

CTIS 256 Web Technologies II

Note # 3

Serkan GENÇ

Arrays - functions

- bool `is_array($arr)`
- bool `in_array($item, $arr)`, bool `array_key_exists($item, $arr)`
- mixed `array_search($item, $arr)`
- int `array_push(&$arr, $item)`, mixed `array_pop(&$arr)`
- int `array_unshift(&$arr, $item)`, mixed `array_shift(&$arr)`
- array `array_merge($arr1, $arr2, ...)`
- array `array_slice($arr, $start, $length)`
- array `array_splice(&$arr, $start, $length, $insertArr)`
- bool `sort(&$arr)`, bool `rsort(&$arr)`
- bool `asort(&$arr)`, bool `arsort(&$arr)`
- bool `ksort(&$arr)`, bool `krsort(&$arr)`
- bool `usort(&$arr, callbackFn)`
- array `array_filter($arr, callbackFn)`
- array `array_fill($start_index, $num, $value)`
- array `range($low, $high, $step)`
- array `array_values($arr)`, array `array_keys($arr)`
- int `extract($arr, $extract_rule, $prefix)`
- bool `shuffle(&$arr)`
- string `http_build_query($arr)`

```
$num = range(0, 10, 2) ; // array from 0 to 10 with increment 2.
shuffle( $num ) ; // shuffles the array.
print join($num, ' ') . '<br>' ;
echo in_array(4, $num) ? '4 exists' : 'not Available' , '<br>' ;
array_push($num, 1, 3, 5, 7, 9) ;
sort( $num ) ;
print join($num, ' ') . '<br>' ;
echo ($pos = array_search(4, $num)) ? "4 at $pos" : 'not' , '<br>' ;
array_unshift( $num, 11, 12, 13) ;
$num = array_merge($num, [14, 15, 16]) ;
rsort($num) ;
print join($num, ' ') . '<br>' ;
array_splice($num, 4, 3 , [-1, -2, -3, -4, -5]) ; //delete and insert
print join($num, ' ') . '<br>' ;
array_splice($num, 9) ; // delete items after index 9.
print join($num, ' ') . '<br>' ;

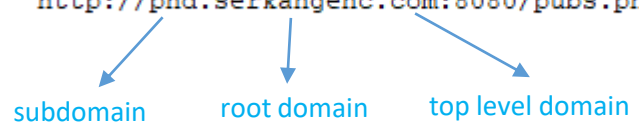
$users = [ ['name'=>'veli', 'id' => 125],
            ['name' => 'can', 'id'=>111],
            ['name' => 'ali', 'id'=>171] ] ;
// sort based on id.
usort($users, function($a, $b){
    return $a['id'] <=> $b['id'] ;
});
$person = $users[0];
extract( $person) ; // $name=can, $id=111
echo array_key_exists('name', $person) ? 'name exists' : 'not','<br>';
```

URL – URL Encoding

Structure of a URL:

scheme://**user:password**@host:**port**/path?**queryString**#**fragment** ■ = optional

http://phd.serkangenc.com:8080/pubs.php#journals



subdomain root domain top level domain

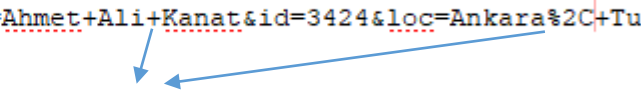
```
'scheme' => string 'http' (length=4)
'host'   => string 'phd.serkangenc.com' (length=18)
'port'   => int 8080
'path'   => string '/pubs.php' (length=9)
'fragment' => string 'journals' (length=8)
```

ftp://sgenc:test@ftp.bilkent.edu.tr:22/publications

```
'scheme' => string 'ftp' (length=3)
'host'   => string 'ftp.bilkent.edu.tr' (length=18)
'port'   => int 22
'user'   => string 'sgenc' (length=5)
'pass'   => string 'test' (length=4)
'path'   => string '/publications' (length=13)
```

