CET218

Advanced Web Programming

03 - Arrays and Files

Dr. Ahmed Said

Start →

Arrays

```php
$name = array();                                # create
$name = array(value0, value1, ..., valueN);
$name[index]                                    # get element value
$name[index] = value;                           # set element value
$name[] = value;                                # append
```

```php
$a = array();                                   # empty array (length 0)
$a[0] = 23;                                     # stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Appened!";                                 # add string to end (at index 5)
```

- **Append**: use bracket notation without specifying an index
- Element type is not specified
- can mix types

Array Functions

Function	Description
`array_shift`	Removes the first element from an array and returns it.
`array_pop`	Removes the last element from an array and returns it.
`array_push`	Adds one or more elements to the end of an array.
`array_reverse`	Returns an array with elements in reverse order.
`sort`	Sorts an array in ascending order.
`array_slice`	Extracts a slice of the array, returning the selected elements.
`array_merge`	Merges one or more arrays into one array.
`array_diff`	Computes the difference of arrays.
`array_intersect`	Computes the intersection of arrays.
`array_map`	Applies a callback function to the elements of the given arrays.
`array_filter`	Filters elements of an array using a callback function.
`array_reduce`	Iteratively reduces the array to a single value using a callback function.
...	...

Array Function : Example

- The array in PHP **replaces many other collections** in languages like C++, C#, Java
  - List, Stack, Queue, Set, Map, ...

```php
$tas = array("MD", "BH", "KK", "HM", "JP");
for ($i = 0; $i < count($tas); $i++) {  $tas[$i] = strtolower($tas[$i]); }
$morgan = array_shift($tas);
array_pop($tas);
array_push($tas, "ms");
array_reverse($tas);
sort($tas);
$best = array_slice($tas, 1, 2);
```

- Output?

```text
Array ( [0] => md [1] => bh [2] => kk [3] => hm [4] => jp )
md
Array ( [0] => bh [1] => kk [2] => hm )
Array ( [0] => bh [1] => kk [2] => hm [3] => ms )
Array ( [0] => ms [1] => hm [2] => kk [3] => bh )
Array ( [0] => bh [1] => hm [2] => kk [3] => ms )
Array ( [0] => hm [1] => kk )
```

`print_r` Function

- The `print_r` function in PHP is used to print human-readable information about a variable.
- It is particularly useful for debugging.

```php
print_r(mixed $expression, bool $return = false): string|null
```

- `$expression` : The variable to be printed.
- `$return` : If set to true, `print_r` will return the information rather than print it

```php
$array = array('apple', 'banana', 'cherry');
print_r($array);
```

```php
$array = array('apple', 'banana', 'cherry');
$output = print_r($array, true);
echo "The array contains: \n" . $output;
```

`foreach` Loop

- The `foreach` loop is used to iterate over arrays in PHP.

```php
foreach ($array as $variableName) {
    ...
}
```

```php
$fellowship = array("Frodo", "Sam", "Gandalf", "Strider", "Gimli", "Legolas", "Boromir");
echo "The fellowship of the ring members are: \n";
for ($i = 0; $i < count($fellowship); $i++) {
    echo "{$fellowship[$i]}\n";
}
echo "The fellowship of the ring members are: \n";

foreach ($fellowship as $fellow) {
    echo "$fellow\n";
}
```

Multidimensional Arrays

- A multidimensional array is an array that contains one or more arrays.
- PHP supports multidimensional arrays that are two, three, four, five, or more levels deep.

```php
<?php $AmazonProducts = array( array(“BOOK", "Books", 50),
                        array("DVDs", “Movies", 15),
                        array(“CDs", “Music", 20)
                );
for ($row = 0; $row < 3; $row++) {
    for ($column = 0; $column < 3; $column++) { ?>
         <p> | <?= $AmazonProducts[$row][$column] ?>
    <?php } ?>
      </p>
<?php } ?>
```

Multidimensional Arrays (cont.)

```php
<?php $AmazonProducts = array( array(“Code” =>“BOOK", “Description” => "Books", “Price” => 50),
                        array(“Code” => "DVDs", “Description” => “Movies", “Price” => 15),
                        array(“Code” => “CDs", “Description” => “Music", “Price” => 20)
                    );
for ($row = 0; $row < 3; $row++) { ?>
    <p> | <?= $AmazonProducts[$row][“Code”] ?> | <?= $AmazonProducts[$row][“Description”] ?> | <?= $AmazonProducts[$row][“Price”] ?
      </p>
<?php } ?>
```

