PHP Output and Variables - Written by Anthony Adams

Output

Using PHP we can get data from various places and display it on the front end of a web page. This can be done in a couple different ways. Both ways are very similar but have small technical differences. These methods are echo and print statements.

No	Echo	Print
1	echo is a statement i.e used to display the output. it can be used with parentheses echo or without parentheses echo.	Print is also a statement i.e. used to display the output. it can be used with parentheses print( ) or without parentheses print.
2	echo can pass multiple string separated as (,)	using print can doesn't pass multiple argument
3	Echo doesn't return any value	Print always return 1
4	Echo is faster than print.	It is slower than echo
5	In PHP, echo is not a function but a language construct.	print is not a really function but a language construct. However, it behaves like a function in that it returns a value.

Echo

We can use the echo statement to return data on the page:

```
<?php
   echo "Ayeeee!";
?>
```

It can also be written with parenthesis like this:

```
<?php
   echo ("Ayeeee!");
?>
```

It can also be written with single quotes like this:

```
<?php
   echo 'Ayeeee!';
?>
```

Print

We can take the above examples and substitute echo for print:

```
<?php
   print 'Ayeeee!';
?>
```

It can also be written with parenthesis like this:

```
<?php
   print ("Ayeeee!");
?>
```

It can also be written with single quotes like this:

```
<?php
   print 'Ayeeee!';
?>
```

As you can see they both work the same. For the most part however we will use echo.

Variables

What is a variable? It's just a container!

We can use variables like a memory container. It will hold some kind of information in it.

In PHP a variable starts with cash 💰 $. So to make a variable all we have to do is put a $ sign and a name.

```
<?php
   $name = "Anthony";
?>
```

Once we declare the variable and tell it what it is, it is "saved" to memory and can be retrieved later.

```
<?php
   echo $name;
?>
```

This will return the variable we entered

The name below is the result of echoing out a variable

Anthony

We can use variables to store numbers too. Int he example below, x holds the value of 10. Y holds the value of 12 and we can echo the addition of the 2 numbers to output 22 on the web page.

```
<?php
   $x = 10;
   $y = 12;

   echo $x + $y;
?>
```

Variable names should be descriptive. Similar to in HTML where we name div's with a classname that is descriptive of what it is. Same deal here! In the case of PHP variables though, we usually write it in snake case (like_this) or camel case (likeThis).

The $ declares that its a variable

Variable names (after the $) must start with a letter or an underscore. It cannot start with a number.

Variable names can contain alpha numeric characters and _'s: (a-Z 0-9 and _).

Variable names are case sensitive.

Arrays

An array is just a variable with multiple values.

To make an array all we need to do is declare a variable:

```
<?php
   $cars = array("genesis", "Porsche", "Lambo", "BMW");
?>
```

Above we declare the $variable cars = to be an array and list off the cars (“ ”, “ ”, “ ”, “ “);

When we list an array the number of the entries

goes: 0, 1, 2, 3 and on.

We can echo one of the entries in the array:

```
<h2>We will select something from an array and echo it</h2>

<?php
     $cars = array("genesis", "Porsche", "Lambo", "BMW");
?>

<h1>
    I like <?php echo  $cars[0]; ?>
</h1>
```

This will return:

We will select something from an array and echo it

I like genesis

We can also output the total amount of the entries in our array:

```
<?php
   echo count($cars);
?>
```

There is a lot more advanced things we can do with arrays. After understanding the basics you can expand on it here:

https://www.w3schools.com/php/php_ref_array.asp

Global Variables (Super Globals)

Super globals in PHP are built in, predefined variables. There are various global variables and they all offer some different functionality. All of these global variables use the $_GLOBALVARIABLENAME. We will explain them all one by one. Keeping in mind that for super globals that returns information, we will need to echo them out to actually display the information.

$GLOBALS

This allows you to access global variables from anywhere in the PHP script. This includes from within functions or methods.

