.php page that will use the data from the HTML form:

```
<?php
print "Your name is ". $Name;
print "<br />";
print "You are ". $Age . " years old";
print "<br />";
$old = 25 + $Age;
print "In 25 years you will be " . $old . " years old";
?>
```

If you leave out the method="post" section of the form, the URL will show the data. For example, if your first name is Henry Adams and you are 47 years old, our process.php page will display as:

http://your-website.com/process.php?Name=Henry+Adams&Age=47

If you want, you can manually change the URL and the output will change accordingly. Also note, that everything after the ? is called the "Query String".

You can also retrieve information through he form using the "Get and Post" methods). Here is how it's done:

```
<form action="script.php" method="post">
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname"><br>
<input type="submit" value="Submit">
</form>
```

The HTML code would create a form on your website and would showcase two input boxes. One called “firstname” and one called “lastname”. If you look in the code above, you will see that the form is being told to submit the form to ‘script.php’

```
<form action = “script.php” method = “post”>
```

Now, let’s assume that our visitor entered in Marty as their first name and Reno as their last name.

```
<?php

// how we get form info with a POST form
// $_POST['firstname'] contains what they entered in "firstname"
input
// $_POST['lastname'] contains what they entered in "lastname"
input

echo '<p>Hello '.$_POST['firstname'].'
'.$_POST['lastname'].'</p>';
```

```
// would output:
// <p>Hello Marty Reno</p>

?>
```

If you use the POST form method, then all the input values will be contained within an associative array, POST. If you submit an input that is called “name”, it will be available in $_POST['name'].

If you submitted the form using the GET method, it would change.

```
<form action="script.php" method="get">
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname"><br>
<input type="submit" value="Submit">
</form>
```

Notice how the only change in the form above is that we changed the “method” to GET

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

Then the form:

```
<?php
```

```
// how we get form info with a GET form
// $_GET['firstname'] contains what they entered in "firstname"
input
// $_GET['lastname'] contains what they entered in "lastname"
input

echo '<p>Hello '.$_GET['firstname'].' '.$_GET['lastname'].'</p>';

// would output:
// <p>Hello Marty Reno</p>
?>
```

If we submit a form through the GET method then we simply use the $_GET array rather than the $_POST array.

Introduction to MySQL

MySQL and PHP often work closely together especially for web applications and developments. MySQL is often used in conjunction with PHP for the same reason that made PHP so popular. It is free, widely available and the majority of web hosts support it. There are, however other database systems that PHP works with, but MySQL is by far, the most common and well known option.

In this chapter, we will learn how to use MySQL to connect to a MySQL database and perform basic functions.

Don’t worry, MySQL’s commands are written out in a very easy, and comprehensive format, similar to just regular English.

Making The Connection

To begin, you will need to be able to command PHP to connect to a database before you can actually access and use it. The command is relatively straight forward, using mysql_connect().

```
<?php

$dbhost = "localhost"; // the db’s server

$dbname = "mysite_dbname"; // the db’s name

$dbuser = "mysite_dbuser"; // the db’s username

$dbpass = "password"; // the password for the user

mysql_connect ($dbhost, $dbuser, $dbpass) or die (mysql_error()); //connect to db
or show error if failure

echo "Connected to database";

?>
```

Analyzing the code above, the majority of the code should be pretty easy to understand. The dbname means “Database Name” (whatever you called the database when you created it inside of your hosting account, on your server). The dbuser, is the username you assigned to it, and so on.

One thing you may not understand however, is the “or die” commany, which simply helps direct PHP with how to handle any problems or errors it encounters, such as not being able to connect to the database because you entered incorrect information (like a wrong username or database name, etc). The "or die(mysql_error())" tells PHP to display the error, if there is one, so we can determine what caused the connection failure to occur.

If you do experience a connection problem, odds are it was caused by a simple mistake when entering in the database details. It's easy to make a mistake, so before you panic when seeing this error, just open up the page that contains the MySQL details and verify they are correct.

If all goes well, you will be connected to your database. So what's next? Inserting data into the database, and using it effectively.

Here's how we could do it;

To begin, you need to select the database using the mysql_select_db() command like this:

```
<?php
mysql_select_db("my_db") or die (mysql_error());
?>
```

Replacing my_db with the database name.

Next, we can create tables within the database, which will categorize and store information and data. Before that happens however, you will need to determine exactly what kind of information you want to store.

Here is a simple example of a table creation command using the mysql_query:

```
<?php mysql_query("

CREATE TABLE Employees

(employee_id INT,

first_name  VARCHAR(50),

last_name VARCHAR(50)")

?>
```

You will need to define the type of data that is going to be placed inside of each table field, so you should know exactly what data you want to be stored before you begin to create database tables.

Here are a few common types of data fields:

CHAR() – Fixed length field between 0 and 255 characters in length that can store any type of data you desire. The length of the field is set regardless of the size of the actual data that is placed inside of it.

DATE – This would store the date (YYYY – MM – DD) format.

INT – Whole numbers only. The number of digits can also be specific in parentheses but is not required.

TEXT - Variable length text only field with a maximum length of 65535 characters.

VARCHAR() – This variables length field must be between 0 and 255 characters and can store any type of data. Unlike the CHAR() type of field data, the length of each VARCHAR() field is determined by the data that is actually placed into it.

