print_r()

Prints human-readable information about a variable.

var_dump()

Displays structured information about one or more expressions that includes its type and value.

is_array()

Returns TRUE if var is an array, FALSE otherwise.

Example:

```
<?php
$yes = array('this', 'is', 'an array');
echo is_array($yes) ? 'Array' : 'not an Array';
echo "\n";
$no = 'this is a string';
echo is_array($no) ? 'Array' : 'not an Array';
?>
```

Output:

```
Array
not an Array
```

bool is_int()

Return true if var is an integer, false otherwise.

Example:

```
<?php
if(is_int(23))
{ echo "is integer\n"; }
else { echo "is not an integer\n"; }
var_dump(is_int(23));
var_dump(is_int("23"));
var_dump(is_int(23.5));
var_dump(is_int(true));
?>
```

Output:

```
is integer
bool(true)
bool(false)
bool(false)
bool(false)
```

extract()

Convert array name, value pair to variable, value pair

Example:

```
<?php
$var_array = array("color" => "blue",
                   "size" => "medium",
                   "shape" => "sphere");
extract($var_array);
echo"$color, $size, $shape\n";
$_POST = array("name" => "Jitendra",
               "age" => "28");
extract($_POST);
echo"$name, $age";
?>
```

Output:

blue, medium, sphere

Jitendra, 28

implode() to form

Join array elements with a string

Example:

```
<?php
$array = array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);
echo $comma_separated;
?>
```

Output:

lastname,email,phone

explode()

Return array of string, each of which is a substring of string.

Example:

```
<?php
$comma_separated = "lastname,email,phone";
$arr = explode(",", $comma_separated);
echo "$arr['0']\n$arr['1']\n$arr['2']";
?>
```

Output:

lastname
email
phone

```
//Fetching object array
while($row = $result->fetch_object()) {
    printf("%s (%s)\n", $row->Name, $row->Country);
}
```

list()

Assign variable as if they were an array.

Example:

```
<?php
$info = array('coffee', 'brown', 'caffeine');
list($drink, $color, $power) = $info;
echo "$drink is $color and $power makes it special.\n";
list($drink, , $power) = $info;
echo "$drink has $power.\n";
list(, , $power) = $info;
echo "I need $power\n";
?>
```

Output:

```
coffee is brown and caffeine makes it special.
coffee has caffeine
I need caffeine
```

Date 05-02-2021

```
$mysqli = new mysqli("localhost", "my_user",
        "my_pass", "my_db");

//check connection
if($mysqli->connect_error) {
    die('Connect Error ('.$mysqli->connect_errno.')'
        .$mysqli->connect_error);
//Select queries return a result set
if($result = $mysqli->query("SELECT Name FROM City LIMIT
    10")) {
    printf("Select returned %d rows.\n", $result->num_rows);
    //free result set
    $result->close();
}
//close connection
$mysqli->close();
```

fetch_array() affected_rows()
fetch_object()
fetch_row()
fetch_assoc()
);

array mysql_fetch_array()

Fetch a result row as an associative array, a numeric array, or both & moves the internal data pointer ahead.

array mysql_fetch_assoc()

Fetch a result row as an associative array.

array mysql_fetch_row()

Fetch a result row as an numerical array.

object mysql_fetch_object()

Fetch a result row as an object.

Example:

```
<?php
$link = mysql_connect("localhost","root","");
if(!$link)
{
    die('Could not connect: '.mysql_error());
}
mysql_select_db("mydb");
$result = mysql_query("select id, name from mytable");
while($row = mysql_fetch_array($result))
{
    printf("ID: %s Name: %s", $row["id"], $row["name"]);
                    or
    printf("ID: %s Name: %s", $row[0], $row[1]);
}
                    or
while($row = mysql_fetch_object($result))
{
    echo $row->id;
    echo $row->name;
}
```

mysql_connect()

Open a connection to a MySQL server

die()

Equivalent to exit() ⇒ Output a message and terminate the current script.

mysql_error()

Returns the text of the error message from previous MySQL operation.

mysql_select_db() (bool)

Select a MySQL database

mysql_query()

sends an unique query to the currently active database on the server that's associated with the specified link_identifier.

Example:

Look just previous example

floor()

Returns lower bound for a float

Example:

```
<?php
echo floor(4.3);
echo floor(9.999);
echo floor(-3.14);
?>
```

Output:

```
4
9
-4
```

ceil()

Returns upper bound for a float

Example:

```
<?php
echo ceil(4.3);
echo ceil(9.999);
echo ceil(-3.14);
?>
```

Output:

```
5
10
-3
```

round()

Returns the rounded value for a float to specified number of digits after the decimal point.

Example:

```
<?php
echo round(3.4);
echo round(3.5);
echo round(3.6,0);
echo round(1.95583,2);
echo round(1241757,-3);
echo round(5.045,2);
?>
```

Output:

```
3
4
4
1.96
1242000
5.05
```

rand()

Generate a random integer between two specified integers.

Example:

```
<?php
 echo rand()."\n";
 echo rand()."\n";
 echo rand(5,15);
?>
```

If no parameters passed to it, it generates an integer between 0 & getrandmax(). (32767)

Output:

```
7771
22264
11
```

count()

Counts elements in an array or properties in an object

Example:

```
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count($a);
echo $result;
```

```
$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$result2 = count($b);
echo $result2;
```

Output:

```
3
```

```
3
```

substr() string substr(string, int $start [, int $length])

Returns part of a string starting from a specified index for a specified length.

Example:

Output:

```
<?php
$rest = substr("abcdef", 0, -1);        "abcde"
$rest = substr("abcdef", 2, -1);        "cde"
$rest = substr("abcdef", 4, -4);        ""
$rest = substr("abcdef", -3, -1);       "de"

echo substr('abcdef', 1);               "bcdef"
echo substr('abcdef', 1, 3);            "bcd"
echo substr('abcdef', 0, 4);            "abcd"
echo substr('abcdef', 0, 8);            "abcdef"
echo substr('abcdef', -1, 1);           "f"
?>
```

string strtr(string $str, string $from, string $to)
or
(string $str, array $replace_pairs)

This function returns a copy of str, translating all occurrences of each character in from to the corresponding character in to.

Example:

```
<?php $trans = array("hello" => "hi", "hi" => "hello");
      echo strtr("hi all, I said hello", $trans);
?>
```

Output:

hello all, I said hi

substr_count() substr_count($haystack, $needle [, $offset [, $length]])

Returns the no. of times for a substr occured in a string.

Example:

```
<?php
$text = 'This is a test';
echo strlen($text); // 14
echo substr_count($text, 'is'); // 2
// the string is reduced to 's is a test', so it prints 1
echo substr_count($text, 'is', 3);

// the text is reduced to 's i', so it prints 0
echo substr_count($text, 'is', 3, 3);

// generates a warning because 5+10 > 14
echo substr_count($text, 'is', 5, 10);

// prints only 1, because it doesn't count overlapped substrings
$text2 = 'gcdgcdgcd';
echo substr_count($text2, 'gcdgcd');
?>
```

str_replace() str_replace($search, $replace, $subject[, $count])

Replace all occurrences of the search string with the replacement string. $count, if passed, this will hold the no. of matched and replaced needles.

Example:

```
$vowels = array("a","e","i","o","u","A","E","I","O","U");
$onlyconsonants = str_replace($vowels,"","Hello World of PHP");
// Hll Wrld f PHP

$phrase = "You should eat fruits, vegetables, and fiber every
          day.";
$healthy = array("fruits","vegetables","fiber");
$yummy = array("pizza","beer","ice cream");
// You should eat pizza, beer, and ice cream every day
   str_replace($healthy, $yummy, $phrase);
$str = str_replace("ll","","good golly miss molly",$count);
echo $count; //2
```

int strlen()

Returns number of characters in a string.

Example:

```
<?php
$str = 'abcdef';
echo strlen($str);
$str = ' ab cd ';
echo strlen($str);
?>
```

Output:

```
6
7
```

string strtolower()

Returns string with all alphabetic characters converted to lowercase.

string strtoupper()

Returns string with all alphabetic characters converted to uppercase.

Example:

```
<?php
$str = "Marry Had A Little Lamb and She LOVED It So";
$str1 = strtolower($str);
echo $str1; echo "\n";
$str2 = strtoupper($str);
echo $str2;
?>
```

Output:

```
marry had a little lamb and she loved it so
MARRY HAD A LITTLE LAMB AND SHE LOVED IT SO
```

string trim(string $str [, string $charlist])

strip whitespace (or other characters) from the beginning and end of a string.

Example:

```
<?php
$text = "\t These :) ... ";
$hello = "Hello World";
$trimmed = trim($text);          // These :) ...
var_dump($trimmed);              ✓
$trimmed = trim($text, " \t.");  // These :)
var_dump($trimmed);              ✓
$trimmed = trim($hello, "Hdle"); // o Wor
var_dump($trimmed);              ✓
?>
```

Output:

```
string(12) These :) ...
string(8) These :)
string(5) o Wor
```

Date 25/01/2019

string strstr(string $substring, mixed $whole_string [, $before_substring = false])

Returns the substring, starting from the first occurence of the $substring to the end of the $whole string and false if no match found. Searching process is casesensitive.

Example:

```
$email = 'name@example.com';
$domain = strstr($email, '@');
echo $domain; // @example.com
```

addslashes()

Returns a string with backslashes before characters single quote('), double quote("), backslash(\) and NUL (the NULL byte).

Example:

```
<?php
$str = "Is your name O'reilly?";
echo addslashes($str);
?>
```

Output:

Is your name O\'reilly?

mysql_real_escape_string()

Returns a string with backslashes before special characters in a string for use in a SQL statement.

This function uses backslashes for the following characters:

\x00, \n, \r, \, ', ", \x1a

stripslashes()

Un-quote string quoted with addslashes()

Example:

```
<?php
$str = "Is your name O\'reilly?";
echo stripslashes($str);
?>
```

Output:

Is your name O'reilly?

bool file_exists()

checks whether a file or directory exists.

Example:

```
<?php
$filename = '/path/to/foo.txt'; or '/path/to/file';
if(file_exists($filename))
{ echo "The file $filename exists"; }
 else { echo "The file $filename does not exist"; }
?>
```

SESSION

session_start()

Creates a session or resumes the current one based on the current session id.

session_unset()

Free all session variables. (means $_SESSION[] except session_id())

session_destroy()

Destroy or finish the current session. It does not unset any of the global variables associated with the session, or unset the session cookie. (session_id() value destroy but $_SESSION[] still remains until control gets session_start() again)

session_regenerate_id()

Update the current session id with a newly generated one. It will replace the current session id with a new one and keep the current session information.

session_id() (string)

Get and/or set the current session id.

Example:

```
$_SESSION['jitu'] = 25;    $_SESSION['jitu'] = "";
session_id("abc45dekl"); echo session_id(); //abc45dekl
```

```
session_start();
- - - - - - - - - -
- - - - - - - - -
session_unset();
session_destroy();
session_start();
session_regenerate_id();
```

from bombfuse

COOKIE

time()+3600 //expire in 1 hour

setcookie(string $name [, string $value [, int expire [,... ...]]])

If not set, expire after closing browser

can be get cookie through

$_COOKIE['cookiename']

Example:

```
<?php
$value = 'something from somewhere';
setcookie("TestCookie", $value, time()+3600); //expire in 1 hour
echo $_COOKIE["TestCookie"];  //something from somewhere
?>
```

Research

session_start()

$_SESSION[' '] = —;

session_unset(); ⇒ when it occurs, after it, global variable $_SESSION[] will be unavailable but session_id() still available

OR

session_regenerate_id() ⇒ when it occurs, after it, new session_id() would be available in place of old one. global variable would be available means $_SESSION[] as previously.

OR

session_destroy() ⇒ when it occurs, after it, session_id() would be unavailable but global variable $_SESSION[] still be available as previously. But after it, when system again gets session_start(), after it $_SESSION[] would be unavailable & old session_id() would be available.

a and A ⇒ am, pm and AM, PM
G and H ⇒ 0 to 23 and 00 to 23
g and h ⇒ 1 to 12 and 01 to 12
F and M ⇒ January to December and Jan to Dec
l and D ⇒ Sunday to Saturday and Sun to Sat

There are 3 characters:
① ⇒ single quote (')
② ⇒ double quote (")
③ ⇒ Back slash (\)
④ ⇒ Less than (<)

Problem occur when we enter (a<b) → D think, it takes as an open html tag. which may make trouble for making query, that's why we use addslashes() or mysql_real_scape_string() and stripslashes() function.

mktime(2,9,5,2,3,2017);
Set Unix Timestamp for a date
mktime(Hour, minute, second, month, day, year)
Use non leading zero values
2 Feb 2017 02:02:02
H i s m n d j Y
02 02 02 02 2 02 2 2017
Feb
M
17
y
Between 1970 to 2069

Parse any English textual datetime description into a Unix timestamp
strtotime()

"1 January 2009"
"+1 week"
"+1 day"
"-1 day"
"+1 month"
"+1 year"

Create one data format into another:

```
$date = DateTime::createFromFormat('Y-m-d H:i:s', '2009-02-15 15:16:17');
echo $date->format('m-d-Y i');
//Feb-15-2009 16
```

date()
time() → return current timestamp

header()

It is used to send a raw http header (used to redirect to a particular page)

(redirection works only when there is no "echo" statement before 'header' on the page)

Example:

```
header('location: jitu.php');
```

ob_start()

It will return output buffering on. While output buffering is active no output is sent from the script (other than headers), instead the output is stored in an internal buffer.

ob_clean()

Clean (erase) the output buffer.

ob_end_flush()

Flush (send) the output buffer and turn off output buffering.

