PHP - $ and $$ Variables

We know that PHP uses the convention of prefixing the variable names by the "$" symbol. PHP also has the provision of declaring dynamic variables by prefixing two dollar symbols ($$) to the name. A variable variable (or a dynamic variable) can be set and used dynamically.

The declaration of a normal variable is like this −

```
$a = 'good';
```

A dynamic variable takes the value of a normal variable and treats that as the name of the variable. In the above example, "good" can be used as the name of a variable by using two dollar signs "$$" −

```
$$a = 'morning';
```

We now have two variables: "$a" with contents "good" and "$$a" with contents "morning". As a result, the following echo statements will produce the same output −

```
echo "$a {$$a}";
echo "$a $good";
```

Both produce the same output −

```
good morning
```

Example 1

Take a look at this following **example** −

Open Compiler

```
<?php
   $a = 'good';
   $$a = 'morning';

   echo "$a {$$a}\n";
   echo "$a $good";
?>
```

It will produce the following **output** −

```
good morning
good morning
```

Example 2

Let's take a look at another example −

</> Open Compiler

```
<?php
   $x = "foo";
   $$x = "bar";
   echo "Value of x = " .$x . "\n";
   echo 'Value of $$x = ' . $$x . "\n";
   echo 'Value of foo = ' . $foo;
?>
```

Here, you will get the following **output** −

```
Value of x = foo
Value of $$x = bar
Value of foo = bar
```

Using Multiple "$" Symbols

Note that the use of "$" symbol is not restricted to two. Any number of dollar symbols can be prefixed.

Suppose there is a variable "$x" with "a" as its value. Next, we define $$x='as', then "$$x" as well as "$a" will have the same value. Similarly, the statement $$$x='and' effectively declares a "$as" variable whose value is 'and'.

Example

Here is a complete example that shows the use of multiple "$" symbols.

Open Compiler

```
<?php
   $php = "a";
   $lang = "php";
   $World = "lang";
   $Hello = "World";
   $a = "Hello";
   echo '$a= ' . $a;
   echo "\n";
   echo '$$a= ' . $$a;
   echo "\n";
   echo '$$$a= ' . $$$a;
   echo "\n";
   echo '$$$$a= ' . $$$$a;
   echo "\n";
   echo '$$$$$a= ' . $$$$$a;
?>
```

When you run this code, it will produce the following **output** −

```
$a= Hello
$$a= World
$$$a= lang
$$$$a= php
$$$$$a= a
```

Using Dynamic Variables with Arrays

Using dynamic variables with arrays may lead to certain ambiguous situations. With an array "a", if you write $$a[1], then the parser needs to know if you are refering to "$a[1]" as a variable or if you want "$$a" as the variable and then the [1] index from that variable.

To resolve this ambiguity, use ${$a[1]} for the first case and ${$a}[1] for the second.

Example

Take a look at the following example −

```php
<?php
   $vars = array("hw", "os", "lang");
   $var_hw="Intel";
   $var_lang="PHP";
   $var_os="Linux";

   foreach ($vars as $var)
      echo ${"var_$var"} . "\n";

   print "$var_hw\n$var_os\n$var_lang";
?>
```

It will produce the following **output** −

```
Intel
Linux
PHP
Intel
Linux
PHP
```

It may be noted that this technique cannot be used with PHP's Superglobal arrays (Several predefined variables in PHP are "superglobals", which means they are available in all scopes throughout a script) within functions or class methods. The variable "$this" is a special variable in PHP and it cannot be referenced dynamically.