Basic notes about PHP

Best Practices:

1. Use <?php ?> when we want to execute a php script.
2. A closing tag is required when mixing PHP and HTML.
3. Use {} for separation purpose if we want to execute a variable in a string.
4. Follow same naming style for function in whole script.
5. It is good practice to before accessing a function declare the function first.
6. Always return statement inside function.
7. Required argument is placed in first when we call the function.
8. In variable names use camelCase: take advantage of fact variables are case sensitive. E.g. $firstName.
9. Functions always return a value.
10. Return allows us to quickly access the function.

Remainder:

1. String in database must be enclosed in single quote otherwise data will not insert in database.
2. Pass related id of a dependency topic in url or by the hidden field must be.

Notable features of PHP

* &nbsp; none breaking space
* If we want to use a php code in html we must use echo to see the output.
* White space in php is does not matter
* Tab, new line and white space all are considered as white space and php does not care about whitespace
* File extension is must be php because if we write index.html then apache does not process the file but if we write index.php then apache look the file and assume php codes are inside and process the code and give the desire result to the user.
* If we want to use a variable with a string without separation then we use curly brackets {} to separate the string.
* PHP scripts are executed on the server.

Php Comments

* Php has mainly three types of comments e.g.

- Single line comment - #, //

- Multi line comment - /\* \*/

Variables:

* Variables are labels for memory locations, used to store values which can change.
* Variable names start with a “$” and can use camelCase to make the name descriptive.
* The value of a variable is the value of its most recent assignment.

PHP variables can be one of four scope types:

* Local variables
* Function parameters
* Global variables
* Static variables.

Global Variables:

* Global variable can be accessed in any part of the program
* Placing the keyword GLOBAL in front of the variable that should be recognized as global.
* Global means import it.
* If we want to use a global variable inside a function then we should explicitly declare the variable as a global. Example in below:

$a = 10;

function amount\_of\_money(){

global $a;

echo $a;

}

Amount\_of\_money();

* Even if the global variable $a didn’t exists before the function then global means ok creating them in the global scope and also referring them in the global scope and give the opportunity to access and using them.

PHP Data Types

* Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

* String
* Integer
* Float (floating point numbers - also called double)
* Boolean
* Array
* Object
* NULL
* Resource

PHP Object:

* An object is a data type which stores data and information on how to process that data

PHP NULL Value:

* Null is fancy terms for nothing not zeros or not has a value.
* Represents a variable with no value. NULL is the only possible value of type null.

A variable is considered to be null if:

* It has been assigned the constant NULL.
* It has not been set to any value yet.
* It has been unset().
* Only one value of type null, and that is the case-insensitive constant NULL.
* $var = NULL;
* Variable that has been assigned NULL has the following properties:
  + - It evaluates to FALSE in a Boolean context.
    - It returns FALSE when tested with IsSet() function

The Switch Statement:

* If you want to select one of many blocks of code to be executed, use the Switch statement.
* The switch statement is used to avoid long blocks of if..elseif..else code.

PHP 5 Arrays:

* “Collection of different variables under the same label to keep values organized and easily accessible for processing”
* An array stores multiple values in one single variable.

Example: $required\_fields = array("menu\_name","position","visible");

* An array is a special variable, which can hold more than one value at a time.
* An array can hold many values under a single name, and you can access the values by referring to an index number.
* In PHP, there are three types of arrays:
  + - Indexed arrays - Arrays with a numeric index
    - Associative arrays - Arrays with named keys
    - Multidimensional arrays - Arrays containing one or more arrays
* PHP Indexed Arrays

The index can be assigned automatically (index always starts at 0).

* PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them

PHP Loop Types:

* Loops in PHP are used to execute the same block of code a specified number of times.

PHP supports following four loop types.

* for - loops through a block of code a specified number of times.
* while - loops through a block of code if and as long as a specified condition is true.
* do...while - loops through a block of code once, and then repeats the loop as long as a special condition is true.
* foreach - loops through a block of code for each element in an array.
* Foreach loop is used for arrays to take each next value from array and perform whatever action is required on it.