For redirection, also

```
<?php echo "<script>window.location.href='jitu.php'</scr
                ipt>";
        or
<script>
   window.location.href='jitu.php';
</script>
```

http://midas/parentainment.com/spro_view_profile.php?mem_id=2

$_SERVER['HTTP_HOST'] ⇒ midas

$_SERVER['REQUEST_URI'] ⇒
/parentainment.com/spro_view_profile.php?mem_id=2

$_SERVER['SCRIPT_NAME'] ⇒
/parentainment.com/spro_view_profile.php

$_SERVER['PHP_SELF'] ⇒
/parentainment.com/spro_view_profile.php

$_SERVER['QUERY_STRING'] ⇒ mem_id=2

$_SERVER['HTTP_REFERER'] ⇒

phpinfo()

Outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.

phpinfo() is also a valuable debugging tool as it contains all EGPCS (Environment, GET, POST, Cookie, Server) data.

getcwd()

Returns the current working directory.

uniqid()

Gives unique identifier based on the current time in microseconds.

Example:

```
$better_token = md5(uniqid(rand(),true));
(more better way to get a unique identifier)
```

nl2br()

Returns string with '
' inserted before all newlines.

Example:

```
<html><head></head><body>jitendra isn't<br>jitu</body></html>
```

Output:

jitendra isn't
jitu

```
<body><?php echo "jitendra isn't\n jitu"; ?></body>
```

Output: jitendra isn't jitu

```
<body><?php echo nl2br("jitendra isn't\n jitu");
```

Output:

jitendra isn't
jitu

```
<body><?php echo "jitendra isn't<br>jitu"; ?></body>
```

Output:

jitendra isn't
jitu

File Upload

```
if($_FILES['my_image']['name'] != '')
{
    list($width, $height, $image_type, $wh)
    = getimagesize($_FILES['my_image']['tmp_name']);
    if($image_type != '')
    {
        .....
        unlink();
        $file_tmp_name = $_FILES['my_image']['tmp_name'];
        $file_name = $_FILES['my_image']['name'];
        $file_path = md5(uniqid(rand(), true)).$file_name;
        copy($file_tmp_name, $file_path);
      or
        move_uploaded_file($file_tmp_name, $file_path);
    }
}
```

```
<form — enctype="multipart/form-data">
  <input type="file" name="my_image">
</form>
```

define()

Defines a named constant at runtime.

Example:

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT;
echo Constant;
?>
```

Output:

```
Hello world.
Constant
```

Date
25/02/2017

XSS

If you embed strings within HTML markup, we must escape it with htmlspecialchars(). This means that every single echo or print statement (for ex. in zend framework for layout & template i.e. for every .phtml file) should use htmlspecialchars().

~~htmlentities()~~ <? echo htmlspecialchars('<jitu>'); ?>
~~html_entity_decode()~~

output: <jitu>

On browser it will output the same as: <jitu>
(but not mistranslated by the browser)

htmlspecialchars()

Html special characters:
htmlspecialchars_decode()

Translations performed are:

'&' (ampersand)	'&'
'"' (double quote)	'"'
''' (single quote)	'''
'<' (less than)	'<'
'>' (greater than)	'>'

```
</HTML> ji'tu"ka
<script>alert('kutte');</script>
```

a textarea in form

Our midas.inc.php uses addslashes() function for each form value when form posts.

So that when query executes it takes as:

db_query("update table1 set name='ji\'tu\"ka' where id='1'");

it may be correctly as:

db_query("update table1 set name='ji\'tu\"' where id='1'");

In it sql understands that we do not finish query but it is just a single or double quote among query not a stopper.

But in database it saves as it:

ji'tu"ka or </HTML> ji'tu"ka
<script>alert('kutte');</script>

when we display these, we must translate the special html character in it so that site can not be hacked that is mistranslated.

ie.

echo "

Use ms_form_value() ⇒ to show or display value in form element
ms_display_value() ⇒ to display database value out of form as it is ⇒ when show or display value in fck editor.

array_key_exists(mixed $key, array $search) (bool)

Checks if the given key or index exists in the array

Example:

```
<?php
 $search_array = array('first' => 1, 'second' => 4);
 if(array_key_exists('first', $search_array))
 {
    echo "The 'first' element is in the array";
 } ?>
```

Output:

The 'first' element is in the array

bool method_exists (object $object, string $method_name)

Checks if the class method exists in the ~~class~~ given object

Example: class.MyClass.php (this page name)

```
class MyClass
{
    function MyMethod()
    { echo "jitu"; }
}
class Yourclass
{ function MyTest() {
    $a = new MyClass();
    if(method_exists($a, 'mymethod'))
    {
        echo "Jitendra";
    } }
}
```

testing.php

```
<?php
    include_once('class.MyClass.php');
    $b = new Yourclass();
    $b -> MyTest();
?>
```

Output: When running testing.php, we get:

Jitendra

bool in_array(mixed $n, array $h)

Searches h for n and returns true if it is found in the array, false otherwise.

Example:

```
<?php   $os = array("Mac", "Jric", "Antic");
        if(in_array("Jric", $os)
        { echo "Jitendra"; }
?>
```

Output:

Jitendra

array array_map(callback $callback, array $arr)

It returns an array containing all the elements of arr after applying the callback function.

Example:

```
<?php  function cube($n) { return ($n * $n * $n); }
       $a = array(1, 2, 3, 4);
       $b = array_map("cube", $a);
       print_r($b);
?>
```

Output: Array ([0] => 1, [1] => 8, [2] => 27, [3] => 64)

mixed array_shift(array &$array)

It shifts the first value of the array off and returns it. If array is empty (or is not an array), NULL will be returned.

Example:

```
<?php
$stack = array("ora", "baba", "app", "rasp");
$fruit = array_shift($stack);
print_r($stack);
?>
```

Output:

```
Array
(
    [0] => baba
    [1] => app
    [2] => rasp
)
```

int array_push (array &$array, mixed $var [, mixed $...])

It treats array as a stack, pushes the passed variables onto the end of array. It returns the new total number of elements in the array.

Example:

```
<?php
$stack = array("orange", "banana");
array_push($stack, "apple", "raspberry");
print_r($stack);
?>
```

Output:

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
    [3] => raspberry
)
```

str ini_get(string $varname)

Returns the value of the configuration option on success otherwise null value or empty string.

Example:

```
<?php
 echo ini_get('upload_max_filesize');
?>
```

Output:

2M

array ini_get_all()

Returns all configuration options in the form of associated array.

Date 07/02/2017

short_open_tag [make it On in php.ini to make use of php short open tag <? instead of <?php]

<?php echo $a ?> is equal to <?= $a ?>

If a file is pure PHP code, it is preferable to omit the PHP closing tag at the end of the file. This prevents accidental whitespace.

short open tag is discouraged since it is available only if enabled in php.ini.

About mxtoolbox.com

Checking that an email is comming from a secure email server or not.

Example: Go to jitendra.research@gmail.com and search with 'mxtoolbox.com'

1. get comming mail & click on 'show original'
2. Take copy of mail header
3. Go to mxtoolbox.com & click on 'Analyze Headers'
4. Taking from step2, paste in textarea & click on 'Analyze Header'.
5. Check your mail is comming through your used email server or not. In this case, I am using an email server 'secureserver.net'

array_search()

Searches the array for a given value and returns the first corresponding key if successful; FALSE otherwise.

Example:

```
$array = array(0 => 'blue', 1 => 'red', 2 => 'green',
               3 => 'red');
$key = array_search('green', $array); //$key = 2
$key = array_search('red', $array); //$key = 1
```

array_combine()

Creates an array by using one array for keys another for its values.

Return the combined array, FALSE if the number of elements for each array isn't equal.

Example:

```
<?php
  $a = array('green', 'red', 'yellow');
  $b = array('avocado', 'apple', 'banana');
  $c = array_combine($a, $b);
 print_r($c);
?>
```

Output:

```
Array
(
    [green] => avocado
    [red] => apple
    [yellow] => banana
)
```

Date
27/04/2014

Auto Refresh the PHP Page :

```
<?php
$page = $_SERVER['PHP_SELF'];
$sec = "10";
header("Refresh: $sec; url = $page");
echo "Watch the page reload itself in 10 seconds";
?>
```

Date
07/02/2017

Sorting Arrays

bool sort(): array &$array

- Sort an array. Return TRUE on success or FALSE on failure.

Example: and reset index association.

```
<?php
$fruits = array("lemon","orange","banana",
          "apple");
sort($fruits);
foreach($fruits as $key => $val) {
    echo "fruits[" . $key . "] = " . $val . "\n";
}
```

Output:

```
fruits[0] = apple
fruits[1] = banana
fruits[2] = lemon
fruits[3] = orange
```

Example:

```
$arr = array(45, 32, 21, 101);
$resultArr = sort($arr);
print_r($arr);
echo "<br>";
print_r($resultArr);
```

Output:

```
Array ([0] => 21 [1] => 32 [2] => 45 [3] => 101)
1
```

Date
07/02/2017

~~ksort()~~

bool ksort (array &$array) : Sort an array by key.

bool asort(array &$array) : Sort an array and maintain index association.

bool rsort(array &$array) : Sort an array in reverse order.
(opposite of sort())

bool krsort(array &$array) : Sort an array by key in reverse order.
(opposite of ksort())

bool arsort(array &$array) : Sort an array in reverse order and maintain index association.
(opposite of asort())

Date
07/02/2017

file_get_contents()

Reads a file & storing it in a string variable.

Example: $xyz = file_get_contents("http://blog.ajinsa.com");

An amp thing

```
$xyz = 123;
echo '$xyz'; // $xyz
echo "{$xyz}"; // '123'
```

An amp thing

```
$x = 123;
$aa = 'a';
$ab = 2;

echo $x; // 123
echo "$x"; // 123
echo "'$x'"; // '123'
echo '$x'; // $x
echo ${$aa.'b'}; // 2
```

Date
09/02/2017

bool is_numeric()

To check whether a variable is numeric or not.

Example:

```
echo is_numeric("123");   // 1
echo is_numeric(123);     // 1
echo is_numeric("Hi");    // Nothing
echo is_numeric(9.1);     // 1
echo is_numeric(array()); // Nothing (not even 0 to)
```

---

bool is_bool()

Finds whether a variable is boolean or not.
Remember boolean type only has values
TRUE or true or True & FALSE or false or False

Example:

```
is_bool(false); // return TRUE   On echo // 1
is_bool(0);     // return False  On echo // Nothing
```

(0 ← zero)

---

bool is_int()
bool is_float()
bool is_double()
bool is_array()
bool is_null()
bool is_object()

Example:

```
error_reporting(E_ALL);
$foo = NULL;
var_dump(is_null($x));
var_dump(is_null($foo));
```

Output:

```
Notice: Undefined variable: x
// bool(true)
// bool(true)
```

Date
08/02/2017

bool is_string()

Find whether the type of variable is string.

Example:

```
var_dump(is_string(false)); // bool(false)
         (true)             // bool(false)
         (null)             // bool(false)
         ('abc')            // bool(true)
         ('23')             // bool(true)
         (23)               // bool(false)
         ('23.5')           // bool(true)
         (23.5)             // bool(false)
         ('')               // bool(true)
         (' ')              // bool(true)
         ('0')              // bool(true)
         (0)                // bool(false)
```

Variables are case sensitive in PHP

Yes, either it is windows or Linux.

Example: $x & $X are two variables different

```
$x=25;
echo $X;
```

Output:

Notice: Undefined variable: X

Date
09/02/2017

bool isset()

Determine if a variable is set and is not NULL.

Example:

```
$a = '';
$b = "test";
$c = NULL;
var_dump(isset($a));    // TRUE
var_dump(isset($b));    // TRUE
var_dump(isset($c));    // FALSE
unset($b);
var_dump(isset($b));    // FALSE
```

If multiple parameters are supplied then isset() will return TRUE only if all of the parameters are set. Evaluation goes from left to right & stops as soon as an unset variable is encountered.

