Zend PHP 5.3
Certification

zend
CERTIFIED
ENGINEER
PHP 5.3

Z zend™ | The PHP Company

Zend Technologies, Inc.
2006 - 2011

Introduction to the Certification

Zend Certified Engineer (ZCE) Credential

Why become a Zend Certified Engineer? Well in today's competitive market, it's more difficult than ever to stand out from the competition. When polled, your colleagues overwhelmingly (98%) endorsed receiving the ZCE - attaining the 'Zend Certified Engineer' credential demonstrates the highest degree of PHP expertise to current or potential employers. As credentialed developers will tell you, this can translate to better jobs with higher pay.

Studying for the PHP 5.3 Certification

This study guide provides a complete outline of the ten major topic areas on which you are tested. At the beginning of each topic section, you are additionally given the list of required subtopic areas for which you are responsible.

This guide cannot encompass all you need to know for the certification... instead, it highlights within each subtopic some of the major concepts. You will still need to explore each subtopic within the PHP Manual. This guide is meant to help you focus on the areas which are considered the most important for certifying you as an expert in PHP v5.3.

About Zend

Zend is the PHP company. Businesses utilizing PHP know Zend as the place to go for PHP expertise and sound technology solutions. Zend delivers premier web application platform products and services for PHP applications. With commercial products and services that enable developers and IT personnel to deliver business-critical PHP applications, Zend is taking the power of PHP to the enterprise.

If you have any questions about the PHP 5.3 Certification, or would like to provide feedback to us on this guide, please contact us at: certification@zend.com.

PHP 5.3 CERTIFICATION: SNAPSHOT

- **THE EXAM**

    - COMPOSED OF ~ 70 RANDOMLY-GENERATED QUESTIONS

    - QUESTIONS VARY IN THEIR LEVEL OF DIFFICULTY

        - THAT IS WHY THE NUMBER OF QUESTIONS PER EXAM VARIES

    - QUESTIONS WILL OFTEN TEST MORE THAN ONE CONCEPT AT A TIME

    - QUESTIONS WILL COVER TEN DIFFERENT TOPIC AREAS

    - ALLOWED 90 MINUTES IN TOTAL TO ANSWER THE QUESTIONS

- **THE TEST TOPICS**

    - THE TEN TOPIC AREAS FROM WHICH THE QUESTIONS ARE DERIVED :

        1. PHP BASICS
        2. DATA FORMATS AND TYPES
        3. STRINGS
        4. ARRAYS
        5. INPUT / OUTPUT
        6. FUNCTIONS
        7. OBJECT-ORIENTED PROGRAMMING
        8. DATABASES
        9. SECURITY
        10. WEB FEATURES

- **THE TEST TOPICS (CONTINUED)**

  - QUESTIONS REFLECT THE CURRICULUM SPECIFIED BY THE ZEND PHP EDUCATION ADVISORY BOARD

  - CERTAIN TOPICS ARE GIVEN MORE WEIGHT IN THE CERTIFICATION:

    - **HIGHEST EMPHASIS:**

      PHP BASICS

      SECURITY

      WEB FEATURES

    - **AVERAGE EMPHASIS:**

      FUNCTIONS

      OOP

      ARRAYS

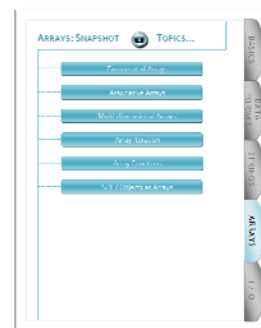   STRINGS & PATTERNS

    - **LOWEST EMPHASIS:**

      DATABASES

      DATA TYPES AND FORMATS

      INPUT / OUTPUT

  - PASSING THE TEST IS BASED ON A BREADTH OF KNOWLEDGE OF THESE TOPICS... BEING AN EXPERT IN 1 OR 2 TOPICS WILL NOT BE ENOUGH