- output ?

```html
| BOOK | Books | 50
| DVDs | Movies | 15
| CDs | Music | 20
```

`strcmp` function

- The `strcmp` function compares two strings.
- It returns:
  - `0` if the two strings are equal
  - `a positive number` if the first string is greater than the second
  - `a negative number` if the first string is less than the second

```php
$var1 = "Hello";
$var2 = "hello";
echo strcmp($var1, $var2);
echo strcmp($var2, $var1);
echo strcmp($var1, $var1);
```

```text
1
-1
0
```

- Variations with non case sensitive functions `strcasecmp`

Embedded PHP

Printing HTML tags in PHP = bad style

```php
<?php
print "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.1//EN\"\n";
print " \"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd\">\n";
print "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n";
print " <head>\n";
print " <title>Geneva's web page</title>\n";
...
for ($i = 1; $i <= 10; $i++) {
  print "<p> I can count to $i! </p>\n";
}
?>
```

- best PHP style is to minimize `print` / `echo` statements in embedded PHP code
- but without `print` or `echo`, how do we insert dynamic content into the page?

PHP expression blocks

- `PHP expression block` : a small piece of PHP that evaluates and embeds an expression's value into HTML
- They can be used anywhere in the HTML document

```php
<?= expression ?>
```

```php
<h2> The answer is <?= 6 * 7 ?> </h2>
```

```text
The answer is 42
```

- <?= expression ?> is equivalent to:

```php
<?php echo expression; ?>
```

Common errors

```php
...
<body>
<p>Watch how high I can count:
<?php
for ($i = 1; $i <= 10; $i++) {
?>
    <? $i ?>
</p>
</body>
</html>
```

- The above code will not work because the PHP block is not closed properly
- The `<? $i ?>` is missing the `=` sign to print the value of `$i`
- if you forget to close your braces, you'll see an error about 'unexpected $end' if you forget = in <?=, the expression does not produce any output

Complex expression blocks

```php
...
<body>
<?php
for ($i = 1; $i <= 3; $i++) {
    ?>
    <h<?= $i ?>>This is a level <?= $i ?> heading.</h<?= $i ?>>
    <?php
}
?>
</body>
```

- The above code will generate three headings with different levels

This is a level 1 heading.

This is a level 2 heading.

This is a level 3 heading.

Functions

Functions in PHP

- A function is a block of code that can be called to perform a specific task.
- Functions are defined using the `function` keyword.
- Functions can take parameters and return values.

```php
function functionName($parameter1, $parameter2, ...) {
    // code to be executed
}
```

```php
function quadratic($a, $b, $c) {
    return -$b + sqrt($b * $b - 4 * $a * $c) / (2 * $a);
}
```

- parameter types and return types are not written
- a function with no return statements implicitly returns NULL

Default Parameter Values

```php
function print_separated($str, $separator = ", ") {
    if (strlen($str) > 0) {
        print $str[0];
        for ($i = 1; $i < strlen($str); $i++) {
            print $separator . $str[$i];
        }
    }
}
```

```php
print_separated("hello"); # h, e, l, l, o
print_separated("hello", "-"); # h-e-l-l-o
```

Practical PHP

Using `printf`

- You've already seen the `print` and `echo` functions, which simply output text to the browser.
- But a much more powerful function, `printf`, controls the format of the output by letting you put special formatting characters in a string.
- The `printf` function is used to format strings in PHP.
- For each formatting character, `printf` expects you to pass an argument that it will display using that format.
- For instance, the following example uses the `%d` conversion specifier to display the value 3 in decimal:

```php
printf("There are %d items in your basket", 3);
```

- If you replace the `%d` with `%b`, the value 3 will be displayed in binary (`11`).

Using printf (cont.)

- The following table lists some of the most common conversion specifiers:

Specifier	Description
`%b`	Binary number
`%c`	ASCII character
`%d`	Signed decimal number
`%e`	Scientific notation (e.g., 1.2e+2)
`%u`	Unsigned decimal number
`%f`	Floating-point number
`%o`	Octal number
`%s`	String
`%x`	Hexadecimal number (lowercase)
`%X`	Hexadecimal number (uppercase)
`%%`	A literal percent sign

Using `printf` (cont.)