Query String is used to send data within URL or content of POST packet, it is composed of key and value pairs separated by & (ampersand). **Values** must be encoded in URL friendly format (*application/x-www-form-urlencoded*). In this format, space is encoded to (+) sign, and all non-alphanumeric characters (ASCII) except -_. have been replaced with a percent(%) sign followed by two hex digits. For non-ASCII characters, their UTF8 encoding is used such as "ç" => %C3%A7

http://www.social.net/project/add.php?name=Ahmet+Ali+Kanat&id=3424&loc=Ankara%2C+Turkey



urlencoded values in query string

```
'scheme' => string 'http' (length=4)
'host'   => string 'www.social.net' (length=14)
'path'   => string '/project/add.php' (length=16)
'query'  => string 'name=Ahmet+Ali+Kanat&id=3424&loc=Ankara%2C+Turkey'
```

URL-related Functions

```
$urlEncoded = urlencode("Konu: Ahmet Çağıl Konuk, Mehmet Dinç@Ankara") ;  
var_dump($urlEncoded) ;
```

 `string urlencode(string)`

```
'Konu%3A+Ahmet+%C3%87a%C4%9F%C4%B1l+Konuk%2C+Mehmet+Din%C3%A7@Ankara'
```

```
$value = urldecode('Konu%3A+Ahmet+%C3%87a%C4%9F%C4%B1l+Konuk%2C+Mehmet+Din%C3%A7@Ankara') ;  
var_dump($value) ;
```

 `string urldecode(string)`

```
'Konu: Ahmet Çağıl Konuk, Mehmet Dinç@Ankara'
```

```
// Build URL with data
```

```
$data = [  
    'name' => 'Ahmet Demir',  
    'id' => 43526,  
    'location' => 'Ankara, Turkey',  
    'friends' => 'Ali&Neşe&Mehmet'  
] ;
```

```
$urlencoded_queryString = http_build_query($data) ;  
var_dump($urlencoded_queryString) ;
```

 `string http_build_query(array)`

it builds/generates query string from an associative array, it uses urlencode() for values.

```
'name=Ahmet+Demir&id=43526&location=Ankara%2C+Turkey&friends=Ali%26Ne%C5%9Fe%26Mehmet'
```

```
// to parse (break down) URL into meaningful parts.
```

```
$url = 'http://www.social.net/project/add.php?name=Ahmet+Ali+Kanat&id=3424&loc=Ankara%2C+Turkey' ;  
$url_parts = parse_url($url) ;  
var_dump($url_parts) ;
```

 `array parse_url(string)`

it parses a URL into its logical parts

```
C:\wamp64\www\test\url_related_string.php:32:
```

```
array (size=4)  
  'scheme' => string 'http' (length=4)  
  'host' => string 'www.social.net' (length=14)  
  'path' => string '/project/add.php' (length=16)  
  'query' => string 'name=Ahmet+Ali+Kanat&id=3424&loc=Ankara%2C+Turkey'
```

URL-related Functions

```
// to parse (break down) URL into meaningful parts.
```

```
$url = 'http://www.social.net/project/add.php?name=Ahmet+Ali+Kanat&id=3424&loc=Ankara%2C+Turkey' ;
```

```
$url_parts = parse_url($url) ;
```

```
var_dump($url_parts) ;
```



array **parse_url**(string)

it parses a URL into its logical parts

```
C:\wamp64\www\test\url_related_string.php:32:
```

```
array (size=4)
```

```
'scheme' => string 'http' (length=4)
```

```
'host' => string 'www.social.net' (length=14)
```

```
'path' => string '/project/add.php' (length=16)
```

```
'query' => string 'name=Ahmet+Ali+Kanat&id=3424&loc=Ankara%2C+Turkey'
```



Decompose query string

```
// parse query string into key-value pairs
```

```
parse_str( $url_parts['query'], $qs) ;
```

```
var_dump($qs) ;
```



void **parse_str**(string, &array)

Reverse of http_build_query(), it uses urldecode() for values.

```
C:\wamp64\www\test\url_related_string.php:28:
```

```
array (size=3)
```

```
'name' => string 'Ahmet Ali Kanat' (length=15)
```

```
'id' => string '3424' (length=4)
```

```
'loc' => string 'Ankara, Turkey' (length=14)
```