  - WITHIN A TOPIC, THERE ARE CONCENTRATION AREAS - THESE ARE LISTED ON THE FIRST PAGE OF EACH SECTION

- **THE REGISTRATION PROCESS**

  - THE PHP 5.3 CERTIFICATION EXAM IS ADMINISTERED BY PEARSON VUE TRAINING CENTERS (HTTP://WWW.PEARSONVUE.COM)

  - REGISTER FOR THE EXAM EITHER ONLINE, BY PHONE, OR AT A TEST CENTER

    - OPTIONS VARY BY COUNTRY... PLEASE CHECK THE PEARSON WEB SITE

  - YOU NEED TO BRING <u>2</u> FORMS OF IDENTIFICATION - BOTH MUST HAVE YOUR SIGNATURE, ONE MUST HAVE YOUR PICTURE

  - THE TESTING CENTER WILL SUPPLY YOU WITH EITHER "SCRATCH" PAPER OR AN ERASABLE BOARD FOR ANY CALCULATIONS YOU MIGHT NEED TO MAKE...

    YOU ARE <u>NOT</u> ALLOWED TO BRING ANYTHING INTO THE EXAM WITH YOU (NOTES, CALCULATORS, ETC.)

PHP 5.3 CERTIFICATION: 🔍 FOCUS

- **TEST QUESTION TYPES: 3**

  ### MULTIPLE CHOICE... ONLY ONE ANSWER CORRECT

  MOST FREQUENT TYPE OF QUESTION

  ### MULTIPLE CHOICE... TWO OR MORE ANSWERS CORRECT

  QUESTIONS WILL NOTE THAT MORE THAN ONE ANSWER IS REQUIRED

  ### FREE TEXT... OPEN ANSWER

  NO WHITESPACE, EXPLANATIONS, OR COMMENTS ALLOWED HERE

  YOU DO <u>NOT</u> HAVE TO CODE LARGE BLOCKS... THESE ANSWERS ARE SHORT - FOR EXAMPLE, YOU MAY HAVE TO IDENTIFY A FUNCTION OR ITS PARAMETERS, OR ANALYZE CODE

- **GUESS!**

  - THERE IS NO PENALTY FOR GETTING AN ANSWER WRONG... YOU ONLY GET CREDIT FOR CORRECT ANSWERS

- **MARK QUESTIONS FOR REVIEW**

  - YOU CAN EASILY RETURN TO QUESTIONS MARKED FOR REVIEW BEFORE SUBMITTING YOUR ANSWERS

- **EMPHASIS ON ANALYSIS VS. MEMORIZATION**

    - GENERALLY, THE QUESTIONS WILL FOCUS MORE ON ANALYSIS OF CODE RATHER THAN HAVING YOU SUPPLY MEMORIZED ELEMENTS

    - HOWEVER, YOU WILL NEED TO KNOW COMMONLY USED CODE ELEMENTS, SUCH AS COMMON FUNCTIONS AND THEIR PARAMETERS, SOME VARIABLE NAMES, ...

- **ASSUMED ENVIRONMENT**

    - THE QUESTIONS ARE INDEPENDENT OF OPERATING SYSTEM AND SPECIFIC DATABASES/ ADAPTERS

        - HOWEVER, A GENERAL UNDERSTANDING OF RELATED TECHNOLOGIES LIKE HTTP OR SQL IS REQUIRED

            EXAMPLE: YOU SHOULD BE ABLE TO UNDERSTAND THE FOLLOWING QUERY: `SELECT * FROM table WHERE id > 10 ORDER BY name`

    - QUESTIONS REFER TO A VIRTUAL PHP SYSTEM WITH THE RECOMMENDED CONFIGURATION:

        - `REGISTER_GLOBALS` IS OFF

        - `magic_quotes_gpc` IS OFF

        - ERROR REPORTING IS SET TO `E_ALL`

        - ERRORS ARE DISPLAYED (UNLESS OTHERWISE NOTED)

- **TEST RESULTS**

    - YOU ARE IMMEDIATELY INFORMED OF WHETHER YOU HAVE PASSED OR NOT

    - IF YOU DO NOT PASS, YOU WILL BE GIVEN PRINTED FEEDBACK ON EACH TOPIC TO SHOW YOU AREAS REQUIRING ADDITIONAL STUDY... NO DETAILED SCORE IS GIVEN

BASICS: SNAPSHOT TOPICS...