- You can have as many specifiers as you like in a printf function, as long as you pass a matching number of arguments and as long as each specifier is prefaced by a `%` symbol.

```php
printf("My name is %s. I'm %d years old, which is %X in hexadecimal",
'Simon', 33, 33);
```

- A more practical example of printf sets colors in HTML using decimal values.

```php
printf("<font color=\"#%X%X%X\">%s</font>", 65, 127, 245, "Hello, world!");
```

```html
<span style="color: #417FF5">Hello, world!</span>
```

Using `printf` (cont.)

Precision Setting

- Not only can you specify a conversion type, but you can also set the precision of the displayed result.
- The following example uses the `%f` conversion specifier to display the value 3.14159 to two decimal places:

```php
printf("The value of pi is approximately %.2f", 3.14159);
```

```html
The value of pi is approximately 3.14
```

Using `printf` (cont.)

```php
<?php
    echo "<pre>"; // Enables viewing of the spaces
    // Pad to 15 spaces
    printf("The result is $%15f\n", 123.42 / 12);
    // Pad to 15 spaces, fill with zeros
    printf("The result is $%015f\n", 123.42 / 12);
    // Pad to 15 spaces, 2 decimal places precision
    printf("The result is $%15.2f\n", 123.42 / 12);
    // Pad to 15 spaces, 2 decimal places precision, fill with zeros
    printf("The result is $%015.2f\n", 123.42 / 12);
    // Pad to 15 spaces, 2 decimal places precision, fill with # symbol
    printf("The result is $%'#15.2f\n", 123.42 / 12);
?>
```

```html
The result is $ 10.285000
The result is $00000010.285000
The result is $ 10.29
The result is $000000000010.29
The result is $##########10.29
```

Using `printf` (cont.)

String Padding

- You can also pad strings to required lengths (as you can with numbers), select different padding characters, and even choose between left and right justification.

```php
echo "<pre>"; // Enables viewing of the spaces
$h = 'Web';
printf("[%s]\n", $h); // Standard string output
printf("[%12s]\n", $h); // Right justify with spaces to width 12
printf("[%-12s]\n", $h); // Left justify with spaces
printf("[%012s]\n", $h); // Pad with zeros
printf("[%'#12s]\n\n", $h); // Use the custom padding character '#'
$d = 'Rasmus Lerdorf'; // The original creator of PHP
printf("[%12.8s]\n", $d); // Right justify, cutoff of 8 characters
printf("[%-12.12s]\n", $d); // Left justify, cutoff of 12 characters
printf("[%-'@12.10s]\n", $d); // Left justify, pad with '@', cutoff 10 chars
```

```html
[Web]
[         Web]
[Web         ]
[000000000Web]
```

Using `sprintf`

- The `sprintf` function is similar to `printf`, but instead of outputting the formatted string, it returns it.
- This is useful if you want to store the formatted string in a variable or use it in a function.

```php
$hexstring = sprintf("%X%X%X", 65, 127, 245);
```

Or you may wish to store output in a variable for other use or display:

```php
$out = sprintf("The result is: $%.2f", 123.42 / 12);
echo $out;
```

Date and Time Functions

- To keep track of the date and time, **PHP uses standard Unix timestamps**, which are simply the number of seconds since the start of January 1, 1970.
- To determine the current timestamp, you can use the time function:

```php
echo time();
```

- Because the value is stored as seconds, to obtain the timestamp for this time next week, you would use the following code,
  - which adds 7 days × 24 hours × 60 minutes × 60 seconds to the returned value:

```php
echo time() + 7 * 24 * 60 * 60;
```

Date and Time Functions (cont.)

- If you wish to create a timestamp for a given date, you can use the `mktime` function.

```php
echo mktime(0, 0, 0, 12, 1, 2022);
```

- Its output is the timestamp 1669852800 for the first second of the first minute of the first hour of the first day of December in the year 2022:
- The parameters to pass are, in order from left to right:
  - The number of the hour (0–23)
  - The number of the minute (0–59)
  - The number of seconds (0–59)
  - The number of the month (1–12)
  - The number of the day (1–31)
  - The year (1970–2038, or 1901–2038 with PHP 5.1.0+ on 32-bit signed systems)

Date and Time Functions (cont.)

- To display the date, use the date function, which supports a plethora of formatting options enabling you to display the date any way you wish.
- The format is as follows:

```php
date($format, $timestamp);
```

```php
echo date("l F jS, Y - g:ia", time());
```

- The output will be something like this:

```html
Wednesday December 1st, 2022 - 2:00pm
```

Date and Time Functions (cont.)

