Escaping to PHP

The PHP parsing engine needs a way to differentiate PHP code from other elements in the page. The mechanism for doing so is known as 'escaping to PHP'. There are four ways to do this −

Canonical PHP tags

The most universally effective PHP tag style is −

```
<?php...?>
```

If you use this style, you can be positive that your tags will always be correctly interpreted.

Short-open (SGML-style) tags

Short or short-open tags look like this −

```
<?...?>
```

Short tags are, as one might expect, the shortest option You must do one of two things to enable PHP to recognize the tags −

- Choose the --enable-short-tags configuration option when you're building PHP.
- Set the short_open_tag setting in your php.ini file to on. This option must be disabled to parse XML with PHP because the same syntax is used for XML tags.

ASP-style tags

ASP-style tags mimic the tags used by Active Server Pages to delineate code blocks. ASP-style tags look like this −

```
<%...%>
```

To use ASP-style tags, you will need to set the configuration option in your php.ini file.

HTML script tags

HTML script tags look like this −

```
<script language = "PHP">...</script>
```

Commenting PHP Code

A *comment* is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP −

Single-line comments − They are generally used for short explanations or notes relevant to the local code. Here are the examples of single line comments.

```
<?
   # This is a comment, and
   # This is the second line of the comment

   // This is a comment too. Each style comments only
   print "An example with single line comments";
?>
```

Multi-lines printing − Here are the examples to print multiple lines in a single print statement −

```
<?
   # First Example
   print <<<END
   This uses the "here document" syntax to output
   multiple lines with $variable interpolation. Note
   that the here document terminator must appear on a
   line with just a semicolon no extra whitespace!
   END;

   # Second Example
   print "This spans
   multiple lines. The newlines will be
   output as well";
?>
```

Multi-lines comments − They are generally used to provide pseudocode algorithms and more detailed explanations when necessary. The multiline style of commenting is the same as in C. Here are the example of multi lines comments.

```
<?
   /* This is a comment with multiline
      Author : Mohammad Mohtashim
      Purpose: Multiline Comments Demo
      Subject: PHP
   */

   print "An example with multi line comments";
?>
```

PHP is whitespace insensitive

Whitespace is the stuff you type that is typically invisible on the screen, including spaces, tabs, and carriage returns (end-of-line characters).

PHP whitespace insensitive means that it almost never matters how many whitespace characters you have in a row.one whitespace character is the same as many such characters.

For example, each of the following PHP statements that assigns the sum of 2 + 2 to the variable $four is equivalent −

```
$four = 2 + 2; // single spaces
$four <tab>=<tab2<tab>+<tab>2 ; // spaces and tabs
$four =
2+
2; // multiple lines
```

PHP is case sensitive

Yeah it is true that PHP is a case sensitive language. Try out following example −

```
<html>
   <body>

      <?php
         $capital = 67;
         print("Variable capital is $capital<br>");
         print("Variable CaPiTaL is $CaPiTaL<br>");
      ?>

   </body>
</html>
```

This will produce the following result −

```
Variable capital is 67
Variable CaPiTaL is
```

Statements are expressions terminated by semicolons

A *statement* in PHP is any expression that is followed by a semicolon (;).Any sequence of valid PHP statements that is enclosed by the PHP tags is a valid PHP program. Here is a typical statement in PHP, which in this case assigns a string of characters to a variable called $greeting −

```
$greeting = "Welcome to PHP!";
```

Expressions are combinations of tokens

The smallest building blocks of PHP are the indivisible tokens, such as numbers (3.14159), strings (.two.), variables ($two), constants (TRUE), and the special words that make up the syntax of PHP itself like if, else, while, for and so forth

Braces make blocks

Although statements cannot be combined like expressions, you can always put a sequence of statements anywhere a statement can go by enclosing them in a set of curly braces.

Here both statements are equivalent −

```
if (3 == 2 + 1)
   print("Good - I haven't totally lost my mind.<br>");

if (3 == 2 + 1) {
   print("Good - I haven't totally");
   print("lost my mind.<br>");
}
```

Running PHP Script from Command Prompt

Yes you can run your PHP script on your command prompt. Assuming you have following content in test.php file

```
<?php
   echo "Hello PHP!!!!!";
?>
```

Now run this script as command prompt as follows −

```
$ php test.php
```

It will produce the following result −

```
Hello PHP!!!!!
```

Hope now you have basic knowledge of PHP Syntax.