- Syntax
- Operators
- Variables
- Control Structures
- Language Constructs & Functions
- Constants
- Namespaces
- Extensions & AJAX
- Configuration
- Performance

SYNTAX   FOCUS

- **PUNCTUATION**
  - TERMINATE CODE STATEMENTS WITH A SEMI-COLON (;)
  - USE APPROPRIATE TAGS

- **TAGS**

OPENING …	**CLOSING** …
`<?php`	`?>`
`<script language='php'>`	`</script>`

- **COMMENTS**

`//`	USED FOR A SINGLE COMMENT LINE
	MUST BE REPEATED FOR MULTIPLE LINES...
`/* AND */`	USED TO DELINEATE A COMMENT BLOCK
	`/*` USED ONCE AT BEGINNING
	`*/` USED ONCE AT END

OPERATORS 🔍 FOCUS

- **ARITHMETIC OPERATORS**

    - **BASIC CALCULATIONS**

        + (ADDING)

        - (SUBSTRACTING)

        * (MULTIPLYING)

        / (DIVIDING)

    - **MODULUS** (REMAINDER WHEN DIVIDING)

        Ex: `$m = 5 % 2; // $m == 1`

- **BITWISE OPERATORS**

    - USE TO WORK WITH BITS WITHIN AN INTEGER; ARITHMETIC

    - INTEGRAL NUMBERS ARE INTERNALLY CONVERTED INTO BITS

        Ex: `5 -> 0101 = 0*8 + 1*4 + 0*2 + 1*1`

    - **LOGICAL**   **SYMBOL**   **CRITERIA (BY PLACE)**

        AND   `&`   MATCHING "1" IN BOTH OPERANDS

        OR   `|`   AT LEAST ONE "1" IN AN OPERAND

        EITHER-OR   `^`   ONLY ONE "1" IN BOTH OPERANDS

    - SHIFT BITS   `<< x`   MOVE BITS BY X TIMES

        Ex: `4 >> 2 == 1`   [LIKE DIVIDING BY 4]

    - NEGATE BITS   `~`   CONVERT 0S INTO 1S; 1S INTO 0S

- **ASSIGNMENT OPERATORS**

  - **ASSIGN** (=)

    WHEN USING ARRAYS, ASSIGN VALUES TO KEYS WITH =>

  - **SHORT FORMS (COMBINED)**
    - **ASSIGNMENT** (+ AND *OPERATOR*)

    WORKS WITH OPERATORS: –   *   /   & | ^   >>   <<

    EX: `$a += 1;` IS SHORT-HAND FOR `$a = $a + 1;`

  - **COMBINED/CONCATENATING ASSIGNMENT (. =)**

    EX:

    `$a = "Hello,";`

    `$a .= "World !";` ... RESULTS IN "HELLO, WORLD !"

  - **INCREASE / DECREASE (** ++   --   **)**

    PLACEMENT IMPORTANT: IN FRONT OF EXPRESSION - INCREASED OR DECREASED FIRST; AFTER EXPRESSION, THE REVERSE

- **COMPARISON OPERATORS**

  - **EQUALITY** (==)   **INEQUALITY** (!=)

    PHP HANDLES DATA TYPE CONVERSION `"123" == 123`

      (===)   (!==)

      PHP CHECKS THE DATA TYPE `"123"  !== 123`

  - **GREATER THAN** (>)   **LESS THAN** (<)

    GREATER OR EQUAL (>=)   **LESS THAN OR EQUAL** (<=)