Character	Description
`d`	Day of the month, 2 digits with leading zeros
`D`	A textual representation of a day, three letters
`j`	Day of the month without leading zeros
`S`	English ordinal suffix for the day of the month, 2 characters
`F`	A full textual representation of a month
`m`	Numeric representation of a month, with leading zeros
`M`	A short textual representation of a month, three letters
`n`	Numeric representation of a month, without leading zeros
`t`	Number of days in the given month
`Y`	A full numeric representation of a year, 4 digits
`y`	A two-digit representation of a year

Using `checkdate`

- The `checkdate` function is used to validate a Gregorian date.
- It returns `true` if the date is valid and `false` if it is not.

```php
$month = 9; // September (only has 30 days)
$day = 31; // 31st
$year = 2025; // 2025
if (checkdate($month, $day, $year)) echo "Date is valid";
else echo "Date is invalid";
```

- Valid or not ?
- The above code will output `Date is invalid` because September only has 30 days.

die and exit Functions

- The `die` and `exit` functions are used to terminate the execution of a script.
- They are often used to display an error message and stop the script from running.

```php
$month = 9; // September (only has 30 days)
$day = 31; // 31st
$year = 2025; // 2025
if (!checkdate($month, $day, $year))
die("Invalid date");
```

- `die` is an alias of `exit` and they are used interchangeably.

File Handling

- Powerful as it is, `MySQL` is not the only (or necessarily the best) way to store all data on a web server.
- Sometimes it can be quicker and more convenient to directly access files on the hard disk.
- Cases in which you might need to do this are when modifying images such as uploaded user avatars or with logfile that you wish to process.
- First, though, a note about file naming: if you are writing code that may be used on various PHP installations, there is no way of knowing whether these systems are casesensitive.
- For example, Windows and macOS filenames are not case-sensitive, but Linux and Unix filenames are.

Therefore, you should always assume that the system is case-sensitive and stick to a convention such as all-lowercase filenames.

File Handling (cont.)

Checking for File Existence

- The `file_exists` function is used to check whether a file exists.
- It returns `true` if the file exists and `false` if it does not.

```php
if (file_exists("file.txt")) echo "File exists";
else echo "File does not exist";
```

File Handling (cont.)

Creating a File

- The `fopen` function is used to open a file.
- It returns a file handle that can be used to read, write, or append to the file.

```php
$fh = fopen("testfile.txt", 'w') or die("Failed to create file");
$text = <<<_END
Line 1
Line 2
Line 3
_END;
fwrite($fh, $text) or die("Could not write to file");
fclose($fh);
echo "File 'testfile.txt' written successfully";
```

- Should a program call the `die` function, the open file will be automatically closed as part of terminating the program.
- When you run this in a browser, all being well, you will receive the message File 'testfile.txt' written successfully. If you receive an error message, your hard disk may be full or, more likely, you may not have permission to create or write to the file.

File Handling (cont.)

- This simple example shows the sequence that all file handling takes:
  1. Always start by opening the file. You do this through a call to `fopen`.
  2. Then you can call other functions; here we write to the file (`fwrite`), but you can also read from an existing file (`fread` or `fgets`) and do other things.
  3. Finish by closing the file (`fclose`). Although the program does this for you when it ends, you should clean up by closing the file when you're finished.
- Every open file requires a file resource so that PHP can access and manage it.

> **Note**
>
> Don't worry about the content of the $fh variable; it's a number PHP uses to refer to internal information about the file—you just pass the variable to other functions.

File Handling (cont.)

- Upon failure, FALSE will be returned by `fopen`.
- Notice the second parameter to the `fopen` call. It is simply the character w, which tells the function to open the file for writing. The function creates the file if it doesn't already exist.

> **Note**
>
> Be careful when playing around with these functions: if the file already exists, the `w` mode parameter causes the `fopen` call to delete the old contents (even if you don't write anything new!).

File Handling (cont.)

- There are several different mode parameters that can be used here, including:

Mode	Description
`r`	Open for reading only; place the file pointer at the beginning of the file
`r+`	Open for reading and writing; place the file pointer at the beginning of the file
`w`	Open for writing only; place
`w+`	Open for reading and writing;
`a`	Open for writing only; place
`a+`	Open for reading and writing;
`x`	Create and open for writing only;
`x+`	Create and open for reading and writing;

Reading from a File

- The easiest way to read from a text file is to grab a whole line through fgets (think of the final s as standing for string),

```php
$fh = fopen("testfile.txt", 'r') or
die("File does not exist or you lack permission to open it");
$line = fgets($fh);
fclose($fh);
echo $line;
```

- If you created the file, you'll get the first line:

```text
Line 1
```

Reading from a File

- You can retrieve multiple lines or portions of lines through the `fread` function, as in

```php
$fh = fopen("testfile.txt", 'r') or
die("File does not exist or you lack permission to open it");
$text = fread($fh, 3);
fclose($fh);
echo $text;
```

- We have requested three characters in the fread call, so the program displays this:

```text
Lin
```

> **Note**
>
> The `fread` function is commonly used with binary data. If you use it on text data that spans more than one line, remember to count newline characters.