User-defined Function

- Function is a way to repeat the same functionality without duplicating the codes.
- A function is usually composed of "name", "argument(s)", "content" and "return value (output)".
- Arguments can be input (call by value) or output (call by reference)
- Primitive types and arrays are passed by value if there is no **&** before the argument. However, all objects are passed by reference by default.
- Function names have global scope.
- Variable number of argument is possible.
`func_get_args()` stores arguments in an array.
- An argument can take a default value.

```
<?php
```

```
$num1 = 5; $num2 = 7;
$result = sum( $num1, $num2) ; // 22

// Simple function definition, can be defined anywhere
// in the file. $n3 has a default value of 10.
function sum( $n1, $n2, $n3 = 10) {
    $total = $n1 + $n2 + $n3 ; // local variable
    return $total ;
}

$result = sumVarArg(3, 4, 5, 2, 1) ; // 15
// function with a variable number of arguments.
function sumVarArg() {
    $total = 0 ;
    foreach( func_get_args() as $num) {
        $total += $num;
    }
    return $total ;
}

// Function with input and output arguments.
// $inp1 and $inp2 => INPUT arguments called by value
// $add, $diff, $mul => OUTPUT, called by reference
function doArith($inp1, $inp2, &$amp;add, &$amp;diff, &$amp;mul) {
    $add = $inp1 + $inp2 ;
    $diff = $inp1 - $inp2 ;
    $mul = $inp1 * $inp2 ;
}
doArith(5, 6, $a, $b, $c) ; // a=11, b=-1 c=30

$arr = array(1,2,3,4) ;
// $arr is sent by value.
function change($arr) {
    $arr[0] = 15 ;
}
print $arr[0] ; // still 1
```