- **STRING OPERATORS**

  CONCATENATE (.) AND CONCATENATING ASSIGNMENT (.=) *SEE ABOVE*

- **ARRAY OPERATORS**

+	UNION
==	EQUAL
===	IDENTICAL
!=	NOT EQUAL
<>	NOT EQUAL
!==	NOT IDENTICAL

- **LOGICAL OPERATORS**

EXAMPLE	OPERATOR	EVALUATES AS TRUE WHEN...
`$a and $b`	`and`	BOTH `$a` AND `$b` TRUE
`$a or $b`	`or`	EITHER `$a` OR `$b` TRUE
`$a xor $b`	`xor`	EITHER `$a`, `$b` TRUE, NOT BOTH
`! $a`	`not`	`$a` NOT TRUE
`$a && $b`	`and`	BOTH `$a` AND `$b` TRUE
`$a || $b`	`or`	EITHER `$a` OR `$b` true

- **EXECUTION OPERATORS**
  - USE BACKTICKS `` `` `` TO EXECUTE THE CONTENTS ENCLOSED BY THEM AS A SHELL COMMAND, EQUIVALENT TO `shell_exec()`

- **OPERATOR PRECEDENCE**
  - FOLLOWS MATHEMATICAL PRECEDENCE IN MOST INSTANCES (EX: MULTIPLICATION/DIVISION PRECEDES ADDITION/SUBTRACTIONS)
  - USE PARENTHESES TO ENFORCE NON-STANDARD PRECEDENCE

VARIABLES 🔍 FOCUS

- **NAMING**

  - START WITH A "`$`"

  - CONTAINS LETTERS, NUMBERS, AND UNDERSCORES

  - BY CONVENTION, START WITH LOWER CASE

  - CASE-SENSITIVE

- **REFERENCING**

  - VARIABLES CAN BE ASSIGNED BY VALUE OR BY REFERENCE

  - ATTACH AN "`&`" TO THE BEGINNING OF THE VARIABLE BEING ASSIGNED

- **INITIALIZING**

  - VARIABLES HAVE THEIR TYPE SET BY DEFAULT, IF NOT INITIALIZED (WHICH IS OPTIONAL)

  - NOT INITIALIZING VARIABLES CAN POTENTIALLY LEAD TO REPETITIVE VARIABLE NAMES WHEN WORKING WITH MULTIPLE FILES

  - `isset()` IS USED TO DETERMINE WHETHER A VARIABLE HAS BEEN INITIALIZED

CONTROL STRUCTURES 🔍 FOCUS

- **CONDITIONS**

  - **IF**

    EVALUATES FOR A CONDITION (BOOLEAN VALUE), TO DETERMINE WHETHER TO EXECUTE CODE; CAN BE NESTED

  - **ELSE**

    PROVIDES ALTERNATIVE EXECUTION, WHEN COMBINED WITH IF (=`FALSE`)

  - **ELSEIF (ELSE IF)**

    PROVIDES ALTERNATIVE EXECUTION, WHEN COMBINED WITH IF (=`FALSE`), BUT ITS OWN CONDITION MUST BE MET (FLOW: IF... ELSEIF ... ELSE)

  - **IF-ELSE (TERNARY OPERATOR)**

    SPECIAL FORM: (EXPRESSION) ? VALUEIFTRUE : VALUEIFFALSE

  - **SWITCH**

    USE TO EVALUATE (BOOLEAN VALUE) AGAINST A <u>SERIES</u> OF CONDITIONS, TO DETERMINE WHICH CODE TO EXECUTE FOR EACH CONDITION

- **LOOPS**

  - **WHILE**

    EXECUTES STATEMENT UNTIL CONDITION IS NO LONGER EVALUATED AS BOOLEAN TRUE; CONDITION EVALUATED AT BEGINNING

  - **DO-WHILE**

    EXECUTES STATEMENT UNTIL CONDITION IS NO LONGER EVALUATED AS BOOLEAN TRUE; CONDITION EVALUATED AT END

- **LOOPS (CONTINUED)**

  - **FOR**

    EXECUTES FIRST STATEMENT ONE TIME AS AN ASSIGNMENT, THE SECOND STATEMENT AS A LOOPING CONDITION CHECKED AT THE BEGINNING OF THE FIRST AND SUBSEQUENT ITERATIONS UNTIL CONDITION IS NO LONGER EVALUATED AS BOOLEAN TRUE, THEN EXECUTES THE THIRD AND FINAL STATEMENT AT THE END OF EACH ITERATION