Copying a File

- The `copy` function is used to copy a file.
- It returns `true` if the file is copied successfully and `false` if it is not.

```php
copy('testfile.txt', 'testfile2.txt') or die("Could not copy file");
echo "File successfully copied to 'testfile2.txt'";
```

- if you don’t want your programs to exit on a failed copy attempt

```php
if (copy("testfile.txt", "testfile2.txt")) echo "File copied";
else echo "File not copied";
```

Moving a File

- The `rename` function is used to move a file.
- It returns `true` if the file is moved successfully and `false` if it is not.

```php
if (!rename('testfile2.txt', 'testfile2.new'))
echo "Could not rename file";
else echo "File successfully renamed to 'testfile2.new'";
```

- You can use the rename function on directories, too. To avoid any warning messages if the original file doesn’t exist, you can call the file_exists function first to check.

Deleting a File

- Deleting a file is just a matter of using the unlink function to remove it from the filesystem,

```php
if (!unlink('testfile2.new')) echo "Could not delete file";
else echo "File 'testfile2.new' successfully deleted";
```

> **Note**
>
> Whenever you access files on your hard disk directly, you must also always ensure that it is impossible for your filesystem to be compromised. For example, if you are deleting a file based on user input, you must make absolutely certain it is a file that can be safely deleted and that the user is allowed to delete it.

Updating Files

- You can use one of the append write modes.
- The file pointer is the position within a file at which the next file access will take place, whether it's a read or a write.
- To update a file, you can open it in read/write mode (r+ or w+), read the contents, and then write the new contents back to the file.

```php
$fh = fopen("testfile.txt", 'r+') or die("Failed to open file");
$text = fgets($fh);
fseek($fh, 0, SEEK_END);
fwrite($fh, "\n$text") or die("Could not write to file");
fclose($fh);
echo "File 'testfile.txt' successfully updated";
?>
```

- the `fgets` function to read in a single line from the file (up to the first line feed)
- The `fseek` function is called to move the file pointer right to the file end

Locking Files for Multiple Accesses

- Web programs are often called by many users at the same time. If more than one person tries to write to a file simultaneously, it can become corrupted. And if one person writes to it while another is reading from it, the file is all right, but the person reading it can get odd results.
- To prevent this, you can lock a file so that only one person can access it at a time.

```php
$fh = fopen("testfile.txt", 'r+') or die("Failed to open file");
$text = fgets($fh);
if (flock($fh, LOCK_EX))
{
fseek($fh, 0, SEEK_END);
fwrite($fh, "$text") or die("Could not write to file");
flock($fh, LOCK_UN);
}
fclose($fh);
echo "File 'testfile.txt' successfully updated";
```

Note

There is a trick to file locking to preserve the best possible response time for your website visitors: perform it directly before a change you make to a file, and then unlock it immediately afterward. Having a file locked for any longer than this will slow down your application unnecessarily

Reading an Entire File

- The `file_get_contents` function is used to read an entire file into a string.
- It returns the contents of the file as a string.

```php
echo "<pre>"; // Enables display of line feeds
echo file_get_contents("testfile.txt");
echo "</pre>"; // Terminates <pre> tag
```

- This function is actually a lot more useful than that, because you can also use it to fetch a file from a server across the internet.

```php
echo file_get_contents("http://google.com");
```

PHP Include File

- Insert the content of one PHP file into another PHP file before the server executes it
- The `include` statement is used to include a file in the current script.
- The `require` statement is used to include a file in the current script and will generate a fatal error if the file is not found.

```php
include 'header.php';
```

- The `include_once` statement is used to include a file in the current script only once.

```php
include_once 'header.php';
```

PHP Include File (cont.)

```php
// menu.php
<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/contact.php">Contact Us</a>
```

```php
// index.php
<html>
<body>

<div class="leftmenu">
<?php include("menu.php"); ?>
</div>

<h1>Welcome to my home page.</h1>
<p>I have a great menu here.</p>

</body>
</html>
```

THANK YOU