Example:

```
var_dump(isset($a, $b));        // bool(TRUE)
var_dump(isset($a, $b, $c));    // bool(FALSE)
```

Also note that a null character ("\0") is not equivalent to the PHP NULL constant.

Date
09/02/2017

void unset (mixed $var [, mixed $...])

It destroys the specified variables.

Example:

```
$x = 25;
unset($x);
echo $x;
```

Output:

Notice: Undefined variable: x

If a globalized variable is unset() inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before unset() was called.

Example:

```
function destroy_foo()
{
    global $foo;
    unset($foo);
}
$foo = 'bar';
destroy_foo();
echo $foo;
```

Output:

bar

Example:

```
function foo()
{
    unset($GLOBALS['bar']);
}
$bar = "something";
foo();
echo $bar;
```

Output:

Undefined variable: bar

To unset a global variable inside of a function, use $GLOBALS array to do so.

If a variable that is PASSED BY REFERENCE is

Date
09/02/2017

unset() inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before unset() was called.

Example:

```
function foo(&$bar)
{
    unset($bar);
    $bar = "blah";
}
$bar = 'something';
echo "$bar\n";
foo($bar);
echo "$bar\n";
```

Output:

```
something
something
```

Example:

```
function foo()
{
    static $bar;
    $bar++;
    echo "Before unset: $bar, ";
    unset($bar);
    $bar = 23;
    echo "after unset: $bar\n";
}
foo();
foo();
foo();
```

If a static variable is unset() inside of a function, unset() destroys the variable only in the context of the rest of a function. Following calls will restore the previous value of a variable.

Output:

```
Before unset: 1, after unset: 23
Before unset: 2, after unset: 23
Before unset: 3, after unset: 23
```

Date
10/02/2015

Some similar functions

htmlspecialchars()
htmlspecialchars_decode()
htmlentities()
html_entity_decode()
urlencode()
urldecode()
addslashes()
stripslashes()

Date
11/02/2017

bool ctype_alnum()

It checks for alphanumeric character(s). Return TRUE if every character in provided string/text is either a letter or a digit, FALSE otherwise.

Example:

```
ctype_alnum('AbCd1zyZ9');   // TRUE
ctype_alnum('foo!#$bar');   // FALSE
```

bool ctype_alpha()

It checks for alphabetic character(s). It means only letters allowed.

Example:

```
ctype_alpha('KjgWZC');   // TRUE
ctype_alpha('arf12');    // FALSE
```

bool ctype_digit()

It checks for numeric character(s). It means only digits allowed.

Example:

```
ctype_digit('1020.20');   // FALSE
ctype_digit('10802');     // TRUE
ctype_digit('wsl!12');    // FALSE
```

Date
11/02/2017

bool empty()

It returns FALSE if var exists and has a non-empty, non-zero value, otherwise returns TRUE.

Example:

```
$a = 11;
$b = 0
$c = NULL;
$d = 23; $f;
empty($a);    //TRUE
empty($b);    //TRUE
empty($c);    //TRUE
empty($d);    //FALSE
empty($e);    //TRUE
empty($f);    //TRUE
```

Remember, if a variable declared only just as $f in above example, it means it has NULL value by default.

Boolean values:

TRUE/True/true/TRue/TrUe/TruE are all same.

FALSE/False/false/FAlse/FaLse are all same.

NULL/null/Null/NULL/NuLl are all same.

```
$a = TRUE;
$b = FALSE;
echo $a;    //1
echo $b;    //  <- Nothing means blank
```

Date
14/02/2017

Continuous Integration (CI)

Continuous integration (CI) is a software engineering practice in which isolated changes are immediately tested and reported on when they are added to a larger code base. The goal of CI is to provide rapid feedback so that if a defect is introduced into the code base, it can be identified and corrected as soon as possible. Continuous integration software tools can be used to automate the testing and build a document trail.

Famous CI tool in market are PHPCI (specifically designed for PHP), Jenkins, Travis CI, etc.

Continuous integration reduces bugs, increases productivity & improves software quality.

Continuous Delivery (CD)

Continuous delivery (CD) is an extension of the concept of continuous integration (CI). Whereas CI deals with the build/test part of the development cycle for each version, CD focuses on what happens with a committed change after that point. With continuous delivery, any commit that passes the automated tests can be considered a valid candidate for release.

Date
14/02/2017

An important goal of continuous delivery is to make feedback loops as short as possible. Because code is delivered in a steady stream to user acceptance testing (UAT) or the staging environment, cause and effect can be observed early & code can be tested for all aspects of functionality, including business rule logic.

PHPCI

PHPCI is a continuous integration tool, written in PHP & specifically designed for PHP.

https://www.phptesting.org

Date
17/02/2017

Difference between 'abstract' & 'final'

abstract class & final class are exactly opposite to each other.

abstract class: We can not create object of an abstract class but ~~we can create~~ a class can extend an abstract class.

final class: We can create an object of a final class but any class can not extend a final class.

Important fact in OOP programming

A protected variable and functions can be accessed in child class as well as in grand child class, grand grand child class, etc.

In above point, a protected function can be accessed through parent:: as well as $this-> too where the function is non-static in a child, grand child or grand grand child class etc.; but a non-static protected variable can be accessed only through $this-> in child or grand child or grand grand child class, etc.

Example:

```
abstract class A
{
    protected $aaa = 25;
    protected function setUp()
    {
        echo '<br> I am inside setUp function'.__CLASS__.'!';
    }
}
```


get_class($this);

Date
17/02/2017

```
abstract class B extends A
{
    function sayHello()
    {
        echo '<br>I am inside sayHello function.';
    }
    private function sayHi()
    {
        echo '<br>I am inside sayHi function.';
    }
}
class C extends B
{
    function setUp()
    {
        echo '<br> I am inside child setUp function.';
        parent::setUp();
        /* $this->setUp(); // If it executes, trap in
            an infinite loop as it calls itself i.e.
            setUp() */
        /* But if this setUp() does not exist in
            current class C, then $this->setUp() can
            be called from below function sayHi() and
            it will present the same results as parent::
                                        setUp() */
    }
    function sayHi()
    {
        echo '<br> I am inside child sayHi function.';
    }
}
```

Date
17/02/2017

```
$abc = new C();
$abc->setUp();
$abc->sayHi();
```

Output:

```
I am inside child setUp function.
I am inside setUp function. A C
I am inside child sayHi function.
```

Imp fact about chain of interface, abstract and normal class

```
interface A
{
    public function infa();
}
interface B
{
    public function infb();
}
interface C extends A,B
{
    public function infc();
}
interface D
{
    public function infd();
}
```

Date
17/02/2017

```
abstract class E implements C, D
{
    public abstract function abstracte();
    public abstract function abstractee();
}
abstract class F extends E
{
    public abstract function abstractf();
}

class G extends F
{
    public function infa() { echo '<br>infa'; }
    public function infb() { echo '<br>infb'; }
    public function infc() { echo '<br>infc'; }
    public function infd() { echo '<br>infd'; }
    public function abstracte() { echo '<br>abstracte'; }
    public function abstractee() { echo '<br>abstractee'; }
    public function abstractf() { echo '<br>abstractf'; }
}
```

```
$abc = new G();
$abc->infa();
$abc->infb();
$abc->infc();
$abc->infd();
$abc->abstracte();
$abc->abstractee();
$abc->abstractf();
```

Output:

infa
infb
infc
infd
abstracte
abstractee
abstractf

Date
17/02/2017

Conclusion: When an abstract class implements an interface then it has flexibility to define or not define the all functions of interface.

But when a normal class extends an abstract class then it's responsibility/duty of normal class to define all methods of abstract class as well as the all methods of interface extended by the abstract class.

Date
20/02/2017

preg_split()

Split string by a regular expression. Return an array on success & FALSE on failure. split() is deprecated & removed in PHP 7.0.

Example:

```
$date = "04/30/1973";
list($month, $day, $year) = preg_split('/[\/]/'
                                , $date);
echo "Month: $month; Day: $day; Year: $year";
```

Output:

Month: 04; Day: 30; Year: 1973

Example:

```
$keywords = preg_split("/[\s,]+/", "hypertext lan
print_r($keywords);                guage, programming");
```

(1 space)

Output:

```
Array
(
    [0] => hypertext
    [1] => language
    [2] => programming
)
```

Date
20/02/2017

file_put_contents()

Write a string to a file. This function is identical to calling fopen(), fwrite() & fclose() successively to write data to a file. It returns the number of bytes that were written to the file, or FALSE on failure.

Example:

```
$file = 'people.txt';
// open the file to get existing content
$current = file_get_contents($file);
// Append a new person to the file
$current .= "John Smith\n";
// Write the contents back to the file
file_put_contents($file, $current);
```

Example:

```
$file = 'people.txt';
// the new person to add to the file
$person = "John Smith\n";
// Write the contents to the file,
// using the FILE_APPEND flag to append the content
// to the end of the file and the LOCK_EX flag
// to prevent anyone else writing to the file at the
// same time
file_put_contents($file, $person, FILE_APPEND | LOCK_EX);
```

Date 20/02/2015

bool mail()

bool mail (string $to, string $subject, string $message [, string $additional_headers [, string $additional_parameters]])

It is used to send mail in PHP.

---

bool unlink(string $filename)

It deletes a file. It returns TRUE on success and FALSE on failure.

Example:

```
unlink('test.html');
```

---

bool file_exists(string $filename)

checks whether a file or directory exists. It returns TRUE if the file/directory exists and FALSE otherwise.

Example:

```
$fileName = '/path/to/foo.txt';
//or $dirName = '/path/to';
if(file_exists($fileName)){
//or if(file_exists($dirName)){
    echo "file/directory exists";
}else{
    echo "file/directory does not exist";
}
```

Date
20/02/2011

bool rmdir(string $dirname)

Attempts to remove the directory. The directory must be empty & the relevant permissions must permit this.

Example:

```
if(!is_dir('example')){
    mkdir('example');
}
rmdir('example');
```

bool is_dir(string $filename)

Checks whether $filename is a directory.

bool mkdir(string $pathname [, int $mode = 0777 [, bool $recursive = false]])

$mode ⇒ The mode is 0777 by default, which means the widest possible access.

$recursive ⇒ Allow the creation of nested directories specified in $pathname

bool is_readable(string $filename)

Checks whether a file exists & is readable.

bool is_writable(string $filename)

Checks whether the filename is writable.

Date
20/02/2017

search start position. By default, it is 0

mixed strpos ($str, $substr [, int $offset = 0])

Find the first occurrence of a substring in a string. It returns the position 0 & onwards and FALSE on failure.

Example:

```
$mystr = 'abc';
$findme = 'a';
$pos = strpos($mystr, $findme);  // 0
```

Remind one thing:

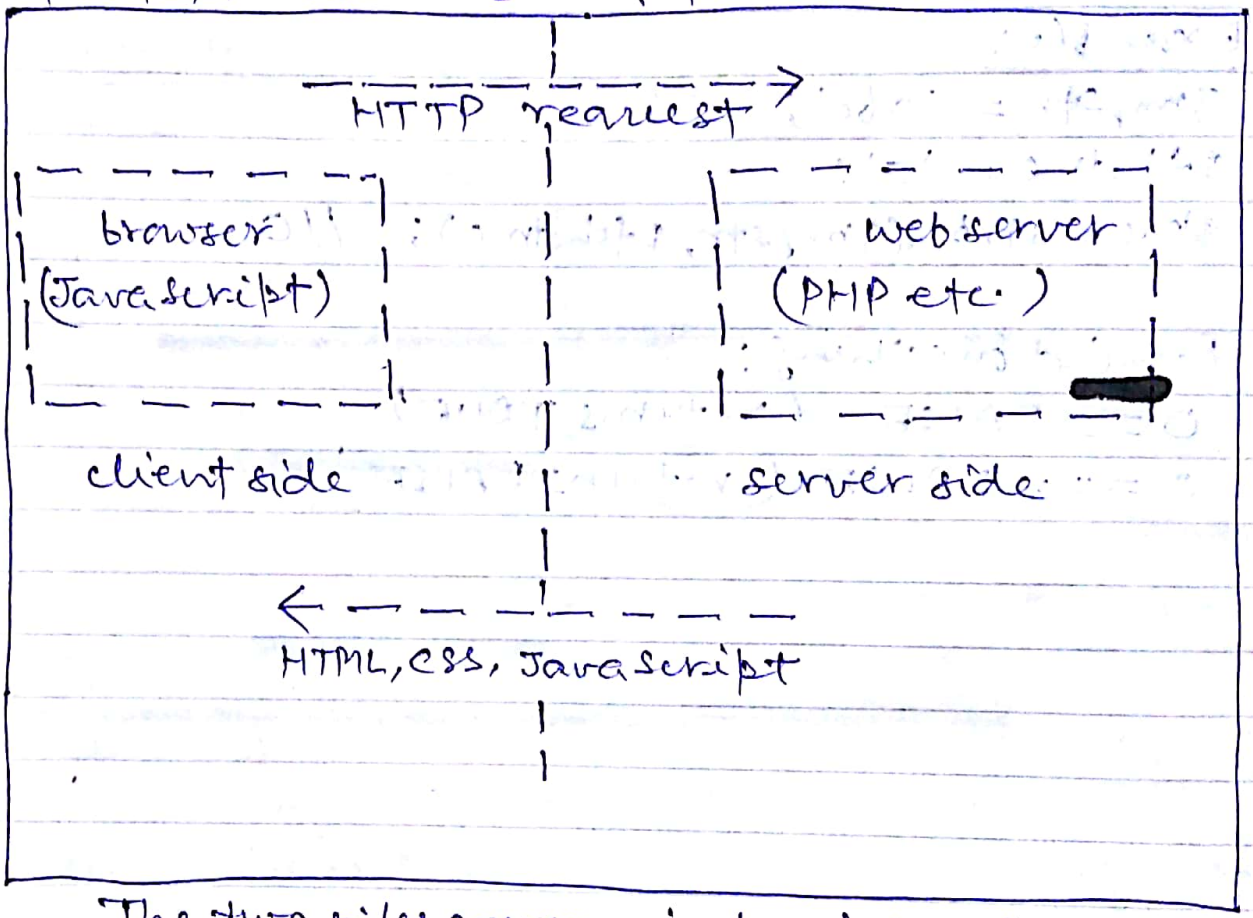
0 == FALSE (returns TRUE)