Variable Scopes

- Variable Scopes: **Super Global**, **Global**, **Local** and **Static**
- **Super global** : Accessible from any scope : `$_SERVER`, `$_ENV`, `$_POST`, `$_GET`, `$_REQUEST`, `$_FILES`, `$_COOKIE`, `$_SESSION`, and `$_GLOBALS`
- **Global** : defined outside of the functions, accessible from anywhere. From function, use `$_GLOBAL` array or `global` keyword.
- **Local** : defined within function, its scope is limited by the function where it is defined. They are created when the function is called and deleted when the function exits.
- **Static** : Life time is global (not deleted after the execution of the function), scope is limited by the function. It is initialized once within the function.

```
// SUPER GLOBAL and LOCAL
function currentFile() {
    // $_SERVER is super global, $file is local variable.
    $file = $_SERVER["PHP_SELF"] ; // .
    return "$file is the current filename" ;
}

print currentFile() ; // displays this php file name.

// GLOBAL
$peopleList = array ( "Can" => 8, "Ozan" => 5, "Özge" => 7, "Aslı" => "2") ;
function findClass($person){
    global $peopleList ; // declare as global variable otw. it creates local.
    // OR use $GLOBAL['peopleList'] without global keyword.
    if ( isset($peopleList[$person])) {
        return $peopleList[$person];
    }
    return -1 ;
}

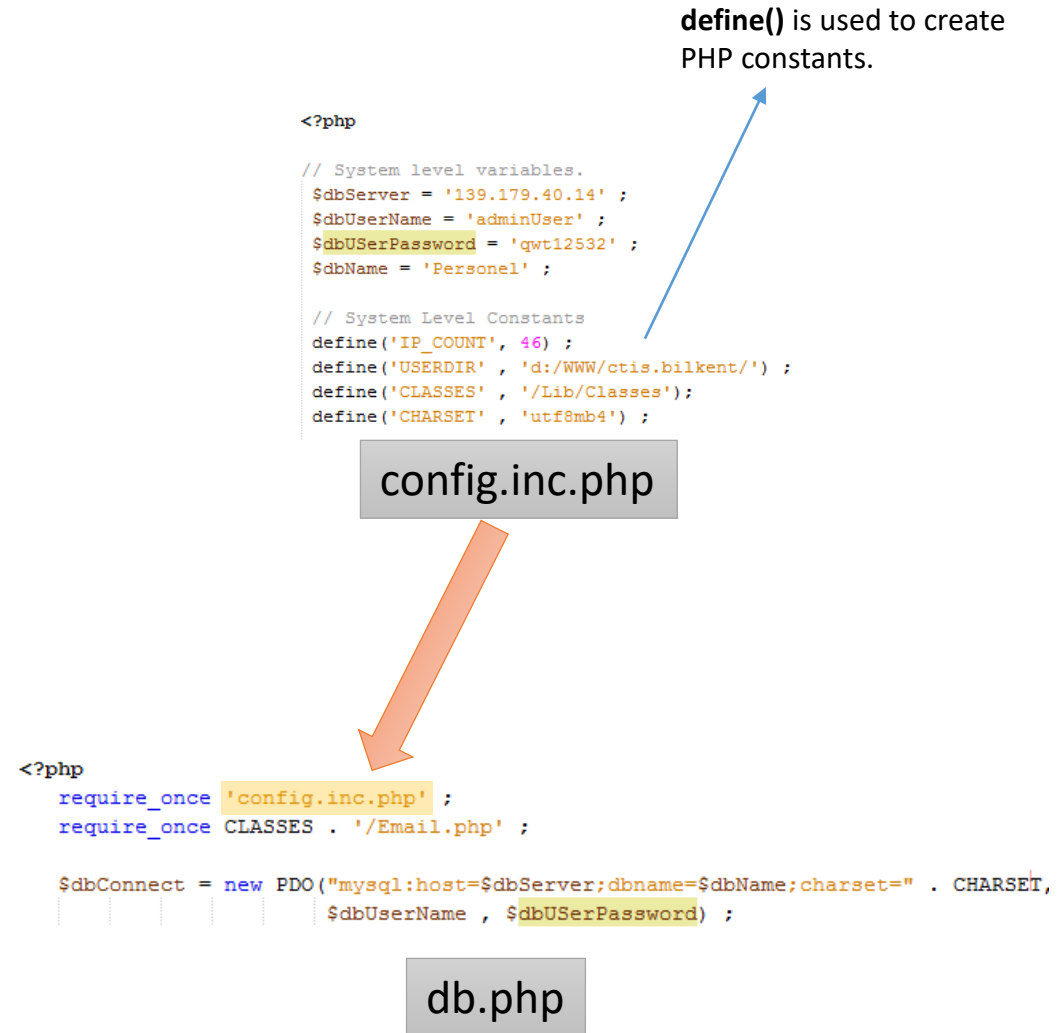
print findClass( "Özge") ; // 7

// STATIC
function addOne() {
    static $count = 0 ;
    $count++;
    return $count ;
}

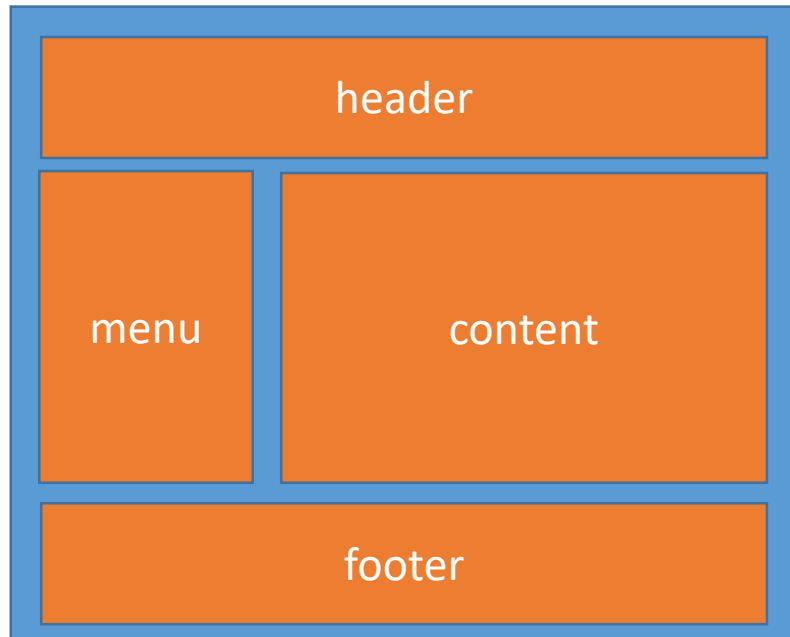
addOne();
addOne();
$result = addOne();
print $result ; // 3
```

