Scope of variables

Local :A variable declared within a PHP function is local and can only be accessed within that **function.**

- ✓ The echo statement refers to the local scope variable $x, which has **not** been **assigned a value within this scope**.
- ✓ Local variables are only **recognized** by the function in which they are declared & are **deleted** as function is completed
- ✓ Local variables with the s**ame name in different functions** is possible

```
<?php
  $x=5;  //global scope
      function myTest()
                  {
echo $x; //local scope
}
myTest();   ?>
```

- ✓ Global A variable that is defined **outside** of any function, has a global scope.
- ✓ It can be accessed from **any part** of the script, **EXCEPT** from within a **function**.
- ✓ To access a global variable from within a function, use the **global keyword**.
- ✓ It also stores all global variables in an array called **$GLOBALS[*index*]**

- ✓ The *index* holds the **name of the variable**

- ✓ This array is also **accessible** from within **functions** and can be used to **update** global variables **directly**

GLOBAL

```
<?php
$x=5; // global scope
$y=10; // global scope

function myTest()
{
global $x,$y;                    //First way to access a global variable through global keyword
$y=$x+$y;
}

myTest();
echo $y;?>
```

```
<?php
$x=5;
$y=10;

function myTest()
{
$GLOBALS['y']=$GLOBALS['x']+$GLOBALS['y'];// way to access a global
variable
}

myTest();
echo $y;
?>
```

- ✓ Static When a function is completed, all of its variables are normally deleted.
  - ✓ you want a local variable to **not be deleted**
  - ✓ each time the **function is called**, that variable **will still have the information** it contained from the last time the function was called

Parameter

- ✓ Parameter is a local variable whose value is **passed to the function** by the calling code.

```
function myTest($x)
{
echo $x;
}

myTest(5);
```

Super global variables

Super global variables are built-in variables that are always available in all scopes. Several predefined variables in PHP are "superglobals", which means they are available in all scopes throughout a script. There is no need to do global $variable; to access them within functions or methods. Examples of super global variables are: *$GLOBALS, $_GET ,$_POST ,$_REQUEST etc.*

```
<?php
$x=5;
$y=10;

function myTest()
{
$GLOBALS['y']=$GLOBALS['x']+$GLOBALS['y'];
```

//GLOBALS array can directly be used without declaring in the function
}

Variable variables

PHP enables us to have "variable" variable names i.e. a variable name can be set and used dynamically. It creates a new variable where the first variable's value act as the name of the new variable. Variable variables, concept can be used to access the contents of a variable without knowing its name

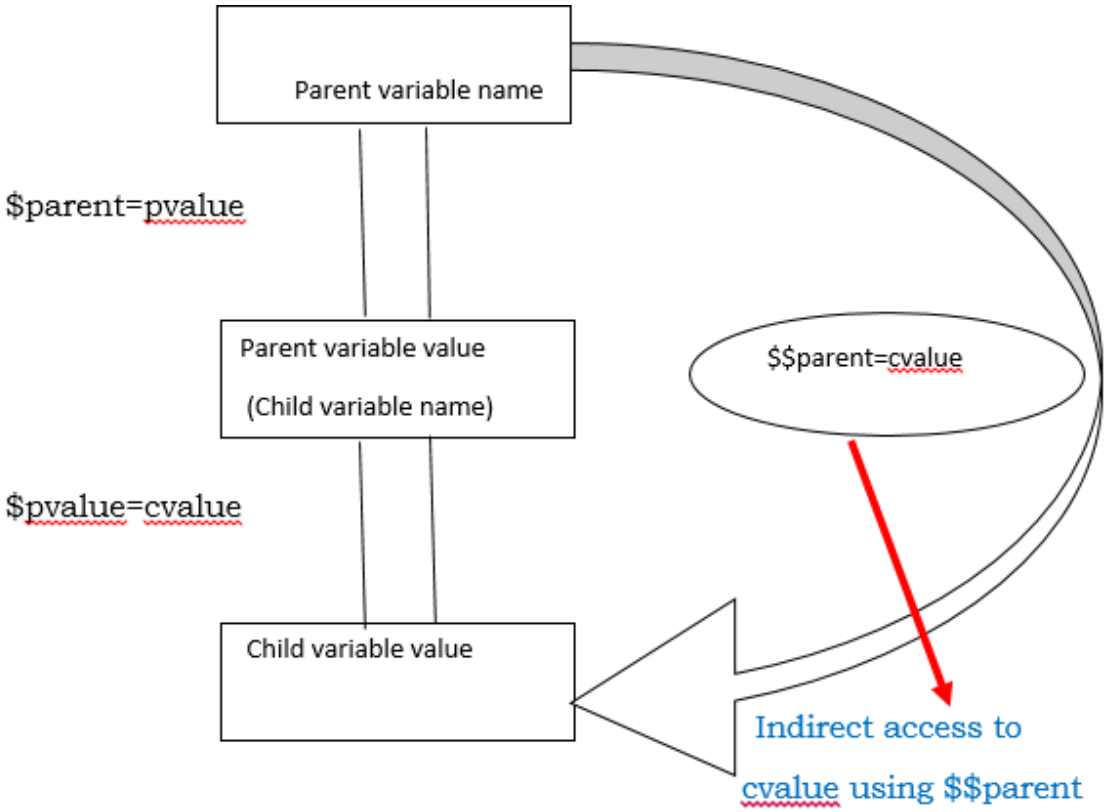

Figure 1 : "Variable Variables" working Structure

```
$a="hello";          //Parent variable assigned value
$hello=25;           //Child variable assigned value
echo $a;             //hello
echo $hello;         //25
//Using "variable variables"
```

```
$a="hello";          //Parent variable assigned value
$$a=25;              //Child variable assigned value
echo $a;             /*Parent value/Child variable name displayed i.e
                     hello*/
echo $$a;            /*Child value  displayedusing $$ sign prefixed with
                     parent variable i.e.25*/
```

References

References in PHP are a means to access the same variable content by different names. They are not like C pointers; for instance, you cannot perform pointer arithmetic using them, they are not actual memory addresses, and so on. Basically, references are alias names of a variable's content.

```
$a=5;
$b=&$a;
echo $a;          //5
echo $b;          //5
$b=6;             //This assignment  will  also change the value of "a" variable
echo $a;          //6
echo $b;          //6
unset($a);        //Will only unset "a",no change in "b"
echo $a;          // error because of  undefined symbol
echo $b;          //6
```

For pass and return by refrence go to page no__

WAYS TO SEND OUTPUT TO THE BROWSER.

- ✓ The **echo** constructs lets you print many values at once

```
echo (Printy);          echo "First", "second", "third"
```

- ✓ **print()** prints a string
- ✓ The **printf()** function builds a formatted string by inserting values into a template.

```
printf('%.2f', 27.452);
```

- ✓ The **print_r()** function is useful for debugging—it prints the contents **of arrays, objects, and other things**, in a more-or-less human-readable form.
  - intelligently displays what is passed to it, r**ather** than casting **everything to a string**, as echo and print() do

```
$a = array('name' => 'Fred', 'age' => 35, 'wife' => 'Wilma');

print_r($a);
```

Difference between print, printf, echo

1. **printf —** Output a formatted string and returns the length of the outputted string.
2. **print:** Output a string and returns *1*, always.
3. **echo:** Output one or more strings.
4. **print_r:** Prints human-readable information about a variable and also used for outputting given string.