0 === FALSE (returns FALSE)

Date
21/02/2017

Difference between client-side and server-side programming:

Our code is split into two entirely separate parts, the server side & the client side.



The two sides communicate via HTTP requests and responses. PHP is executed on the server and outputs some HTML and may be javascript code which is sent as (HTTP) response to the client where the HTML is interpreted and the Javascript is executed. Once PHP has finished outputting the response, the script ends and nothing will happen on the server until a new HTTP request comes in.

The example code executes like this:

```
<script type="text/javascript">
    var foo = 'bar';
```

```
    <?php file_put_contents('foo.txt', '.$foo.'); ?>
    var baz = <?php echo 42; ?>;
    alert(baz);
</script>
```

step 1, PHP executes all code between <?php ?> tags. The result is this:

```
<script type="text/javascript">
    var foo = 'bar';
    var baz = 42;
    alert(baz);
</script>
```

The file_put_contents call did not result in anything, it just wrote "."$foo."" into a file. The <?php echo 42; ?> call resulted in the output "42", which is now in the spot where that code used to be.

This resulting HTML/Javascript/CSS code is now sent to the client (browser), where it gets evaluated. The 'alert' call works, while the 'foo' variable is not used anywhere

Date 21/02/2017

Ajax ⇒ Asynchronous Javascript and XML

XML HttpRequest

All modern browsers have a built-in XMLHttpRequest object to request data from a web server.

Benefits of XMLHttpRequest object

The XMLHttpRequest object is a developer's dream, because using it, we can:

1. Update a web page without reloading the page
2. Request data from a server - after the page has loaded.
3. Receive data from a server - after the page has loaded.
4. Send data to a server - in the background.

Reference: https://www.w3schools.com/xml/xml_http.asp

Date
21/02/2017

cURL

PHP supports libcurl, a library created by Daniel Stenberg, that allows you to connect and communicate to many different types of servers with many different types of protocols.

libcurl currently supports the http, https, ftp, gopher, telnet, dict, file, and ldap protocols.

libcurl also supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, cookies, and user+password authentication.

In other words:

cURL is a library that lets you make HTTP request in PHP.

cURL is a way you can hit a URL from your code to get a html response from it. cURL means client URL which allows you to connect with other URLs and use their response in your code.

Reference: php.net/manual/en/intro.curl.php

Date
21/02/2017

Reference: www.pcds.co.in/java/what-is-pear.php

PEAR

PEAR (PHP Extension and Application Repository) is a framework and distribution system for reusable PHP components.

This project provides a structured library of code, maintain a system for distributing code and for managing code packages & promote a standard coding style.

Date
21/02/2017

php.ini directives (variables)

display_errors : On/Off

error_reporting : 0/E_ALL

E_NOTICE, E_WARNING, etc.

0 - zero to report nothing

E_ALL - To report all level/type of errors

display_errors determines whether errors should be printed to the screen as part of the output or if they should be hidden from the user.

error_reporting determines to which level of errors to report and which not. The parameter is either an integer or a named constant (php.net/manual/en/errorfunc.constants.php).

Imp when we are not on production server i.e. if we are on development server/mode, we must use following 2 lines to get reported & display all type of errors.

```
<?php
error_reporting(E_ALL);
ini_set("display_errors", 1);
?>
```

The same as above is used in zend-framework project/public/index.php

---

Reference : www.pcds.co.in/java/how-to-get-useful-error-messages-in-php.php

Date
22/02/2017

Timezone

If we want to change our timezone then we will have to change the following line in xampp/php/php.ini file:

date.timezone = Europe/Berlin

to

date.timezone = Asia/Kolkata

We can see the changes made in phpinfo.php file on browser under 'date' section.

For every change in php.ini file, needs to restart Apache to make the changes effective on fleor.

Now all date/time functions in PHP script would use this timezone i.e. Asia/Kolkata-UTC+05:30 in place of Europe/Berlin-UTC+01:00.

Only php error log i.e. xampp/php/logs/php_error_log.log will still use Europe/Berlin timezone. This is a php bug. While apache error log i.e. xampp/apache/logs/error.log will also start using Asia/Kolkata timezone. PHP7 has solved this php error log bug.

We can also change timezone (for our project/application) in the code without touching the ini file, at the beginning of our code add:

```
date_default_timezone_set('Asia/Kolkata');
```

Reference: php.net/manual/en/timezones.php
(List of PHP supported Timezones)

Error log files & Configuration files

To make exist php error_log file, we will need to set following 2 directives (variables) in xampp/php/php.ini file:

```
error_log="D:\xampp\php\logs\php_error_log"
log_errors=On
```

Apache error log file location:
xampp/apache/logs/error.log

Always use 'XAMPP Control Panel' box to open and edit/view config files i.e. Apache config file/httpd.conf, PHP config file xampp/php/php.ini, xampp/apache/conf/ etc. and error log file i.e. Apache error log file xampp/apache/logs/error_log and php error log file xampp/php/logs/php_error_log.log etc. so that we can save ourselves to open & edit wrong config or error log files.

The above includes MySQL config file (xampp/mysql/bin/my.ini) & error log file (xampp/mysql/data/mysql_error.log) too.

Date 22/02/2017

Reference: stackoverflow.com/questions/23740548/how-to-pass-variables-and-data-from-php-to-javascript

How to Pass variables and data from PHP to Javascript!

First, let's understand the flow of events when a page is served from our server.

1. First PHP is run on server, it generates the HTML that is served to the client.
2. Then the HTML is delivered to the client after PHP is done with it. I would like to emphasize that once the code leaves the server - means PHP is done with it and can no longer access it.
3. Then the HTML with Javascript reaches to the client, which can execute JS on that HTML.

So really, the core thing to remember here is that "HTTP is stateless". Once a request left the server, the server can not touch it.

So, that leaves our options to:

1. Send more requests from the client after the initial request is done.
2. Encode (Get) what the server had to say in the initial request.

Solutions:

This is the core question we should be asking ourselves is:

Am I writing a website or an application?

Websites are mainly page based, and the page load times needs to be as fast as possible (for example - Wikipedia). Web applications are more AJAX heavy and perform a lot of round trips to get the client fast information (for example - a stock dashboard).

Reference: www.dyn-web.com/tutorials/php-js/

Date 22/02/2017

Passing PHP variables to Javascript

Here, we demonstrate & describe passing PHP variables containing simple data types to Javascript. We demonstrate with string, boolean and numeric values (i.e. scalar values).

To pass scalar data in a PHP variable ($val) to a Javascript variable, place the following in a Javascript segment:

```
var val = "<?php echo $val; ?>";
```

Notice the quotes around the PHP tags. This will result in a string value in Javascript. If the PHP value is numeric, we don't need to include the quotes.

Example:

Here we demonstrate with boolean, numeric & string values assigned to PHP variables:

```
<?php
$bool = false;
$num = 3 + 4;
$str = "A string here";
?>
```

We can output them into Javascript with the following:

```
<script type="text/javascript">
// boolean outputs "" if false, "1" if true
var bool = "<?php echo $bool; ?>";
// numeric value, both with & without quotes
var num = <?php echo $num; ?>; // 7
var str_num = "<?php echo $num; ?>"; // "7" (a string)
var str = "<?php echo $str; ?>"; // "A string here"
</script>
```

Date 23/02/2017

JSON ⇒ JavaScript Object Notation

The PHP json_encode function can be used to ~~resolve these problems~~ preserve data type of booleans and numbers.

Pass PHP Arrays to JSON & JS with json_encode:

The PHP json_encode function returns a string containing the JSON equivalent of the value passed to it. We will demonstrate here with a numerically indexed array:

```
<?php
$ar = array('apple','orange','banana','strawberry');
echo json_encode($ar);
// ["apple","orange","banana","strawberry"]
?>
```

We can pass the JSON string output by json_encode to a Javascript variable as follows:

```
<script type="text/javascript">
// pass PHP variable declared above to Javascript variable
var ar = <?php echo json_encode($ar); ?>;
</script>
```

A numerically indexed PHP array is translated to an array literal in the JSON string. A JSON_FORCE_OBJECT option can be used if you want the array to be output as an object instead:

```
<?php
echo json_encode($ar, JSON_FORCE_OBJECT);
// {"0":"apple","1":"orange","2":"banana","3":"strawberry"}
?>
```

Date 23/02/2017

PHP Interpreter ⟹ (means) Zend Engine

Notice that the JSON string contains no white space, which can make it difficult to read. A JSON_PRETTY_PRINT option can be used to format the JSON output.

Note: Javascript Array is/means only a numeric array as in PHP. Javascript object use 'named index' to access it's members like person.firstName. Javascript variables can be objects. Arrays are special kind of objects. Because of this, we can have variables of different types in the same Array.

Reference: https://www.w3schools.com/js/js_arrays.asp

Numerically Indexed Array Example:

This example demonstrates passing a numerically indexed PHP array consisting of string, numeric, boolean, and null values to json_encode. We echo the output of json_encode into a Javascript variable & the result is an array literal:

```
<?php
$ar = array('apple', 'orange', 1, false, null, true,
?>          3+5);
<script type="text/javascript">
var ar = <?php echo json_encode($ar); ?>;
// ["apple","orange",1,false,null,true,8];
// access 4th element in array
alert(ar[3]); // false
</script>
```

Date
23/02/24

we use an alert to demonstrate access to elements of the array. Notice that, here, the data types have been preserved.

Associative Array Example:

This example shows an associative PHP array output into Javascript using json_encode. Notice that, PHP's associative array becomes an object literal in Javascript:

```
<?php
$book = array(
    "title" => "Javascript: The Definitive Guide",
    "author" => "David Flanagan",
    "edition" => 6
);
?>
<script type="text/javascript">
var book = <?php echo json_encode($book, JSON_PRETTY_PRINT); ?>;
/* var book = {
    "title": "Javascript: The Definitive Guide",
    "author": "David Flanagan",
    "edition": 6
}; */
console.log(book);
alert(book.title);
</script>
```

We use the JSON_PRETTY_PRINT option as the second argument to json_encode to display the output in a readable format.

We can access object properties using dot syntax as

Date
23/02/2017

displayed with the alert included above, or square bracket syntax: book['title'].

Note: console.log() in javascript is equivalent to var_dump() in PHP. We can see the output of console.log() in chrome developer tool under "Console" tab. Under "Source" tab, we can see page source code including HTML, Javascript, and inline css. Under "Network" → "XHR" tab, we can see all ajax request/response details.

Using JavaScript's JSON.parse()

Javascript's JSON.parse method parses a JSON string and returns the Javascript equivalent.