  - **FOREACH**

    USED ONLY FOR ARRAYS; ASSIGNS VALUE OF CURRENT ELEMENT TO THE VARIABLE AND ADVANCES THE ARRAY POINTER UNTIL IT REACHES THE LAST ELEMENT

  - **CONTINUE**

    WITHIN LOOPS, USED TO PASS OVER ANY REMAINING CODE WITHIN THE ITERATION AND RETURN TO THE INITIAL CONDITION EVALUATION STEP

  - **BREAK**

    HALTS EXECUTION OF LOOPS UTILIZING THE `FOR`, `FOREACH`, `WHILE`, `DO-WHILE`, `SWITCH` CONTROL STRUCTURES

LANGUAGE CONSTRUCTS 🔍 FOCUS

- **OUTPUT CONSTRUCTS**

  - **DIE() AND EXIT()**
    THESE CONSTRUCTS ARE EQUIVALENT
    USED TO OUTPUT A RESULT AND THEN TERMINATE THE RUNNING SCRIPT

  - **ECHO()**
    USED TO OUTPUT A RESULT (TEXT, STRINGS, VARIABLES)

    IF USING <u>STRINGS</u> CONTAINING QUOTATIONS, MAKE SURE YOU HANDLE THEM
    CORRECTLY (USE APOSTROPHE OR ESCAPE WITH \)

  - **RETURN()**
    USED TO HALT EXECUTION OF A FUNCTION (CALLED WITHIN FUNCTION) OR OF A
    SCRIPT (CALLED WITHIN GLOBAL SCOPE)

  - **PRINT()**
    USED TO OUTPUT A STRING

- **EVALUATION CONSTRUCTS**

  - **EMPTY()**
    USED TO ASSESS WHETHER A VARIABLE (ONLY) IS EMPTY

  - **EVAL()**
    USED TO EVALUATE THE CONTENTS OF A STRING AS PHP CODE

  - **INCLUDE() AND INCLUDE_ONCE()**
    USED TO BOTH <u>INCLUDE</u> AND <u>EVALUATE</u> A FILE;

- **REQUIRE() AND REQUIRE_ONCE()**
  THESE CONSTRUCTS ARE SIMILAR TO `include()` AND `include_once()`, EXCEPT THAT A FAILURE IN EXECUTION RESULTS IN A <u>FATAL ERROR</u>, WHILE `include()` GENERATES A WARNING

- **OTHER CONSTRUCTS**

  - **ISSET() AND UNSET()**
    `isset()`: USE TO DETERMINE WHETHER A VARIABLE HAS BEEN SET (THEREFORE, IS NOT NULL)

    `unset()`: USE TO UNSET THE VARIABLE

  - **LIST()**
    USE TO ASSIGN A GROUP OF VARIABLES IN ONE STEP

CONSTANTS 🔍 FOCUS

- **DEFINITION:**

    - IDENTIFIER FOR A VALUE THAT DOES NOT CHANGE ONCE DEFINED

- **NAMING:**

    - START WITH A LETTER OR UNDERSCORE, ARE CASE SENSITIVE, CONTAIN ONLY ALPHANUMERIC CHARACTERS AND UNDERSCORES

    - BY CONVENTION USE ONLY UPPERCASE LETTERS

- **ACCESS:**

    - MAY BE DEFINED AND ACCESSED ANYWHERE IN A PROGRAM

    - MUST BE DEFINED BEFORE USE; CANNOT BE CHANGED SUBSEQUENTLY

"MAGIC" CONSTANTS (__XXX__)