The foreach loop statement:

* The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax:

foreach (array as value)

{

code to be executed;

}

Most important String Functions in PHP a programmer should know

1. substr()

Some useful functions of PHP

1. isset()

* isset() is used to determine whether a variable is set or not.
* Determine if a variable is set and is not NULL
* It returns true only when the variable is not null.
* ISSET returns TRUE if the variable exists and has a value other than NULL.
* That means variables assigned a " ", 0, "0", or FALSE are set, and therefore are TRUE for ISSET.

1. is\_null()

* Finds whether a variable is NULL
* It returns true only when the variable is null.
* is\_null() is opposite of isset(), except for one difference that isset() can be applied to unknown variables, but is\_null() only to declared variables.

1. empty()

* Determine whether a variable is empty
* It will return true if the variable is an empty string e.g. " ", false, array()(an empty array), NULL, 0.0(0 as float), 0(0 as an integer), and an unset variable and "$var;" (a variable declared, but without a value in a class) and "0" (0 as a string).
* Empty values are: “”, null, 0, 0.0, “0”, false, array ().

1. Nl2br()

* Insert line breaks where newlines (\n) occur in the string.
* Example- <?php echo nl2br("One line.\nAnother line."); ?>
* Browser Output: One line.

Another line.

* View Source: One line.<br />

Another line.

* The nl2br() function inserts HTML line breaks (<br> or <br />) in front of each newline (\n) in a string.

1. Unset

* Unset () is used to destroy a variable in PHP. In can be used to remove a single variable, multiple variables, or an element from an array. It is phrased as Unset ($remove).
* Syntax - unset (var1, var2.... )
* Unset Global Variables - unset($GLOBALS['bar'])

1. Strip\_tags

* Strip HTML and PHP tags from a string.

1. print\_r to print an array because you cannot print an array with echo or print function (both are used to display output) though you can use echo or print to display single items from the array
2. heredoc allows the programmer create multi-line strings without using quotations.

There are a few very important things to remember when using heredoc.

* Use <<< and some identifier that you choose to begin the heredoc. In this example we chose TEST as our identifier.
* Repeat the identifier followed by a semicolon to end the heredoc string creation. In this example that was TEST;
* The closing sequence TEST; must occur on a line by itself and cannot be indented!
* Example: $my\_string = <<<TEST

Tizag.com <br/>

Webmaster Tutorials

Unlock your potential!

TEST;

1. Defined - Checks whether a given named constant exists.

* Checks whether the given constant exists and is defined.

1. Exit - Output a message and terminate the current script.
2. PHP The static Keyword:

* Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.
* In static variable each time the function is called, that variable will still have the information it contained from the last time the function was called.
* The variable is still local to the function.

<?php  
function myTest() {  
    static $x = 0;  
    echo $x;  
    $x++;  
}  
myTest();  
myTest();  
myTest();  
?>

Resources:

Are special variables that hold references to resources external to PHP (such as database connections)?

For the pass encountering continue statement, rest of the loop code is skipped and next pass starts.

The escape-sequence replacements are:

* \n is replaced by the newline character
* \r is replaced by the carriage-return character
* \t is replaced by the tab character
* \$ is replaced by the dollar sign itself ($)
* \" is replaced by a single double-quote (")
* \\ is replaced by a single backslash (\)

Here are the steps required to read a file with PHP.

* Open a file using fopen() function.
* Get the file's length using filesize() function.
* Read the file's content using fread() function.
* Close the file with fclose() function.

Conditional Operator: ? : Conditional Expression If Condition is true ? Then value X : Otherwise value Y

Differences between constants and variables are:

* There is no need to write a dollar sign ($) before a constant, where as in Variable one has to write a dollar sign.
* Constants cannot be defined by simple assignment, they may only be defined using the define() function.
* Constants may be defined and accessed anywhere without regard to variable scoping rules.
* Once the Constants have been set, may not be redefined or undefined.

Difference Between $var and $$var in Php:

* $$var uses the value of the variable whose name is the value of $var.
* It means $$var is known as reference variable whereas $var is normal variable.
* It allows you to have a “variable’s variable” – the program can create the variable name the same way it can create any other string.

<?php

$name="Rajeev";

$$name="Sanjeev";

echo $name."<br/>";

echo $$name."<br/>";

echo $Rajeev;

?>

Output : Rajeev

Sanjeev

Sanjeev

Printf()

* Printf() is used when you need precise output formatting.
* A format string includes text and placeholders which start with “%” format codes include %s %d and %.Nf where N = decimal places.