Remember:

PHP numeric array ⇒ is equivalent to Javascript Array

PHP associative array ⇒ is equivalent to Javascript object

```
<?php
$numArr = array("apple", "rose", "table");   // PHP7 way of declaring array
$assArr = ["fruit"⇒"apple", "flower"⇒"rose", "furniture"⇒"table", "salary"⇒6];
?>
```

JSON representation of a numerically indexed array (equivalent to Javascript Array):

```
var json = '["apple", "rose", "table"]';   // string
```

JSON representation of a named indexed array (equivalent to Javascript object):

```
var json = '{"fruit": "apple", "flower": "rose", "furniture": "table", "salary": 6}';   // string
```

Date
23/02/2017

```
<?php $arr = ["apple", "rose", "table"]; } ⇒ ajax response
$json = json_encode($arr); ?>
```

```
<script type="text/javascript">
var json = JSON.stringify( ); var
```

ajax response

We pass the JSON string to JSON.parse() and assign the result to a data variable. We pass the data variable to console.log() to verify that the result is an array. Then we demonstrate access to an element of the data array with an alert.

```
<script type="text/javascript">
var json = '["apple","rose","table"]';
var data = JSON.parse(json);
console.log(data); //["apple","rose","table"]
alert(data[2]); // table
</script>
```

There may be situations where you are sure of your data and would not need to use JSON.parse. However, use of JSON.parse is advised if the data is coming from an untrusted source. JSON.parse checks the data passed to it and throws a SyntaxError if it's not valid JSON. Any data that could pose a risk is deleted.

Using JSON.parse() with PHP's json_encode():

The following demonstrates how to use JSON.parse() on the JSON string output by PHP's json_encode function:

```
<?php
$books = [
    [
        "title" => "title1";
```

Date 23/02/2017

var data = JSON.parse(json); </script>

```
        "author" => "author1"
    ],
    [
        "title" => "title2",
        "author" => "author2"
    ],
    [
        "title" => "title3",
        "author" => "author3"
    ]
];
?>
```

In the following Javascript segment, we show passing the PHP $books array to json_encode and the json_encode output to JSON.parse. The result is assigned to the Javascript variable books. We display the value of books commented out!

```
<script type="text/javascript">
// using JSON.parse on the output of json_encode
var books = JSON.parse('<?php echo json_encode(
                                $books); ?>');
/* output (with some whitespace added for readability)
[
    {"title": "title1", "author": "author1"},
    {"title": "title2", "author": "author2"},
    {"title": "title3", "author": "author3"}
]
*/
console.log(books[1].author); // author2
</script>
```

Date
23/02/2017

The PHP tags are enclosed in single quotes inside the JSON.parse function call since JSON uses double quotes for names & values. If you leave out the quotes or use double quotes, Javascript errors will be triggered.

JSON to PHP using json_decode()

PHP's json_decode function takes a JSON string and converts it into a PHP variable. Typically, the JSON data (i.e. JSON string) will represent a Javascript array or object literal which json_decode will convert into a PHP array or object. The following 2 examples demonstrate, first with an array & the 2nd with an object:

```
$json = '["apple", "orange", "banana", "strawberry"]';
$ar = json_decode($json);
//access first element of $ar array:
echo $ar[0]; //apple
```

By default, objects are converted to standard objects by json_decode:

```
$json = '{
            "title": "title1",
            "author": "author1",
            "edition" : 6
}';
$book = json_decode($json);
// access title of $book object
echo $book->title; // title1
```

Date
23/02/2019

The json_decode function provides an optional 2nd argument to convert object to associative array. On above example, now title & other elements can be accessed using array syntax:

```
// $json same as example object above
// pass true to convert objects to associative arrays:
$book = json_decode($json, true);
// access title of $book array
echo $book['title']; // title1
```

On previous page, a multidimensional array $books has been defined:

By default the result of json_decode will be a numerically indexed array of objects:

```
$json = '[
    {
        "title": "title1",
        "author": "author1"
    },
    {
        "title": "title2",
        "author": "author2"
    },
    {
        "title": "title3",
        "author": "author3"
    }
]';
$books = json_decode($json);
```

Date
23/02/2017

// access property of object in array
echo $books[1]->title; //title2

If we pass true as the 2nd argument to json_decode, the result is a multidimensional array that is numerically indexed at the outer level and associative at the inner level:

// $json same as example object above
// pass true to convert objects to associative arrays
$books = json_decode($json, true);
// numeric/associative array access
echo $books[1]['title']; // title2

Pass variables & data from PHP to JavaScript

There are actually several approaches to do this. Some require more overhead than others and some are considered better than others.

1. Use AJAX to get the data you need from the server.
2. echo the data into the page somewhere, and use Javascript to get the information from the DOM.
3. echo the data directly to Javascript

1. Use AJAX to get the data you need from the server.

This method is considered the best, because your server side and client side scripts are completely separate.

Pros:

1. Better separation between layers → If tomorrow

Date
23/02/2017

you stop using PHP, and want to move to a servelet, a REST API, or some other service, you don't have to change much of the Javascript code.

2. More readable → Javascript is Javascript, PHP is PHP. Without mixing the two, you get more readable code on both languages.

3. Allows for async data transfer → Getting the information from PHP might be time/resources expensive. Sometimes you just don't want to wait for the information, load the page, and have the information reach whenever.

4. Data is not directly found on the markup → This means that your markup (i.e. HTML) is kept clean of any additional data, and only Javascript sees it.

This means that, the data through AJAX can be seen only through Chrome developer tool box (which is also an event driven javascript box) & can not be seen through page view source (i.e. ctrl+u).

Cons:

1. Latency → AJAX creates an XML HTTP request & XML HTTP request are carried over network and have network latencies.

2. Put & get from DOM

This method is less preferable to AJAX, but it still has its advantages. It's still relatively

Date
23/02/2017

separated between PHP and JAVAscript in a sense that there is no PHP directly in the Javascript.

Pros:

Fast → Dom operations are often quick and you can store & access a lot of data relatively quickly.

Cons:

1. <input type="hidden"> can be used to store information, but doing so, means that we have a meaningless element in your HTML.
2. Data that PHP generates is outputted directly to the HTML source, meaning that we get a bigger & less focused HTML source.
3. Harder to get structured data
4. Tightly couples PHP to your data logic

Example: index.php

```
<div id="dom-target" style="display:none;">
  <?php $output=42; echo htmlspecialchars($output);
  // we have to escape because the result will not
  // be valid HTML otherwise
  ?>
</div>
<script>
  var div = document.getElementById("dom-target");
  var myData = div.textContent;
</script>
```

3. echo the data directly to Javascript:

This is probably the easiest to understand, and

Date
23/02/2017

the most horrible to use. Don't do this unless you know what you're doing.

Pros:

1. very easily implemented → It takes very little to implement this, and understand.
2. Does not dirty source → Variable are outputted directly to javascript, so the DOM is not affected.

Cons:

1. Insecure →
2. Harder to get structured data
3. Tightly couples PHP to your data logic

Example:

```
<script>
  var data = <?php echo json_encode("42"); ?>;
</script>
```

Date
24/02/2015

int preg_match(string $pattern, string $subject [, array &$matches])

Perform a regular expression match. It returns 1 if match found, 0 otherwise and FALSE if an error occured.

Example:

```
if(preg_match("/php/i", "PHP is the")){
    echo "A match was found";
}else{
    echo "A match was not found";
}
```

(/ or ~ or # — delimiters; i — case insensitive)

Output:

A match was found

Delimiters can be pretty much anything that is not alpha-numeric. Backslash (\) is not allowed. The most used are generally ~, / and #.

for

number abs(mixed $number)

Return the absolute value of number.

Example:

```
echo abs(-4.2);  // 4.2
echo abs(5);     // 5
echo abs(-5);    // 5
```

Date
24/02/2017

int printf(string $format [, mixed $args [, ...]])

Output a formatted string. It returns the length of the outputted string.

Example:

```
$xyz = printf("I am %d years %d month %s", 35, 2, 'old');
echo '<br>'.$xyz;
```

Output:

I am 35 years 2 month old
25

How a web server work, what is mod_rewrite, URL rewriting and pretty-links, Reg Ex

To understand what mod_rewrite does you first need to understand how a web server works. A web server responds to HTTP requests. An HTTP request at its most basic level looks like this:

```
GET /foo/bar.html HTTP/1.1
```

This is the simple request of a browser to a web server requesting the URL /foo/bar.html from it. It is important to stress that it does not request a file, it requests just some arbitrary URL. The request may also look like this:

```
GET /foo/bar?baz=42 HTTP/1.1
```

This is just as valid a request for a URL, and it has more obviously nothing to do with files.

The web server is an application listening on a port, accepting HTTP requests coming in on that port and returning a response. A web server is entirely free to respond to any request in any way it sees fit/in any way you have configured it to respond. This response is not a file, it's an HTTP response which may or may not have anything to do with physical files on any disk. A web server does not have to be Apache, there are many other web server which are all just programs which run persistently and are attached to a port which respond to HTTP requests. You can write one yourself if. This paragraph was intended to divorce you from any notion that URLs directly equal files, which is really important to understand (:).

Date
24/02/2017

The default configuration of most web servers is to look for a file that matches the URL on the hard disk. If the document root of the server is set to, say, /var/www, it may look whether the file /var/www/foo/bar.html exists and serve it if so. If the file ends in ".php" it will invoke the PHP interpreter and then return the result. All this association is completely configurable; a file does not have to end in ".php" for the web server to run it through the PHP interpreter, and the URL does not have to match any particular file on disk for something to happen.

mod_rewrite is a way to rewrite the internal request handling. When the web server receives a request for the URL /foo/bar, you can rewrite that URL into something else before the web server will look for a file on disk to match it. Simple example:

RewriteEngine On
RewriteRule /foo/bar /foo/baz

This rule says whenever a request matches "/foo/bar", rewrite it to "/foo/baz". The request will then be handled as if /foo/baz had been requested instead. This can be used for various effects. For Example:

RewriteRule (.*) $1.html

Date
24/02/2017

This rule matches anything (.*) and captures it ((..)), then rewrites it to append ".html". In other words, if /foo/bar was the requested URL, it will be handled as if /foo/bar.html had been requested. See (www.regular-expressions.info) for more information about regular expression matching, capturing and replacements.

Another often encountered rule is this:

RewriteRule (.*) index.php $url=$1

This, again, matches anything and rewrites it to the file index.php with the originally requested URL appended in the url query parameter i.e. for any and all requests coming in, the file index.php is executed and this file will have access to the original request in $_GET['url'], so it can do anything it wants with it.

What mod_rewrite does not do

mod_rewrite does not magically make all your URLs "pretty". This is a common misunderstanding. If you have this link in your website:

<a href="/my/ugly/link.php?is=not&very=pretty">

there is nothing mod_rewrite can do to make that pretty. In order to make this a ~~preety~~ pretty link. You have to:

1. Change the link to a pretty link:

<a href="/my/pretty/link">

Date
24/02/2017

2. Use mod_rewrite on the server to handle the request to the URL /my/pretty/link using any one of the methods described above.

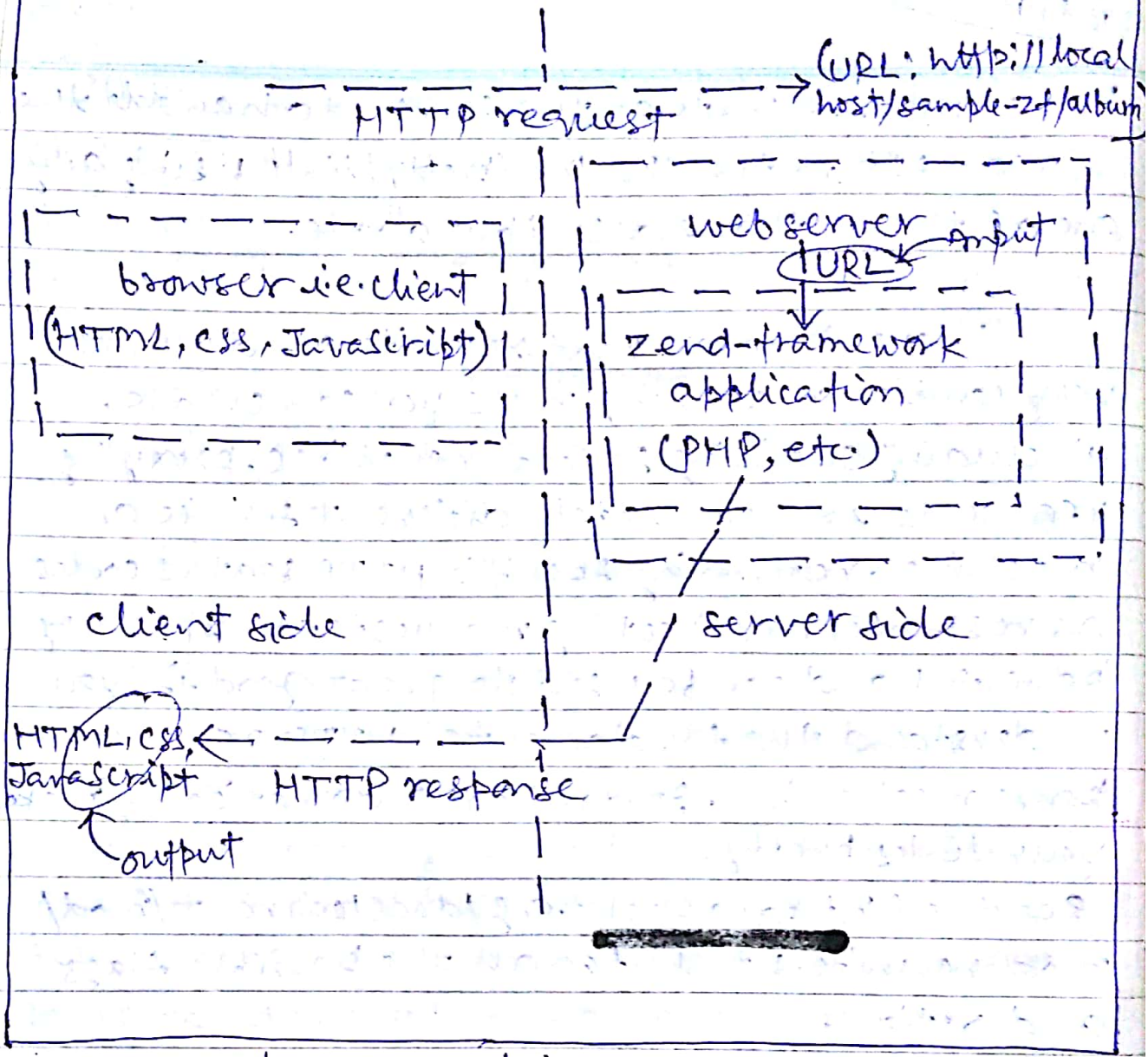
There is a lot mod_rewrite can do and very complex matching rules you can create, including chaining several rewrites, proxying requests to a completely different service or machine, returning specific HTTP status codes as responses, redirecting requests etc. It is very powerful and can be used to great good if you understand the fundamental HTTP request-response mechanism. It does not automatically make your links pretty.