The Primary Key

Spreedshot programs such as Microsoft Excel utilize row numbers to identify what row specific data should be stored in.

Databases have a similar identifier called the “Primary Key”.

You can draw out your database with relative ease, or you can create a mock-up of it in a spreadsheet.

For example:

When you create the tables there are going to be additional details that you are going to have to include, in order to remember what the primary key is when you make a new row of data, etc.

This extra information includes “Auto_Increment” and NULL or / NOT NULL.

Here is an example:

```
<?php
// Make a MySQL Connection
mysql_connect("localhost", "user", "password") or die(mysql_error());
mysql_select_db("test_db") or die(mysql_error());

// Create a MySQL table in the selected database
mysql_query("CREATE TABLE Employees(
id INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY(id),
 employee_id INT,
 first_name VARCHAR(50),
 last_name VARCHAR(50)")
 or die(mysql_error());

echo "Table Created!";

?>
```

Let’s take a closer look at this code:

First of all, we made a connection to the MySQL database and selected the database that we wanted to access and work with.

Then, we made a table called “Employees”, and included a column called “ID” that will automatically increment by 1 each time a new record is created. This has a NOT NULL value (which simply means that the ID value is real and searchable), and that the Primary Key is set to be the ID field. Next, we created three data fields (employee_id, first_name, and last_name), and included the “or die” command that will help us determine when/if something goes wrong.

IF NOT EXITS

People have asked if creating a table that is already being used can “break” a script, and the answer is absolutely.

This is why we want to use the “IF NOT EXISTS” command with the CREATE command. This will prevent the table from being created if an identical one already exists. If this is a brand new database that you just created however, you will not have to be concerned with existing tables.

It is a good idea to get into the habit of using the “IF Not Exists” command early on though, for when you are adding tables to an existing database structure.

```
<?php mysql_query("
CREATE TABLE IF NOT EXISTS Employees
(employee_id INT,
first_name VARCHAR(50),
last_name VARCHAR(50)")
?>
```

The “If Not Exists” command can be used for other things in MySQL as well, such as creating databases. You can use it and it’s partner “If Exists” in data selects and other queries as well.

Inserting Data

Using the “Employees” example above, here’s the syntax for inserting data into an existing table. You can save time by using a PHP function or an include to set up the MySQL_connect statement, since it has to exist whenever you connect to a database.

```
<?PHP

mysql_connect (My_DB, user, pass) or die (mysql_error());

mysql_query("INSERT INTO Employees

(employee_id, first_name, last_name) VALUES(‘103’, 'Michael', 'Bolton' ) ")

or die(mysql_error());

?>
```

Calling Data with Select, From

Placing data into a table is important, however there are times you are going to want to extract data from a database. Here is where the Select and From command come into play.

Select – selects data from a table

If you wanted to select all of the data from an existing table, you would use a * symbol which equates to “all/everything”.

(Example: Select*From Table)

If you wish to select specific data from the table, by the row's Primary Key, you would use this:

(Example: Select 3 From Table)

You can also use "Where" to dig into the data and extract what you are looking for by using:

Select*From Table Where Last_Name=" Petersisonian"

Let's take a look at the Employee database table once again:

```
<?PHP

mysql_connect (My_DB, user, pass) or die
(mysql_error());

mysql_query("SELECT * FROM Employees WHERE
last_name='lumbergh'") or die(mysql_error());

?>
```

Now, what if you couldn't remember how to spell Petersisonian (it is a rather unusual last name, after all). You could locate all last names that begin with the letter P, to narrow down your search. Just use the % as a wildcard, like this:

```
mysql_query("SELECT * FROM Employees WHERE
last_name='P%'") or die(mysql_error());
```

Using this command, the search query will return all data in the last_name field that begins with the letter “P”.

Updating Data

If data needs to be modified, you can update it using the “Update” and “Set” commands. For instance, let’s pretend that Mr Petersisonian changed his last name to Peters. We would simply update the field like this:

```
<?php

We’ll skip the connection, you should have it as a
function already!

mysql_query(“UPDATE Employees SET
last_name='Peters' WHERE employee_id='104'”)

?>
```

Deleting / Removing Data

Just as we may wish to add data, there are times when removing data is required.

Here is how we could do this:

```
<?php

Once again...you should be able to connect by now!

mysql_query(“DELETE FROM Employees WHERE
employee_id='104'”)

?>
```

You are identifying the person you wish to delete by using the “employee_id” which would be specific and unique to each user.

Final Thoughts

Learning PHP can be an enjoyable and rewarding experience if you take it one step at a time and try to avoid becoming overwhelmed with the many ways that PHP can be used to power and manage websites and applications.

Spend some time exploring the PHP tutorial websites, try your hand at simple code before moving into the more advanced methods, and you will quickly discover just how simple it really is.

Listed below are a few useful resources that will help you continue along your way to mastering PHP.

PHP Main Page:
 http://www.PHP.net

MySQL Main Page:
 http://www.MySQL.com

PHP Builder Community:
 http://www.PHPbuilder.com

PHP Resource:
 http://www.php.resourceindex.com

PHP Tutorials:
 http://www.GoodPHPTutorials.com

Video PHP Tutorials:
 http://www.KillerPHP.com

For Resale Rights And PLR Ebooks & Software
The Best Resource BAR None is:
 http://www.publisherproducts.com