Echo/Print

* Print and Echo both used for printing purpose.
* Print method can return a true/false (1/0) value so it can be used in expressions.
* The echo statement can be used with or without parentheses: echo or echo().
* Echo does not return a value, but has been considered as a faster executed command.
* Example

$val = Print “LearnPHP7”;

Echo $val; //It will return 1, because it assumes the print work, since ‘print’ returns true/false (1/0) value, so it prints value 1, not LearnPHP7

* Using “echo” we can print multiple values but it is not possible with “print”.
* Print(‘I am at’,’LearnPHP7’);
* Echo(‘I am at’,’LearnPHP7);
* Comma(,) will not work in ‘print’ but works in ‘echo’
* Print function takes only one string at a time while as echo function takes more string at a time.
* Echo function is faster than print function.
* Print "Multiple things " . $on . " one line";
* //Outputs a string, then a variable, then a string. All are separated with a [.] period
* Same output: Print ("Multiple things " . $on . " one line");
* The print statement can be used with or without parentheses: print or print().
* Echo function is faster than print function.

Combination Operators:

* These operators combine arithmetic with assignment such as: += -= \*= /=
* Represents a shorter way of representing a simple arithmetic operation. Example: +=

Single Quotation and Double Quotation

* Single Quotation: it will print the whole statement in output not parse the line.

- Double Quotation: parse each line and print the value of the variable.

Most Popular Array Functions in PHP

Count() - How many elements are present in the array.

Max() - Determine the maximum value from the array.

Min() - Determine the minimum value from the array.

Sort() - Sort the array in ascending order.

Rsort() - Sort the array in descending order.

Implode() - Turn an array into string in other words combined values together to get a string

Explode() -

* Takes a string every time it takes the user divider as a new object in array
* Explode split the string in divider place the object as a new element in array
* Explode is extremely useful in comma separator list

in\_array() - determines whether an element is present in array or not.

Type Juggling:

* PHP converted one type to another type for us. For example if we want to add an integer with a string php converted the string in integer return the Boolean true and returns the value. This is referred as type juggling.

Type Casting:

* Convert from one type to another type.

Type casting is done in two ways:

- settype($var, “integer”) – permanent type conversion

- (Integer) $var – instant type convertion

Type Casting:

1. More often we used two types of function for type casting one is for set the type named SETTYPE() - Conversion is done in place and another one is GETTYPE() to get the data type.
2. Custom type conversion is done if we assigned the conversion in a variable.
3. Custom type conversion is not a permanent conversion.

Constants:

1. Variable can change but constant can never change.
2. Constants are defined with the define() command
3. As a convention, constants use UPPER\_CASE
4. Constants can be assigned any of the 4 basic data types
5. Constants can be assigned to variables
6. Constants can be echoed
7. Predefined constants are assigned when php is installed

Magic Constants:

1. The value of a magic constant is defined when our php program starts running.
2. Magic constants start and end with "\_\_"

If Statement:

1. Always use curly brackets if even statement is in a single line it increases more readability.
2. Always apply indent ability.

Continue

1. Continue says as like go to the next element.
2. Is more useful if we want to skip several statements in if condition.
3. Continue (1) 🡪it defines the first loop or it denotes the loop from where it is originated.

Continue (2) 🡪it defines the previous loop or parent of the continuous loop.

1. Break is also same as the continue loop at this point.

Array Pointers:

1. Is more useful to work with databases.

Array Pointers are

Current() - current pointer value

Next() - move the pointer forward

Previous - prev() - move the pointer backward

Reset() - move the pointer to the first element

End() - move the pointer to the last element

Argument Advantages:

* Arguments give flexibility and reusability of the code.

List()

* List is the powerful tool to pack a variable and immediately unpack the variable.

White screen of death:

* If your website is just blank, then typically a syntax error is the cause.
* Enable their display with:

error\_reporting = E\_ALL

display\_errors = 1

Common Problems:

1. No output at all

- Try to access an HTML page

- Make sure web server is running

- If this working then we

- Try to access a PHP page

- For confirming php is working run following code.

- <? php phpinfo(); ?>

- If above tests are successful then problems is probably in our code.