See the (httpd.apache.org/docs/current/mod/mod_rewrite.html) for all the possible flags and options.

Reference: stackoverflow.com/questions/20563772/reference-mod-rewrite-url-rewriting-and-pretty-links-explained

More rewrite example ⇒ see the 2nd answer on this stackoverflow page

Next page ⇒ web server working Chart



web server working chart

Date
25/02/2015

PHP: Pass by reference vs. Pass by value

```
<?php
function pass_by_value($param){
    array_push($param, 3, 4);
}
$ar = array(1, 2, 3);
pass_by_value($ar);
foreach ($ar as $elem) {
    print "$elem ";
}
```

Output:

The code above prints 1 2 3. This is because the array is passed as value.

```
function pass_by_reference(&$param){
    array_push($param, 4, 5);
}
$ar = array(1, 2, 3);
pass_by_reference($ar);
foreach($ar as $elem) {
    print "$elem ";
}
```

Output:

The code above prints 1 2 3 4 5. This is because the array is passed as reference, meaning that the function pass_by_reference() does not manipulate a copy of the variable passed, but the actual variable itself.

In order to make a variable be passed by reference, it must be declared with a preceeding ampersand (&) in the function's declaration.

Date
25/02/2017

int sleep(int $seconds)

Delays/Halt the program execution for the given number of seconds. Return zero on success and FALSE on error.

Example:

```
// current time
echo date('H:i:s').'<br>';
// sleep for 10 seconds
sleep(10);
// wake up!
echo date('H:i:s');
```

output explanation is very important

Output (This example will produce output after 30 sec.

05:31:23
05:31:33

⇒ Both will be displayed exactly after 10 seconds. It will not that firstly 05:31:23 displays & then after 10 seconds 05:31:33 will be displayed.

As we know how a web server work. The web server firstly collect the HTTP response by executing the whole PHP code on server side in one go & then send the HTTP response to the client (side) in form of HTML, CSS, JavaScript (browser)

where firstly javascript execute/run & then HTML ~~interpretation~~ interpretation starts/loads by browser.

Date
25/02/2017

bool set_time_limit (int $seconds)

Limits/sets the maximum execution time for a program/page.

As, by default, max_execution_time sets 30 seconds in php.ini. So, if we pass 25 as argument in set_time_limit(25) on a page, then the script can run a total of 30+25=55 seconds before timing out, after it a fatal error would be generated.

---

bool time_sleep_until (float $timestamp)

make the script sleep until the specified time.

Exa: time_sleep_until(time()+3); //sleep for 3 seconds
time_sleep_until(time()+0.2); //sleep for 0.2 seconds

---

void usleep (int $micro_seconds)

Delay/Halt program execution for the given number of microseconds. No value is returned.

Exa: usleep(2000000); //wait for 2 seconds.

Date
02/03/2017

Refresh a page after every time interval of 5 seconds:

```
<head>
  <meta http-equiv="refresh" content="5">
</head>
```

API (application program interface)

An application program interface (API) is code that allows two software programs to communicate with each other. APIs are implemented by function calls.

Example:

Following is an jQuery api (on Gogle CDN):

```
<head>
  <script src="https://ajax.googleapis.com/ajax/
    libs/jquery/3.1.1/jquery.min.js"></script>
</head>
```

Interface ⇒ a program that controls a display for the user (usually on a computer monitor) and that allows the user to interact with the system.

An application programming interface (API) is a software program which is designed for use by other software program. Whereas a normal software program is used by a (human) computer user, an API is a software program which is used by another software program.

Date
20/03/2017

Some famous terms

HTTP/HTTPS: The Hypertext Transfer Protocol (HTTP) is a networking protocol for distributed information systems. HTTP is the foundation of data communication for the World Wide Web/ Internet. Most website URLs include the http:// protocol at the beginning of the address, however the https:// protocol may also be used for transferring data with encryption.

FTP: File Transfer Protocol (FTP) is a standard network protocol used to copy a file from one host (network/internet connected computer) to another over a TCP-based network, such as the internet. FTP is the standard method used for transferring files to a website or server and FTP is supported/offered by almost all web hosting companies/providers. If you are using a web hosting company/provider then they should supply you with your FTP login details when you sign-up, FTP details consist of a host address (a URL or IP address), a username and a password. Many web hosting providers offer control panels which allow you to manage your FTP accounts, and they often provide online file management tools allowing you to upload files to your website using your web browser.

HTML: Hypertext Markup Language (HTML) is the predominant markup language for web pages, and is often called 'the building-blocks of the web'.

Date
20/03/2017

HTML is used to create almost all web pages, including the page you are viewing now, and popular websites including Google, Bing and Yahoo. Other languages and technologies may be used in addition to HTML, these include Javascript and CSS/Stylesheets. HTML documents are viewed/read using a web browser such as Microsoft Internet Explorer or Mozilla Firefox.

markup - मार्कप - The amount added to the cost price of goods to cover overheads and profit.
Markup refers to the sequence of chasacters or other symbols that you insert at certain places in a text or word processing file to indicate how the file should look when it is printed or displayed. The markup indicators are often called "tags".

hypertext - हाइपरटेक्स्ट - machine-readable text that is not sequential but is organized so that related items of information are connected.

MongoDB: MongoDB is an open source, scalable, high-performance, schema-free, document-oriented database. Being 'schema-free' means MongoDB works differently from MySQL allowing for software to store data in a more natural way, whilst still being able to quickly search the data. दस्तावेज़ (डेटा प्रबंधन)

Date
23/03/2017

PHP-FIG

It stands for ⇒ PHP Framework Interoperability Group

PHP-FIG is a group of established PHP projects whose goal is to talk about commonalities between their projects and find ways they can work better together.

PSR

It stands for ⇒ PHP Standards Recommendations

Coding Styles: Standardized formatting of coding styles reduces the cognitive friction when reading codes from other authors.
PSR-1: Basic Coding Standard
PSR-2: Coding Style Guide

Autoloading: It removes the complexity of including files by mapping namespaces to file system paths.
PSR-4: Improved Autoloading

Reference: http://www.php-fig.org/

Date
23/08/2012

Reference : devdocs.magento.com/guides/v2.1/ext-best-practices/extension-coding/common-programming-bp.html

Programming Best Practices

Follow a set of coding standards : Following a set of coding standards will help make your code consistent and maintainable.

Magento's Coding Standards are based on the following:

1. Zend coding standards
   https://framework.zend.com/manual/1.12/en/coding-standard.html
2. PSR-2
   http://www.php-fig.org/psr/psr-2/
3. PSR-4
   http://www.php-fig.org/psr/psr-4/

→ a. PHP File Formatting
  b. Naming Conventions
  c. Coding Style

Date
29/03/2017

Design Patterns in PHP

Singleton Pattern: There are several scenarios in which you might want to make your constructor private. The common reason is that in some cases, you don't want outside code to call your constructor directly, but force it to use another method to get an instance of your class.

Or

The purpose of using a private constructor ensures that there can be only one instance of a class and provides a global access point to that instance and this is common with The Singleton Pattern.

We only ever want a single instance of our class to exist:

```
class Singleton
{
    private static $instance = null;

    private function __construct()
    {
    }

    public static function getInstance()
    {
        if (self::$instance === null) {
            self::$instance = new self();
        }
        return self::$instance;
    }
}
```

Date
30/03/2017

3rd Party API integration (use) in our application:

Look: hotelogix/application/models/CurrencyConverter

Payment Gateway integration into our application:

Look: hotelogix/application/models/Gateway

3rd Party library use in our application

To use third-party library in our application, we should use adapter.

Adapter: Adapters are the classes follow the adapter pattern (a software design pattern also known as 'wrapper') and wrap-around (- रैपराउन्ड- -चारो तरफ से ढकना/लपेटना/cover करना) classes from third-party libraries. These classes allow you to use functionality from third-party libraries in your code by converting the third-party class interfaces into an interface that is expected by your native code.

When to use Adapter: You should always use adapter classes instead of directly using classes from third-party libraries. This reduces the change impact on your code when the API changes in a third-party library.

We recommend using adapter classes to get access to the functionality provided by third-party classes.

How to write Adapter: A common approach in developing an adapter is to create an interface

Date 30/03/2017

Reference: devdocs.magento.com/guides/v2.1/extension-dev_guide/adapters.html

named 'AdapterInterface' to describe the functionality the third-party class provides. This class is typically found in a directory labeled 'Adapter'. Classes implementing this adapter interface use the third-party class directly to provide indirect functionality.

This approach allows you to update or substitute different implementations provided by other third-party classes without the need to update code that uses your adapter.

Example of Adapters:

1. magento2/vendor/magento/framework/Code/Minifier

   Here, magento has created 2 adapters. First Minifier/Adapter/Css/CssMin.php to minify css & second Minifier/Adapter/Js/JShrink.php to minify javascript.

2. magento2/vendor/magento/framework/Image

   Here, magento has created 2 adapters. First Adapter/Gd2.php & second Adapter/ImageMagick.php. Both adapter classes provides the concrete implementation using the third-party libraries.

   Gd2 : It is a library used to create & manipulate image files in a variety of different image formats, including GIF, PNG, JPEG, WBMP & XPM. It is enabled by default in PHP7.

   ImageMagick : It is a native php extension to create & modify images using the ImageMagick API.

Date
31/03/2017

Plugin/Module/Component

Plug-in applications are programs that can easily be installed and used as part of your application.

Example:

magento2/vendor/magento/module-customer
magento2/vendor/magento/module-cms
are different magento2 components and you can also create a new one for yourself.
In same way, Drupal has a lot of modules & Joomla has a lot of plugins.

Wrappers or Wrapper classes

Wrap - रैप - ढकना/कवर करना/कवर करना
Magento has several wrapper classes.
Example: For superglobal variables
Reference: devdocs.magento.com/guides/v2.1/ext-best-practices/extension-coding/security-performance-data-bp.html
Magento recommends to not directly use any PHP superglobals such as:
$GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_COOKIE, $_SESSION, $_REQUEST, $_ENV
Instead use the
Magento\Framework\HTTP\PhpEnvironment\Request
i.e.
magento2/vendor/magento/framework/HTTP/PhpEnvironment/Request.php
wrapper class to safely access these values.

PHP functions to avoid

The following is a list of PHP functions that are known to be vulnerable and exploitable. Avoid using these functions in your code.

eval, serialize/unserialize, md5, srand, mt_srand

Reference: devdocs.magento.com/guides/v2.1/extension-dev-guide/security/non-secure-functions.html

or

Magento website → Documentation & Resources → Magento Developer Documentation (Magento 2.1) → Development (PHP Developer Guide) → Security (Non-secure functions)

Date
01/04/2017

Debugging Javascript with Google Chrome Developer Tools:

We can debug our javascript code through Google Chrome developer tools very efficiently.
Here are some terms to understand:

ctrl + r → (In browser) This ensures that reload the javascript file rather than using the cached version.