Server-Side Includes (SSI)

- To create a modular project, one module (file, class) should be included by other files.
- PHP can embed(copy) the content (html code, php, text) of a file into another file using "include" or "require" command.
- "require" command stops if it can not find the file.
- "include" command produces "warning", and continues script execution.
- This is the method how you can add libraries or classes into your php file.
- Useful for reusing headers, footers, and php functions and classes in multiple pages.
- For codes, "include_once" or "require_once" are used to prevent multiple inclusion.
- For html code snippets, "include" or "require" may be needed to include more than once in the file.



Modular Page Design



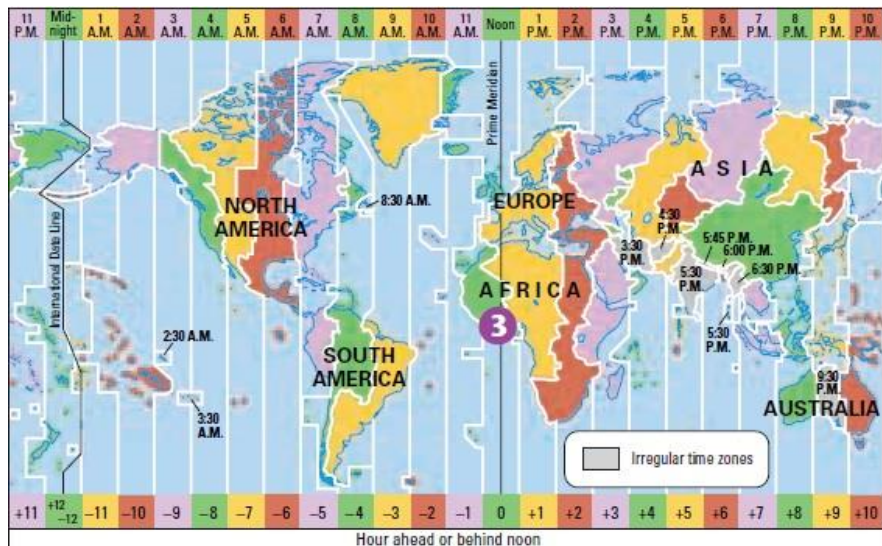
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Main Page</title>
  </head>
  <body>
    <?php include 'header.php' ?>
    <table>
      <tr>
        <td><?php include 'menu.php' ?></td>
        <td><?php include 'index_content.php' ?></td>
      </tr>
    </table>
    <?php include 'footer.php' ?>
  </body>
</html>
```

index.php

Reusability : header.php, menu.php and footer.php can be used in multiple pages

Date and Time

- Date and Time is basically used to log user actions (login, logout, date of an order, comment date), to test expire dates (membership expire date), find remaining days and time, to calculate the age of a user, to track user's actions in time.
- Main classes for date and time operations:
 - DateTime
 - DateTimeZone
 - DateInterval
- Timestamp is number of seconds after 1 Jan 1970 00:00:00 in UTC.
- Take a timezone into account when dealing with timestamp. In Turkey, timestamp is 0, when date is 1 Jan 1970 03:00:00, since Turkey has an offset of 3 hours ahead of UTC.