- Make sure display errors are on and configured.

1. Most common php errors

🡪Typos: misspelled variable name; commented out code.

🡪 Missing semicolons at end of line

🡪Missing closing brace :})]

🡪Missing closing quote: “‘

🡪Case-sensitive variable names: $myvar vs. $myVar

🡪= vs. ==

PHP Errors

* Fatal Errors

🡪PHP understood the code but could not execute it

🡪Most common cause of failure is php call an undefined class or function.

* Syntax Errors

🡪PHP could not understood or process the code.

🡪PHP could not understand what you want to do.

🡪This occurs mainly missing semicolon, mismatch variable names, Typo, quotation mark, parenthesis etc.

🡪This type of error is like that unexpected error.

* Warnings

🡪PHP found a problem, but was able to recover.

🡪Non failure

🡪Warnings occur in case of mathematical operation, incorrect number of arguments, and incorrect path to a file or you have don’t permission to access the file.

* Notices

🡪PHP is offering advice.

🡪Something smells bad.

🡪Good indicator of bug or sloppy programming.

1. Debugging and Troubleshooting

* Echo $variable //variable value
* Print\_r($array) //print readable array
* Gettype($variable) //variable type
* Var\_dump($variable) //variable type and value
* Get\_defined\_vars(); //array of defined variables
* Debug\_backtrace(); //show backtrace

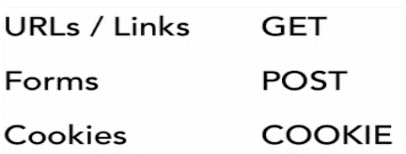
Brief about Debugging and Troubleshooting:

* Echo 🡪 The echo command can also be useful in if statements, foreach statements, functions, and so on, to ensure that these loops are being called or processed correctly.
* The PHP var\_dump() function returns the data type and value.

Links and Urls

1. There are three ways to get information or data from users

* URLs/Links 🡪 GET request
* Forms 🡪POST ||
* Cookies 🡪COOKIE



1. COOKIE is not a request but it’s the way to access the cookie request.
2. Browser cookies restored on browser.

Query String

* One of important things is that we can send lot of dynamic data by sending something query string.
* never use space between query string variable and = sign.
* We can pass data from one page to another by using query string.
* Query string comes after page name e.g.

<a href="second\_page.php**?id=1**"><?php echo $link\_name;?></a>

Query Parameter

* Query parameter is part of the url it’s come after the question mark e.g. somepage.php?page=2
* If we want to pass more than one variable via url then we use & percent like somepage.php?category = 7 & page=2
* Real time example of query parameter is like that <http://google.com/search?q=php>

Super global variable

* “Super global” for short
* Always available in all scopes
* Assigned by PHP before processing page code
* There are 9 super global variables all together
* $\_GET is an httpd method is used to pass value to the URL.

GET Variable:

* All of the URL data would be an associative array in GET variable.

POST Variable:

* All of the form data would be an associative array in post variable.
* You should use $\_GET when someone is requesting data from your application.
* And you should use $\_POST when someone is pushing (inserting or updating; or deleting) data to your application.

Difference between GET and POST:

* Post value is not seen in the URL that’s why post is secure.
* Get value is seen in the URL that’s why get is unsecure.
* $\_GET is used to request data from the server, while $\_POST is used to send data to the server.

URL encode:

* Letters, numbers, underscore, and dash are unchanged.
* Reserved characters become % + 2-digit hexadecimal.
* Spaces become “+”

RawUrlEncode:

* Letters, numbers, underscore, and dash are unchanged.
* Reserved characters become % + 2-digit hexadecimal.
* Spaces become “%20”

UrlEncode vs. RawUrlEncode:

* Rawurlencode the path

-Path is the part before the “?”

-Spaces must be encoded as %20

* Urlencode the query string

-Query string is the part after the “?”

-Spaces are better encoded as “+”

* Rawurlencode is more compatible generally
* URLs use letters, numbers, underscore and dash but some characters are not used in the urls which is known as reserved characters in urls which have special meaning in urls.
* We encode the reserved characters from urls without any affecting the function in url.
* Encoding is just for GET request encode is used for url encoding.
* Everything in left question mark is used rawurlencode() and everything is right in question mark is used urlencode().