Avoid to use Global variable in javascript. So that subsequent included javascript file can't override the variable to produce some error.

Don't use ==, != (compares only value)
Always use ===, !== (compares value & type both)

"Source" Tab ⇒ Breakpoints, conditional breakpoints
Scope (Local & Global),
Watch (Add variable to watch),
Call Stack,
control moves from one breakpoint to another breakpoint. ↳ ▷ → Resumes script execution
↷ → Step over (control moves line by line within the function)
↓ → Step in (Jumps from function usages to fun definition)
↑ → Step out
Snippets

"Console" Tab ⇒ Live edit javascript,
console.log(), console.clear(),
↑ like var_dump() in PHP
console.assert(expression, object),
console.table(),
multiline commands → shift + ↵ (to write 2nd line of code)

(You can assume that these some line of javascript code would be kept within <script> tag at last line of current browser created HTML page to execute)

Date
01/04/2017

"Elements" Tab ⇒ Live edit HTML

"Network" Tab ⇒ Page load performance & Ajax

Reference Videos: http://blog.arjindra.com/javascript/debugging-javascript-with-google-chrome-developer-tools/

Date
09/09/2017

Ideal Page load time in Page performance

The ideal page load time/speed for our website's HTML to be less than 1.5 seconds.

1. If the HTML page load time is less than 1.5 seconds (top 20% of pages on the web), there are some performance tool, shows/indicates no errors with loading the page.
2. If the HTML page load time is between 1.5-5 seconds or longer than 5 seconds there will be a warning message indicating that the page is not loading at an ideal speed.

Note: Google provides Google page speed test/insight tool to test performance of a web page.

Date 05/04/2017

Reference: http://blog.ajinasa.com/php/xdebug-php-debugging/

Xdebug Installation

The Xdebug extension helps you debugging your PHP script by providing a lot of valuable debug information i.e. Xdebug is a PHP extension which provides debugging and profiling capabilities.

To install Xdebug, go to URL:
https://xdebug.org/download.php
Under RELEASES heading, there is a link 'custom installation instructions', click on it.
Here, paste your phpinfo html view source code & click on button 'Analyse my phpinfo() output' and now follow the instructions, specially designed for you.

→ mandatory*, rest are optional

```
[XDebug]
zend_extension = D:\xampp\php\ext\php_xdebug-2.5
                 .1-7.0-VC14.dll
;xdebug.profiler_enable=0          → xdebug.profiler_enable_trigger=1
xdebug.profiler_output_dir="D:\xampp\tmp"
;xdebug.profiler_output_name="cachegrind.out.%t
                              -%s"
xdebug.remote_host="127.0.0.1"     → localhost
xdebug.remote_log="D:\xampp\tmp\xdebug.txt"
xdebug.trace_output_dir="D:\xampp\tmp"
; 3600 (1 hour), 36000=10h
;xdebug.remote_cookie_expire_time=36000
xdebug.remote_enable=on            ← most imp.
xdebug.remote_handler="dbgp"
```

make output_buffering=off in php.ini

imp { IDE key & remote_port must be same in php.ini & in Netbeans configuration

IDEKey=netbeans-xdebug
xdebug.remote_port=9000

Date: 05/04/2017

Reference: My Google Drive (jitendra.research) → Companies → Webvirtue → xdebug_test

Configure NetBeans for xdebug

Go to NetBeans & follows as below:

Tools → Options → General → Web Browser
set "Chrome" and click on "OK" button.

Tools → Options → PHP → Debugging
set Debugger Port: 9000
Session ID: netbeans-xdebug
} as in php.ini

and click on "OK" button.

Always use phpinfo page i.e. localhost/phpinfo.php to look php.ini variables

Configure project in NetBeans

There is a project D:\xampp\htdocs\2017\xdebug_test having only one file index.php

To configure this project in NetBeans, follows as below:

File → New Project
Choose Categories: PHP
Projects: PHP Application with Existing Sources
and click on "Next>" button

→ 1. Choose Project

2. Name and Location
Choose Sources Folder: D:\xampp\htdocs\2017\xdebug_test
Project Name: xdebug_test
PHP Version: PHP 7.0
Default Encoding: UTF-8
And click on "Next>" button.

Now, we may come on 3rd step "Run Configuration" & configure it, but we will do it in another

more appropriate way. So in 2nd step, instead of click on "Next>" button, click on "Finish" button.

Now in NetBeans, at left hand side under "Projects", right click on "xdebug_test" & click on "Properties". Click on 2nd link "Run Configuration":
Configuration: Clicking on "New" button, you can create different configuration names for different settings for the same project.
For Example:
I have created 2 configuration for my project xdebug_test.
1. Configuration: <default>
with following settings:
Run As: Local Web Site (running on local web server)
Project URL: http://localhost/2017/xdebug_test/
Index File: index.php

2. Configuration: xdebugTest ← created a new one
with following settings:
Run As: PHP Built-in Web Server (running on built-in web server)
Hostname: localhost
Port: 8000
Router Script: index.php (keep it blank)
and click on "OK" button to create a new one configuration.
X Don't do this, otherwise for each & every debug action, control will go to index.php file first.

Date
05/04/2017

Start debugging project with Xdebug

Go to NetBeans and in left hand side, select/click on project "xdebug_test" under "Projects" tab.
On header of NetBeans, click on "Debug" & then "Debug Project (xdebug_test)".
As a result, a browser will have been opened with URL:

① http://localhost:8000/?XDEBUG_SESSION_START=netbeans-xdebug

Now back on NetBeans.
Look carefully at bottom in NetBeans:

② netbeans-xdebug [running] ☒

③ The first line of PHP code background, will become green.

The above three indicates that, debugging with xdebug has been started successfully.
Now you can create "Breakpoints" & debug your code with following NetBeans buttons in Header:

▷ ⤵ ⤓ ⤒ ↻

When you complete your debugging, click on ☒ button at NetBeans bottom with netbeans-xdebug [running] to end xdebug session.

Note: To start PHP's ~~in-built server~~ built-in web server, go to in directory xdebug_test (project dir.) and right click & select "Git Bash Here". Write following command & enter (↵):

```
php -S 0.0.0.0:8000 ↵
```

Now, You can access your project by:

Date
05/04/2017

http://localhost:8080/
instead of
http://localhost/2017/xdebug_test/

Remember, after making any change in php.ini you have to terminate PHP's built-in web server by (ctrl + c) command & restart again the same by php -S 0.0.0.0:8080 command.
↑zero

In this case XAMPP control panel box - Apache stop/start button will not work to reflect php.ini changes.
Remember one more thing, during xdebug session, means when there in NetBean's bottom appears:
netbeans-xdebug [running] ☒
then running your project i.e. hit your project URL in browser http://localhost:8080/ will also work under xdebug session & you will find your browser in loading mode as xdebug focuses it's control on first line of your project/php-file and it remains the same until you do not use xdebu g button in NetBeans header ▷ ⇩ ⇧ ⇩ to move the control forward to reach end of your php-file/ project representing the URL http://localhost:8080/.

Date
06/04/2017

Debugging a project with xdebug

To start debugging project xdebug_test:
Go to NetBeans:
Select/Click the project xdebug_test on left hand side → In header, click on "Debug" tab & then click/select "Debug Project (xdebug_test)"
(As a result, xdebug session starts & xdebug control transfers to 1st line of your index file)

or

Select/Click the project xdebug_test on left hand side → In header, click on button  i.e. (Ctrl+F5)

or

Right click to the project xdebug_test on left hand side and select/click "Debug".

Suppose, you have completed your debugging for the flow using above one of the three ways & you again want to debug the same flow then don't do the above process because your xdebug-session is already started when you debugged first time now follow the following:

netbeans-xdebug 

Select/Click the project xdebug_test on left hand side → In header, click on button  i.e. F6

or

Right click to the project xdebug_test on left hand side and select/click "Run".

or

Select/Click the project xdebug_test on left hand side → In header, click on "Run" tab & then click/select "Run Project (xdebug_test) F6"

Date
06/04/2017

Debug a single file in project xdebug_test

There are 2 files in xdebug_test project:

1. DisplayForm.php (only HTML content)
2. DisplayFormAction.php
   (Web form action page)

Select/Click the file DisplayForm.php on left hand side → In header, click on "Debug" tab & then click/select "Debug File"

or

Right click to the file DisplayForm.php on left hand side and select/click "Debug".

There are five buttons in NetBeans header:

Continue (F5) → Runs from one Breakpoint to another breakpoint.

② → Step Over (F8) → Goes to the next PHP line in current file. Does not go into method calls made within current context.

③ → Step in (F7) → debug the code line by line

④ → Step Out (Ctrl+F7) → Return to the previous file/function/execution flow point). It is just opposite of 'Step into (F7)'.

⑤ → Run to Cursor (F4) → jump the debugging to cursor point & then jump to new cursor point.

Step into (F7) → Goes to the next PHP line in current execution, not just the file. Dives into any method calls made, giving far more detail than Step Over.

Now xdebug control focuses after last line of HTML code of DisplayForm.php & browser still will not show the webform of DisplayForm.php. Click on button ▷ or ⤓ to end debugging of file DisplayForm.php, browser page load completes and

Date
06/04/2017

web form looks completely. Fill the data in web form & submit the same but page browser shows only in-process form-submission but does not happen anything, this is because xdebug starts on page DisplayFormAction.php & xdebug-control moves to first line of DisplayFormAction.php. Now use [↓] button to debug line by line. When it ends, browser loads the complete page for DisplayFormAction.php.

Close xdebug session

In NetBeans bottom:

netbeans-xdebug [running] [X]

↑ To end the xdebug session, click on this cross button.

U → timestamp in microseconds
H → HTTP HOST
R → Request URL

Date 08/04/2017

Reference: http://blog.ajinkya.com/php/profiling-with-xdebug-and-qcachegrind/

Profile an application with xdebug and QCacheGrind (KCacheGrind)

Profiling - प्रोफाइलिंग - recording an application's (a person's) behaviour and analyzing (psychological) characteristics in order to predict their ability

For profiling with xdebug, we will have to enable xdebug's profiler first as following:

[XDebug] ← (php.ini)

```
zend_extension="D:\xampp\php\ext\php_xdebug-2.5.1-7.0-vc14.dll"
xdebug.remote_enable=on
xdebug.remote_log="D:\xampp\tmp\xdebug.txt"
;
xdebug.profiler_enable_trigger=1
xdebug.profiler_output_dir="D:\xampp\tmp"
xdebug.profiler_output_name="cachegrind.out.%t.%p.%H_%R"
;
xdebug.trace_output_dir="D:\xampp\tmp"
```

When you completes the above updation in your php.ini, restart apache to get effect of the same. Now whenever you want to profile of an URL, just pass XDEBUG_PROFILE or XDEBUG_PROFILE=1 with this URL in form of GET/POST.

Example: To profile of URL http://magento.local/, hit in browser the URL http://magento.local/?XDEBUG_PROFILE. As a result, there will create a file in dir "D:\xampp\tmp" with name like "cachegrind.out.1491567650_097794_magento_local__XDEBUG_PROFILE" to describe the URL (http://magento.local/) profiling.

Date
08/04/2017

Xdebug created the profiling data (i.e. the file) but to visualize this data we need a tool/software are named "QCacheGrind".

QCacheGrind: QCacheGrind is a tool/software to visualize the xdebug created profiling data. Just search with "QCacheGrind" in google and click on first link i.e.
http://sourceforge.net/projects/qcachegrindwin/
Click on "Download" button to download the tool. Unzip the downloaded file & renamed the folder by "qcachegrind", put this folder in "D:\xampp". No need of installation., just double click on "D:\xampp\qcachegrind\qcachegrind.exe" & open the profiling data file to visualize the same. You can read the file "D:\xampp\qcachegrind\readme" to get usages detail of this software.

Date
26/04/2019

Reference: devdocs.magento.com/guides/v2.1/get-started/qs-web-api-response.html

HTTP Status codes

1. HTTP code: 200
Meaning: Success
Description: The framework returns HTTP 200 to the caller upon success.

2. HTTP code: 400
Meaning: Bad Request
Description: If service implementation throws either Magento_Service_Exception or its derivative, the framework returns a HTTP 400 with an error response including the service specific error code and message. This error code could indicate a problem such as a missing required parameter or the supplied data didn't pass validation.

3. HTTP code: 401
Meaning: Unauthorized
Description: The caller was not authorized to perform the request. For example, the request included an invalid token or a user with customer permissions attempted to access an object that requires administrator permissions.