```
date_default_timezone_set('Europe/Istanbul');
// Create a Date
$turkey = new DateTime(); // current date and time for php's timezone
$la = new DateTime(NULL, new DateTimeZone('America/Los_Angeles'));
$birthday = new DateTime('1990-6-19');
$exactbday = new DateTime('1990-6-19 17:55:03');
$graduate = DateTime::createFromFormat('d-m-y H:i:s', '1-6-90 01:05:03');

// Create a Date from another DateTime object.
$afterGrad = clone $graduate; // how to copy an object
// +/-, years, month, week, day, hours minutes seconds
$afterGrad->modify("+2 weeks -1 day +2 hours +30 seconds");

// Display a Date in a format.
echo 'Graduate Date : ' . $afterGrad->format(DateTime::COOKIE) . '<br>';
echo $afterGrad->format('j-M-Y H:i:sa') . '<br>'; // custom format
echo 'Turkey : ' . $turkey->format(DateTime::COOKIE) . '<br>';
echo 'LA : ' . $la->format(DateTime::COOKIE) . '<br>';

// Comparison with relational operators
if ( $turkey == $la ) {
    print "<p>Both DateTime (Turkey and LA) show the same time</p>" ;
}
$la->modify("+1 second");
if ( $la > $turkey ) {
    print "<p>new LA date is greater than Turkey</p>" ;
}

// Find difference between two dates.
$universityStart = new DateTime("2017-09-24");
$now = new DateTime();
$diff = $universityStart->diff($now); // $diff is DateInterval object.
print "<p>Total Days : ' . $diff->days . ' <br>" ;
// Date starts with P and Time with T
// 1 Year, 2 Months, 2 Days, 5 Hours, 30 Minutes.
$interval = new DateInterval("P1Y2M2DT5H30M");
$graduate->add($interval); // to add interval.
```

Formatting Date and Time

d: day of the month (01-31)
D: day with textual rep. (Sun, Mon)
l: full day name
m: numeric month (01- 12)
M: textual month (Jan, Feb) **F**: full representation
y: two digit year
Y: Four digit year
H: 24-hour format of an hour (00-23)
i: minutes (00-59)
s: seconds (00-59)

Timestamp Functions

`int time()` : returns current timestamp in seconds of the default timezone

`int mktime(hour, min, sec, month, day, year)` :
returns a timestamp for specific date and time for the default timezone.

Misc. Functions

```
bool checkdate( month, day, year)

float microtime( [float format]): current
timestamp in microseconds.

bool date_default_timezone_set('Europe/Istanbul')

string date_default_timezone_get()
```

```
$start = microtime(true) ;
doSomething() ;
$end = microtime(true);
$elapsedTime = $end - $start ;
print '<p>Elapsed Time is ' . $elapsedTime . 'seconds</p>' ;
```

```
function doSomething() {
    $a = 0 ;
    for ( $i=0; $i<10000; $i++) {
        $a++ ;
    }
}
```

Math Functions

```
<?php

// Round fractions DOWN
floor(5.989) ; // 5
floor(5.123) ; // 5
floor(-3.7) ; // -4

// Round fractions UP
ceil(5.89) ; // 6
ceil(5.001) ; // 6
ceil(5.0) ; // 5
ceil(-3.7) ; // -3

// Round a float number
round(3.6) ; // 4
round(3.4) ; // 3
// two digits after decimal point
round( 19.235, 2) ; // 19.24

// Max and Min in an array or values
min( 3, 6, 8, 9, -3) ; // -3
min( array(4,-5, 7,9)) ; // -5
max( 5, 7, 2, -2, 9) ; // 9
```

```
// Integer random number generator.
rand() ; // 4534
rand(3, 10) ; // 6 [3, 10]

// Power of
pow(3, 5) ; // 3*3*3*3*3 = 243
pow( 9, 0.5) ; // sqrt(9) = 3
pow( 16, 0.25) ; // 2

// floating modulus
fmod( 9.2, 3) ; // 0.2
fmod( 4.5, 1.5) ; // 0
fmod( 4.5, 1.1) ; // 0.1 (4 * 1.1 + 0.1)

// absolute value
abs( 4.5) ; // 4.5
abs( -3.56) ; // 3.56

// Base conversion (number, from, to)
base_convert(295, 10, 16) ; // 127 - hex
base_convert("4f3a", 16, 10) ; // 20282
base_convert("81", 16, 2) ; // 1000 0001
base_convert( "81", 10, 2) ; // 0101 0001
```

Check php.net for further math functions.