Encoding for HTML:

* HTML has some reserved words like < > & “ which has a special meaning in html but if we want to use this in browser exactly in same way then we need to encode them.
* In PHP it is done in two ways:
* Htmlspecialchars() 🡪encoding only 4 reserved chars.
* Htmlentities() 🡪encoding all types of reserved chars and words like trademark issues, currency, reserved chars etc.

Require and Include

* Use require when the file is required by the application.
* Use include when the file is not required and application should continue when file is not found.
* When a file is included with the include statement and PHP cannot find it, the script will continue to execute.
* If we do the same work using the require statement, the echo statement will not be executed because the script execution dies after the require statement returned a fatal error.

Include and require statements are identical, except upon failure:

* require will produce a fatal error (E\_COMPILE\_ERROR) and stop the script
* include will only produce a warning (E\_WARNING) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

Including and Requiring Files:

* If we need same code in a file which is located in another file then we can add the file or code of the file in our present file by using include function which is much better instead of copying code because copying code sometimes arise some bugs or looking arrangement is decreased.
* Require() and include() are useful to reuse of codes.
* The only difference between require() and include is that require() generates fatal errors where include() constructs only warnings.
* Include() is useful in following areas:

Include

Functions🡪because we can define them once and only once like form functions, database functions, general functions etc.

Layout sections🡪header, footer, sidebar i.e. navigational enhancement etc.

Reusable HTML/PHP code🡪banner ads, page analytics, and little snippets like images.

CSS and JavaScript.

* Include() there are three variations:

Require()🡪is same thing as include but it raise fatal errors if the file can’t be found it says like that look really it required not able to go forward if can’t find this file. Include() doesn’t do that include will try to find the file but if won’t find the file it can go to forward any way that may cause fail in later we try to call a function or class or something which is undefined.

Include\_once()🡪keeps an array of files to the files that his already included so as an includes the files it just add the file path to an array if we asked to included file again no it will because it is seems that it is included before. This great to way including functions because can’t define functions more than once without getting an error.

An included file can be included only once.

Require\_once()🡪same.

This four functions well organized our code.

* HTTP:

HTTP is a stateless protocol. That means that when you load a page in your browser, and then you navigate to another page on the same website, neither the server nor the browser has any intrinsic way of knowing that it’s the same browser visiting the same site. Another way of saying this is that the way the Web works is that every HTTP request contains all the information necessary for the server to satisfy the request.

* This is a problem, though: if the story ended there, we could never “log in” to anything. Streaming media wouldn’t work. Websites wouldn’t be able to remember your preferences from one page to the next. So there needs be a way to build state on top of HTTP, and that’s where cookies and sessions enter the picture.
* The idea of a cookie is simple: the server sends a bit of information, and the browser stores it for some configurable period of time. It’s really up to the server what the particular bit of information is: often it’s just a unique ID number that identifies a specific browser so that the illusion of state can be maintained.

There are some important things you need to know about cookies:

* Cookies are not secret from the user
* The user can delete or disallow cookies
* Users will notice if you abuse cookies
* If you set a lot of cookies on your users’ computers, or store a lot of data, it will irritate your users, something you should avoid. Try to keep your use of cookies to a minimum.
* Cookies can be used for attacks
* One technique of XSS attacks involves malicious JavaScript modifying the contents of cookies.
* Prefer sessions over cookies

For the most part, you can use sessions to maintain state, and it’s generally wise to do so. It’s easier, you don’t have to worry about abusing your users’ storage, and it can be more secure. Sessions rely on cookies, of course, but with sessions, express will be doing the heavy lifting for you.

* Cookies are not magic: when the server wishes the client to store a cookie, it sends a header called Set-Cookie containing name/value pairs, and when a client sends a request to a server for which it has cookies, it sends multiple Cookie request headers containing the value of the cookies.
* HTTP Header

🡪Data of the database which is located in server.

🡪Headers are come before all others data.

🡪Changes must be made before

🡪Headers are the mechanism for use page redirection.

🡪Headers are most often need in page redirection.

🡪Redirection means able to send the user in another page.

🡪HTTP is the web standard which is able to redirect the browser in another page.