$_SERVER

This variable holds information about the server.

```
          <?php
                //Below are server variables returning
information on the server. The . '<br>' at the end of each line
is simply concatenating another string to the end of each line
and passing a br tag through to break up each line.
                echo $_SERVER['PHP_SELF'] . '<br>'; //Returns
the filename of the currently executing script .
                echo $_SERVER['GATEWAY_INTERFACE'] .
'<br>';    //Returns the version of the Common Gateway Interface
(CGI) the server is using
                echo $_SERVER['SERVER_ADDR'] . '<br>';  //
Returns the IP address of the host server
                echo $_SERVER['SERVER_NAME'] . '<br>';  //
Returns the name of the host server (such as www.w3schools.com)
                echo $_SERVER['SERVER_SOFTWARE'] . '<br>';  //
Returns the server identification string (such as Apache/2.2.24)
                echo $_SERVER['SERVER_PROTOCOL'] . '<br>';  //
Returns the name and revision of the information protocol (such
as HTTP/1.1)
                echo $_SERVER['REQUEST_METHOD'] . '<br>';   //
Returns the request method used to access the page (such as
POST)
```

```
                echo $_SERVER['REQUEST_TIME'] . '<br>'; //
Returns the timestamp of the start of the request (such as
1377687496)
                echo $_SERVER['QUERY_STRING'] . '<br>'; //
Returns the query string if the page is accessed via a query
string
                echo $_SERVER['HTTP_ACCEPT'] . '<br>';  //
Returns the Accept header from the current request
                echo $_SERVER['HTTP_ACCEPT_CHARSET'] .
'<br>';  //Returns the Accept_Charset header from the current
request (such as utf-8,ISO-8859-1)
                echo $_SERVER['HTTP_HOST'] . '<br>';    //
Returns the Host header from the current request
                echo $_SERVER['HTTP_REFERER'] . '<br>'; //
Returns the complete URL of the current page (not reliable
because not all user-agents support it)
                echo $_SERVER['HTTPS'] . '<br>';    //Is the
script queried through a secure HTTP protocol
                echo $_SERVER['REMOTE_ADDR'] . '<br>';  //
Returns the IP address from where the user is viewing the
current page
                echo $_SERVER['REMOTE_HOST'] . '<br>';  //
Returns the Host name from where the user is viewing the current
page
                echo $_SERVER['REMOTE_PORT'] . '<br>';  //
Returns the port being used on the user's machine to communicate
with the web server
                echo $_SERVER['SCRIPT_FILENAME'] . '<br>';  //
Returns the absolute pathname of the currently executing script
                echo $_SERVER['SERVER_ADMIN'] . '<br>'; //
Returns the value given to the SERVER_ADMIN directive in the web
server configuration file (if your script runs on a virtual
host, it will be the value defined for that virtual host) (such
as someone@w3schools.com)
                echo $_SERVER['SERVER_PORT'] . '<br>';  //
Returns the port on the server machine being used by the web
```

```
server for communication (such as 80)
                echo $_SERVER['SERVER_SIGNATURE'] . '<br>'; //
Returns the server version and virtual host name which are added
to server-generated pages
                echo $_SERVER['PATH_TRANSLATED'] . '<br>';  //
Returns the file system based path to the current script
                echo $_SERVER['SCRIPT_NAME'] . '<br>';  //
Returns the path of the current script
                echo $_SERVER['SCRIPT_URI'] . '<br>';   //
Returns the URI of the current page
            ?>
```

https://www.w3schools.com/php/php_superglobals_server.asp

$_SESSION

When you use any computer, the computer knows who you are. You login and use an application, do what you need to do and then close the application and log off. This is like a session. HTTP addresses do not maintain state and therefore do not know who you are. We can use a $_SESSION variable to allow people to login. This global variable is usually use for SENSITIVE DATA (Like a login).

A session ends when the browser is closed.

```php
<?php
    //This creates a session variable
    $_SESSION['user_name'] = "007";
    //echo out the user number
    echo $_SESSION['user_name'];

    if(!isset($_SESSION['user_name'])) {
        echo ' You are not logged in!';
        } else {
        echo ' You are logged in!';
    };

?>
```

$_COOKIE

Saves NON SENSITIVE DATA on the users end. For example, we could use a cookie to save the kind of electronics that a user shops for on a website, which we can use later to suggest certain things to the specific user that they might want to buy.

A cookie can be set to have a certain time limit.

```php
<?php
    //Sets the cookie(name, value, time() + a
```

```
day); time() means right now.
    setcookie("name", "User007", time() +
87000);
    //Here we just echo out the name of the
cookie which in this case is User007
    echo $_COOKIE['name'];
?>
```

$_POST

Used to send data from a form. The POST method will not display the data in the URL bar (Better for security). However the data could be echoed out.

```
<form method="POST">

    <fieldset>
        <legend>Personal Information</legend>

        <label for="fname">First Name</label>
        <br>
        <input type="text" id="fname"
name="f_name">
        <br>
        <input type="submit">
    </fieldset>

</form>

<?php
    //This will echo out the value entered on
```

```
the front end of whatever input which has the
name attribute of "f_name".
    echo $_POST['f_name'];
?>
```

$_GET

This method passes information though the URL bar.

```
<form method="GET">

                <fieldset>
                    <legend>Hardware
Information</legend>

                    <label for="phone">What
brand of phone do you use?</label>
                    <br>
                    <input type="text"
id="phone" name="phone_brand">
                    <br>
                    <input type="submit">
                </fieldset>

            </form>

            <?php
                //This will echo the result but
also note that the result will also be in the
URL bar. This will happen regardless of if we
echo out the result!
```

```
                echo $_GET['phone_brand'];
            ?>
```