4. HTTP code: 403
Meaning: Forbidden ← मना, वर्जित
Description: Access is not allowed for reasons that are not covered by error code 401. In easy words, this status code means that accessing the page or resource you were trying to reach is absolutely forbidden for some reason.

Date
26/04/2017

5. HTTP code: 404
Meaning: Not found
Description: The specified REST endpoint does not exist. The caller can try again.

6. HTTP code: 405
Meaning: Not allowed
Description: A request was made of a resource using a method that it is not supported by that resource. For example, using GET on a form which requires data to be presented via POST, or using PUT on a read-only resource.

7. HTTP code: 406
Meaning: Not acceptable
Description: When the browser sends a request, it will dispatch an Accept header. The browser client will specify the characteristics of the data that it accepts. There are many types of accept headers including accept-charset, accept-encoding and accept-ranges. The Accept header informs the server about the format in which the browser wants to accept the data. The server will return the HTTP error 406 not acceptable if it is unable to send data in the requested format.

8. HTTP code: 500
Meaning: System Errors
Description: If service implementation throws any other exception like network errors, database communication, framework returns HTTP 500.

Date
27/04/2017

OAuth

OAuth is an open standard for authorization, commonly used as a way for internet users to authorize websites or applications to access their information on other websites but without giving them the passwords. This mechanism is used by companies such as Google, Facebook, Microsoft and Twitter to permit the users to share information about their accounts with third party applications or websites.

Date
15/05/2017

Archive - आर्काइव - ऐतिहासिक अभिलेख
Retention - रिटेन्शन - The capacity to hold or retain
Normal

Data Archiving

Data archiving is the process of moving data that is no longer actively used to a separate storage device for long-term retention. Archive data consists of older data that is still important to the organization and may be needed for future reference. Data archives are indexed and have search capabilities so files and parts of files can be easily located and retrieved.

Date 25/05/2019

Reference: https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

S.O.L.I.D.: The First 5 Principles of Object Oriented Programming & Design

SOLID are five basic principles which help to create good software architecture. SOLID is an acronym where: ऐक्रनिम

S → stands for SRP (Single responsibility principle)
O → stands for OCP (Open/Closed principle)
L → stands for LSP (Liskov substitution principle) लिस्कोव सब्स्टिट्यूशन
I → stands for ISP (Interface segregation principle) सेग्रेगेशन
D → stands for DIP (Dependency inversion principle) इन्वर्शन

1. Single responsibility principle: A class should have ~~only~~ one and only one reason to change, meaning that a class should have only one job.

For Ex: We have some shapes and we wanted to sum all the areas of the shapes. Well this is pretty simple right?

```
class Circle
{
    public $radius;
    public function __construct($radius)
    {
        $this->radius = $radius;
    }
}

class Square
{
```

Date 25/05/20

Reference: blog.agindra.com/php/solid-principles/

```
    public $length;
    public function __construct($length)
    {
        $this->length = $length;
    }
}
```

First, we create our shapes classes and have the constructors setup the required parameters. Next we move on by creating the AreaCalculator class and then write up our logic to sum up the areas of all provided shapes.

```
class AreaCalculator
{
    protected $shapes;
    public function __construct($shapes = array())
    {
        $this->shapes = $shapes;
    }
    public function sum()
    {
        //logic to sum the areas
    }
    public function output()
    {
        return 'Sum of the areas of provided
        shapes: '.$this->sum();
    }
}
```

To use the AreaCalculator class, we simply instan

Date
25/05/20

tiate the class and pass in an array of shapes, and display the output at the bottom of the page.

```
$shapes = array(
    new Circle(2),
    new Square(5),
    new Square(6)
);
$areas = new AreaCalculator($shapes);
echo $areas->output();
```

The problem with the output method is that the AreaCalculator handles the logic to output the data. Therefore, what if the user wanted to output the data as json or something else?

All of that logic would be handled by the AreaCalculator class, this is what SRP against; the AreaCalculator class should only sum the areas of provided shapes, it should not care whether the user wants json or HTML.

So, to fix this you can create an SumCalculatorOutputter class and use this to handle whatever logic you need to handle how the sum areas of all provided shapes are displayed.

The SumCalculatorOutputter class would work like this:

```
$shapes = array(
    new Circle(2),
    new Square(5),
    new Square(6)
);
```

Date
25/05/2017

```
$areas = new AreaCalculator($shapes);
$output = new SumCalculatorOutputter($areas);
echo $output->JSON();
echo $output->XML();
echo $output->HTML();
echo $output->JADE();
```

Now, whatever logic you need to output the data to the user is now handled by the SumCalculatorOutputter class.

2. Open/Closed principle: Objects or entities should be open for extension, but closed for modification.

This simply means that a class should be easily extendable without modifying the class itself. Let's take a look at the AreaCalculator class, especially it's sum method.

```
public function sum()
{
    foreach($this->shapes as $shape) {
        if(is_a($shape, 'Square')) {
            $area[] = pow($shape->length, 2);
        } else if(is_a($shape, 'Circle')) {
            $area[] = pi() * pow($shape->radius, 2);
        }
    }
    return array_sum($area);
}
```

Date 25/05/2017

If we wanted the sum method to be able to sum the areas of more shapes, we would have to add more if/else blocks and that goes against the open/closed principle.

A way we can make this sum method better is to remove the logic to calculate the area of each shape out of the sum method and attach it to the shape's class.

```
class Sqaure
{
    public $length;
    public function __construct($length)
    {
        $this->length = $length;
    }
    public function area()
    {
        return pow($this->length, 2);
    }
}
```

The same thing should be done for the circle class, an area method should be added. Now, to calculate the sum of any shape provided should be as simple as:

```
public function sum()
{
    foreach($this->shapes as $shape){
        $area[] = $shape->area();
    }
    return array_sum($area);
}
```

Date
25/05/2017

Now we can create another shape class and pass it in when calculating the sum without breaking our code. However, now another problem arises, how do we know that the object passed into the AreaCalculator is actually a shape or if the shape has a method named area?
Coding to an interface is an integral part of S.O.L.I.D., a quick example is we create an interface, that every shape implements:

```
interface ShapeInterface
{
    public function area();
}
class Circle implements ShapeInterface
{
    public $radius;
    public function __construct($radius)
    {
        $this->radius = $radius;
    }
    public function area()
    {
        return pi() * pow($this->radius, 2);
    }
}
```

In our AreaCalculator sum method we can check if the shapes provided are actually instances of the ShapeInterface, otherwise we throw an exception:

Date
25/05/20

```
public function sum()
{
    foreach ($this->shapes as $shape) {
        if (is_a($shape, 'ShapeInterface')) {
            $area[] = $shape->area();
            continue;
        }
        throw new AreaCalculatorInvalidShapeException;
    }
    return array_sum($area);
}
```

3. Liskov substitution principle: Every subclass/derived class should be substitutable for their base/parent class.

Example:

```
class A
{
    public function fire() { }
}
class B extends A
{
    public function fire() { }
}
function doSomething(A $obj)
{
    // do something with it
}
```

B

LSP says that if we start use of B instead of

Date 07/06/2019

A in function do something, there should not emerge any problem & all processes should work smoothly as previously.

Example:

```
class VideoPlayer
{
    public function play($file)
    {
        //play the video
    }
}
class AviVideoPlayer extends VideoPlayer
{
    public function play($file)
    {
        if(pathinfo($file, PATHINFO_EXTENSION) !==
            'avi')
        {
            throw new Exception; //violates the LSP
        }
    }
}
```

Here highlighted code is responsible to violate the LSP. As VideoPlayer class play method has same execution & return type of all types of video players but it's child class AviVideoPlayer play method throw exception for non-avi video players.

Date
07/06/2017

Example:

```
interface LessonRepositoryInterface
{
    /**
     * Fetch all records
     *
     * @return array
     */
    public function getAll();
}
class FileLessonRepository implements LessonRepositoryInterface
{
    public function getAll()
    {
        //return through filesystem
        return [];
    }
}
class DbLessonRepository implements LessonRepositoryInterface
{
    public function getAll()
    {
        return Lesson::all(); //violates the LSP
    }
}
function foo(LessonRepositoryInterface $lesson)
{
}
```

return Lesson::all()->toArray();

Here we can use both class object i.e. either FileLessonRepository or DbLessonRepository.

Date
07/06/2019

Here highlighted code in DbLessonRepository class violates the L.S.P. As first child class FileLessonRepository return array which is ok but second child class DbLessonRepository of LessonRepositoryInterface returns Collection class object which is wrong. All child classes of same interface must return same type of output.

Principle

4. Interface Segregation: A client should never be forced to implement an interface that it does not use.

or

Clients should not be forced to depend on methods they do not use.

Example:

```
interface ManageableInterface
{
    public function beManaged();
}
interface WorkableInterface
{
    public function work();
}
interface SleepableInterface
{
    public function sleep();
}
```

Date
07/06/2017

```
class HumanWorker implements WorkableInterface,
    SleepableInterface, ManageableInterface
{
    public function work()
    {
        //implement work method
    }
    public function sleep()
    {
        return 'human sleeping';
    }
    public function beManaged()
    {
        $this->work();
        $this->sleep();
    }
}
class AndroidWorker implements WorkableInterface,
    ManageableInterface
{
    public function work()
    {
        return 'android working';
    }
    public function beManaged()
    {
        $this->work();
    }
}
```

Date 07/06/2017

decouple - डिकपल - अलग करना

```
class Captain
{
    public function manage(ManageableInterface
                                $worker)
    {
        $worker->beManaged();
    }
}
```

5. Dependency Inversion Principle: Entities must depend on abstractions not on concretions. It states that the high level module must not depend on the low level module; but they should depend on abstractions.
This principle allows for decoupling, following example seems like the best way to explain this principle.

```
class PasswordReminder
{
    private $dbConnection;
    public function __construct(MySQLConnection
        $dbConnection)
    {
        $this->dbConnection = $dbConnection;
    }
}
```

First the MySQLConnection is the low level module while the PasswordReminder is high level, but according to the definition of D in S.O.L.I.D. which states that 'Depend on Abstraction not on concreti

Date 07/06/2017

ons', this snippet above violates this principle as the PasswordReminder class is being forced to depend on the MySQLConnection class.
Later if you were to change the database engine (for ex. from MySQL to Oracle), you would also have to edit the PasswordReminder class and thus violates Open-closed principle.
The PasswordReminder class should not care what database your application uses, to fix this we "code to an interface", since high level and low level modules (both) should depend on abstraction, we can create an interface:

```
interface DBConnectionInterface
{
    public function connect();
}
```

The interface has a connect method and the MySQLConnection class implements this interface, also instead of directly type-hinting MySQLConnection class in the constructor of the PasswordReminder, we instead type-hint the interface and no matter the type of database your application uses, the PasswordReminder class can easily connect to the database without any problems and OCP is not violated.

lowlevel code

```
class MySQLConnection implements DBConnectionInterface
{
    public function connect
    {
```

Date
07/06/2017

```
        return "Database connection";
    }
}
```

high level code

```
class PasswordReminder
{
    private $dbConnection;
    public function __construct(DBConnectionInterface $dbConnection)
    {
        $this->dbConnection = $dbConnection;
    }
}
```

According to the little snippet above, you can now see that both the high level and low level modules/code depend on abstraction.

High level code is not as concerned with details.

Low level code is more concerned with details and specifics.

Date
28/06/2017

[redacted], Cautions:

(1) Set max_execution_time ⇒ 30 minutes i.e. 1800 in xampp/php/php.ini

(2) In xampp/apache/conf/httpd.conf, set AllowOverride ⇒ All instead of none

(3) In xampp/apache/conf/extra/httpd-xampp.conf, set AllowOverride ⇒ All instead of none

Note: Always use xampp interface to open above configuration files & edit.

Use xampp shell editor when you want to execute any command on cmd with "php".
Ex: php xyz.php ↵
php -v ↵

Otherwise, you will have to set environment variable for PHP:
window → search "System" → Advanced system settings → Environment Variables → system variables → edit "PATH" variable or create new one with name "PATH" if this variable does not exist. Add the value for it is:
D:\xampp\php;
and restart the system to make visible the changes.

To terminate a command execution in window CMD, use ctrl+c (same as in linux) or ctrl+Pause/Break

Date
04/01/2018

Symlink (Symbolic link or Soft link)

In computing, a symlink is the nickname for any file that contains a reference to another file or directory in the form of an absolute or relative path and that affects pathname resolution.

The following command creates a symbolic link at the command-line interface (shell):

ln -s target_path link_path

target_path is the relative or absolute path to which the symbolic link should point. Usually the target will exist, although symbolic links may be created to non-existent targets. link_path is the path of the symbolic link.

Reference : https://en.wikipedia.org/wiki/Symbolic_link
https://www.cybercitibiz/faqr/creating-soft-link-or-symbolic-link/