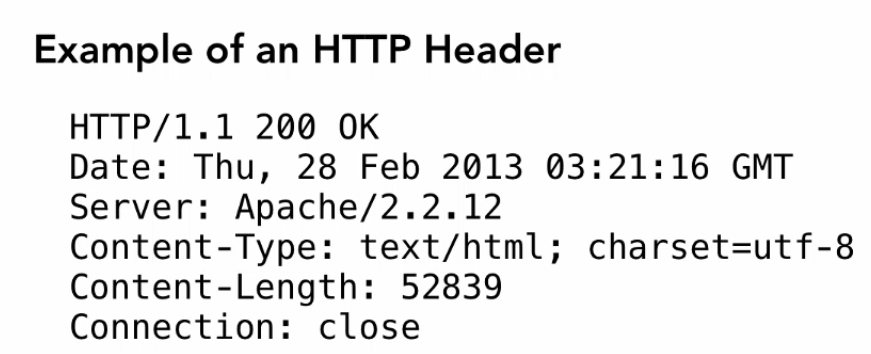
🡪302 Redirect

* HTTP 1.1/302 Found
* Location: path

🡪header(“Location: login.php”);

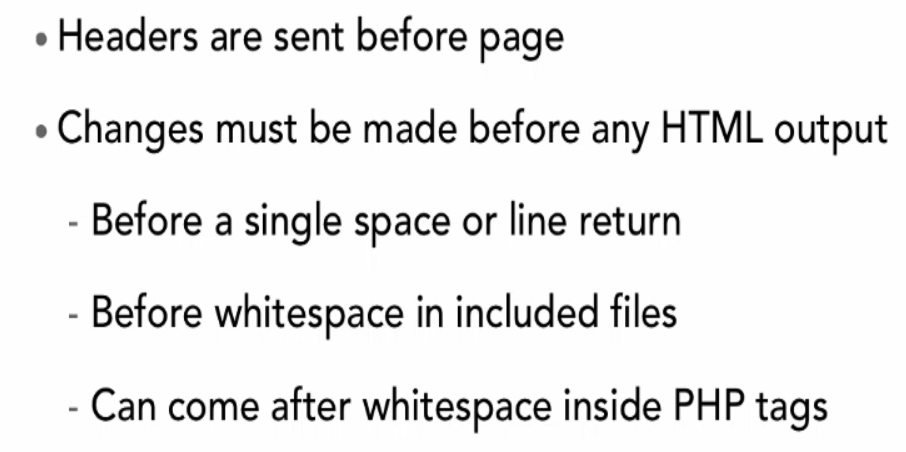
🡪Headers and redirection have to come before any output to the html.

🡪Redirect\_to() halt execution rest of the page



Form Tag: Get data from user / input data.

* PHP global variable store the data in server site.
* PHP global variables are post, get etc. work as like dictionary.
* Data in post variable as a key-value pair return the value which is submit into server site.



Output Buffering:

Output buffering is a mechanism for controlling how much output data (excluding headers and cookies) PHP should keep internally before pushing that data to the client. If your application's output exceeds this setting, PHP will send that data in chunks of roughly the size you specify. Turning on this setting and managing its maximum buffer size can yield some interesting side-effects depending on your application and web server. You may be able to send headers and cookies after you've already sent output through print or echo. You also may see performance benefits if your server is emitting less packets due to buffered output versus PHP streaming the output as it gets it. On production servers, 4096 bytes is a good setting for performance reasons. Note: Output buffering can also be controlled via Output Buffering Control functions.

Attribute: name – specifies the name of an <input> element.

Validating Form Values:

* If data passes validation that means data is acceptable we can use it.
* If data does not passes validation that means data is not acceptable.

Common Validations for Form Fields:

* User submits some value in form fields; form field can’t be left blank-i.e. Presence that means something present in form fields.
* String Length🡪Number of characters they submitted.
* Type🡪make sure syntaxes are int, char or float etc.
* Inclusion in a set🡪like radio button inputs.
* Uniqueness🡪e.g. username
* Format🡪like email contain @ symbol or not, date contains am or pm etc.
* Preg\_match (pattern, subject) {}//first give the pattern and second parameter is the subject.

**Common Problems with Validation:**

**Type juggling during Comparisons:**

* string vs. null: converts null to “ ”
* Empty string vs. null is going to equal.
* Boolean vs. other: converts other to Boolean
* Number vs. other: converts other to number.

Difference between Session and Post

* Sessions data stored on the server. Session ID is simply the key to a record which exists on the server, and this record contains whatever session data you write to it. This means that different PHP pages which provide the same session ID will connect to the same session data on the server. This is the way that one web page passes is state to another web page.
* $\_POST and $\_SESSION are totally different, and it is not a case of using one or the other. You will always use $\_POST when sending data from the client to the server, while the use of $\_SESSION is totally optional and invisible to the client.

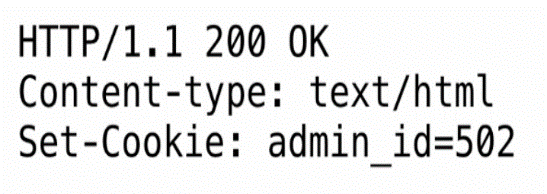
**Brief about Session & Cookie:**

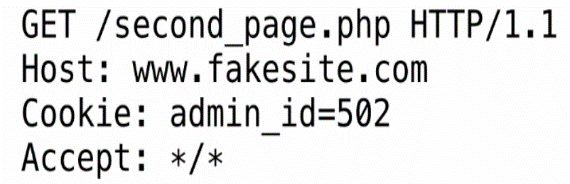
1. Sessions are terminated once the browser closed. Every time user closed the browser then session is closed and site is restored on the server.
2. On the other hand cookies are more permanent we can set the expiration time for the cookie. So that is stayed when the browser is closed as long as expiration date has passed.
3. A session is stored in a file in web server not on the web browsers.
4. We can use session data in order to pass data from one page to another in php in our web page.
5. Sessions are stored on server whereas cookie is stored on client.
6. Session has to come very beginning because it does two things: it either start a new session if the one is before
7. Session is most useful for:

* User authentication
* $logged\_in, $user\_id
* Holding data during a redirect
* $message, $errrors
* Frequently referred to data
* $username, $account\_type

PHP COOKIE:

* A way for data from the user using browser cookies
* Cookie is essentially a piece of information they gets stored in the client browser in that cookie we can stored certain information that we may find useful one of the most common ways of use cookies are used for logging system.
* A small bits of data that the website asks the browser to keep around
* Cookies are important because they give us the validate user’s state to remember who the user is and what they are doing without cookies web server don’t recognize multiple page requests from the same user.
* We want to know what ip address of the user, isn’t that the same, each request probably come from the same ip, more than one person share an ip address, in fact it is common in wireless network.
* Without cookies we don’t recognize multiple page request comes from the same user.
* Server locate the cookie for recognize that request come from the same browser that previous three request comes from the same browser.
* Cookies value are seen in the end user so anything that’s are critical don’t put them into cookie without encrypt it.
* Users has right to delete the cookie.
* PHP takes all cookie values and send it to the request header and puts them in associative array super global variable COOKIE
* COOKIEs hold the value from previous request it changed when set the cookie.
* Example



* COOKIE is a global variable which declared as follows: $\_COOKIE
* Setting COOKIE values:

setcookie($name, $value, $expire);

* $name is that we want to set for the particular cookie.
* $value is that value which is associated with that name.
* $expire tells the browser how long it keeps the cookie around. We can consider the cookie valid at certain day, time or year i.e. how long cookies valid.
* (60\*60\*24\*7) //60 secs, 60 mins, 24 hours, 7 days i.e. 1 week if add 2 week then write (60\*60\*24\*7\*2)
* Remember: Setting cookies must be before \*any\* HTML output unless output buffering is turned on.
* Always use single quote in mysql never use double quote and single is not essential for integer value.

COOKIE

* cookie: a small amount of information sent by a server to a browser, and then sent back by the browser on future page requests
* cookies have many uses:
* authentication
* user tracking
* Maintaining user preferences, shopping carts, etc.
* a cookie's data consists of a single name/value pair, sent in the header of the client's HTTP GET or POST request

SESSION

* Session is rely on cookie to perform their task.
* Session is file stored on the web server.
* The session is identified by a session-id, which is stored at the client and send with each request. Usually the session-id is stored in a cookie, but it can also be appended to urls. (That's the PHPSESSID query-parameter you sometimes see)
* on the client, the session ID is stored as a cookie with the name PHPSESSID
* On the server, session data are stored as temporary files such as /tmp/sess\_fcc17f071...
* you can find out (or change) the folder where session data is saved using the session\_save\_path function
* for very large applications, session data can be stored into a SQL database (or other destination) instead using the session\_set\_save\_handler function

Session Pros

* More storage where cookie is limited. Cookie support 1000 character where session is depend on file storage on the web server how big hard drive of the server that’s the limit.
* Smaller request sizes
* Conceals data values
* Session files accumulate

Session Cons

* Slower to access
* Expire when browser is closed

Web server:

* software that listens for web page requests
* Apache
* Microsoft Internet Information Server (IIS) (part of Windows)
* The web server contains software that allows it to run programs those are written in server side programming language like PHP, Java, ASP and send back their responses to web requests.

**URL Encoding:**

There is a set of characters that can't appear in a URL, and therefore by association, can't appear in a query string either, so they have to be URL encoded.

The encoding process requires you, the user or developer, to do precisely nothing. It's all done for you. The Web browser takes the offending character, whether a bracket or an addition sign, and replaces it with a code value (when sending to the server), and takes the URL encoded value and replaces it with the appropriate character (when displaying in your screen). The URL encoded value is always the same (for example, a blank space is always be represented by %20). The following table lists the most common characters and their code values.

The POST Value

One disadvantage you might have discerned from query strings is the rather public nature of their transmission. If you don't want the information sent to appear in the URL, then you will have to rely on the POST method instead. This works almost identically to the GET method; the difference is that the information in the form is sent in the body of the HTTP request, rather than as part of the URL. This means that it isn't visible to everybody, because it isn't attached to the URL. POST can also allow a greater amount of information to be transmitted. There is a physical limit to the amount you can transmit as part of a URL.

Do You Use GET or POST?

There's a mixture of opinion on this one; some people say you should almost never use the GET method because of its insecurity and limit on size; others maintain that you can use GET to retrieve information, but POST should be used whenever you modify data on the Web server. There are no hard and fast rules, though, and these are just guidelines.

One disadvantage of POST is that pages loaded with it cannot be properly bookmarked, while pages loaded with GET contain all the information needed to reproduce the request right in the URL. In many cases you can bookmark the result of a form submission (a search on Alta Vista, for example) by using the GET method, and that's why most search engines use GET. Another disadvantage of POST is that the method itself isn't secure—while the information is placed in the HTTP body and isn't immediately visible, the information isn't encrypted and is still easily obtained by a hacker. To make sure it is secure, you would need to use a secure connection to a secure server.

Which method you use depends on what you want the form to do. If you do use GET, be aware of its shortcomings and its indiscreet nature. If you use POST, beware that it can't be bookmarked by search engines, and just because it is more discreet doesn't mean it is more secure.

Function Exists

* Return **TRUE** if the given function has been defined
* Checks the list of defined functions, both built-in (internal) and user-defined, for **function\_name**.
* Returns TRUE if **function\_name** exists and is a function, FALSE otherwise.

Method Exists

* Checks if the class method exists
* Checks if the class method exists in the given object.

Directory Exists

* is\_dir() function checks whether the specified file is a directory.
* This function returns TRUE if the directory exists.

Get Object Vars

* Returns an associative array of defined object accessible non-static properties for the specified **object** in scope. If a property has not been assigned a value, it will be returned with a **NULL** value.

Encryption in PHP

* One-way Encryption: MD5
* CRYPT()
* SHA1()

Implode in PHP

* The implode() function returns a string from the elements of an array.
* implode — Join array elements with a string
* Example : $arr = array('Hello','World!','Beautiful','Day!');

echo implode(" ",$arr);

Output : Hello World! Beautiful Day!

Cookie

* A cookie is often used to identify a user.
* Creation of Cookie: setcookie()
* Syntax : setcookie(name, value, expire, path, domain, secure, httponly);

Overloading Method in PHP

* \_\_get, \_\_set, \_\_call, \_\_callStatic

PHP file upload-related functions

* is\_uploaded\_file()
* move\_uploaded\_file()

PHP magic Constants

* \_\_FUNCTION\_\_ return the function name.

|  |  |
| --- | --- |
|  |  |