- **DEFINITION:**

    - PHP PROVIDES A SET OF PREDEFINED CONSTANTS DEFINED BY THE PHP CORE (EX: `E_ERROR; TRUE`)

    - SEVERAL OF THESE CAN CHANGE DEPENDING UPON WHERE USED, AND THEREFORE NOT TRUE CONSTANTS (EX: `__DIR__ ; __NAMESPACE__`)

NAMESPACES   FOCUS

- **DEFINITION:**

    - NAMESPACES ARE A METHOD OF GROUPING RELATED PHP CODE ELEMENTS WITHIN A LIBRARY OR APPLICATION

- **USE:**

    - HELPS TO PREVENT ACCIDENTALLY RE-DEFINING FUNCTIONS, CLASSES, CONSTANTS, ETC.

    - AVOIDS HAVING TO USE LONG, HIGHLY DESCRIPTIVE CLASS NAMES

    - CONSTANTS, CLASSES, AND FUNCTIONS ARE AFFECED BY THE USE OF NAMESPACES

    - CREATE SUB-NAMESPACES TO SUB-DIVIDE A LIBRARY

- **DECLARING NAMESPACES**

    - MUST DECLARE THE USE OF NAMESPACES WITH THE KEYWORD "`namespace`" AT THE BEGINNING OF THE CODE FILE (RIGHT AFTER `<?PHP`)

    - USE ONE NAMESPACE PER CODE FILE (BEST PRACTICE)

    - UNLESS A NAMESPACE IS DEFINED, CLASSES AND FUNCTIONS ARE CONTAINED WITHIN THE GLOBAL SPACE

        - PREPEND "`\`" TO INDICATE USE OF AN ELEMENT FROM THE GLOBAL SPACE

    - ONCE CODE ELEMENTS WITHIN A SINGLE NAMESPACE ARE DEFINED, THEY CAN BE USED IN OTHER PHP FILES

- **IMPORTING / ALIASING NAMESPACES**

  - ONCE DECLARED, IMPORT NAMESPACES WITH THE "`use`" OPERATOR

  - CAN CREATE ALIASES FOR NAMESPACES

    - EX:

      IF COMPLETE NAMESPACE NAME IS PATH1/PATH2/PATH3, SET =E

      THEN, WHEN NEED TO CALL, CAN REFERENCE ALIAS AS E/PATH4

- **NOTE:**

  - NAMESPACES ARE NOT EQUIVALENT TO CLASSES... A CLASS IS AN ABSTRACT DEFINITION OF AN OBJECT, WHILE A NAMESPACE IS AN ENVIRONMENT IN WHICH A CLASS, FUNCTION, OR CONSTANT CAN BE DEFINED

EXTENSIONS 🔍 FOCUS

- **THERE ARE MANY ADD-ONS (EXTENSIONS), AVAILABLE FOR SPECIFIC PROGRAMMING TASKS**

    - ADDED TO THE `php.ini` CONFIGURATION FILE

    - NEED TO CONFIGURE `php.ini` TO ACTIVATE THE EXTENSIONS YOU WANT TO USE, AS WELL AS SPECIFY ALL THE NEEDED PATHS (EX: LIBRARIES)

    - MANY RULES AROUND THE USE OF THE EXTENSIONS IS PROVIDED WITHIN THE RELATED CHAPERS (EX: SIMPLEXML IN THE DATA TYPES & FORMATS CHAPTER)

    - NOT ALL EXTENSIONS CAN BE DISCUSSED WITHIN THIS GUIDE... PLEASE REVIEW THE COMPLETE LISTING AVAILABLE IN THE PHP MANUAL (REFERENCE CITED BELOW)

- **PECL (PHP EXTENSION COMMUNITY LIBRARY)**

    - REPOSITORY FOR PHP EXTENSIONS; SIMILAR STRUCTURE AND CONCEPT TO THE PHP CODE REPOSITORY PEAR (PHP EXTENSION AND APPLICATION REPOSITORY)

- **CORE EXTENSIONS**

    - THERE ARE A SET OF VARIOUS PHP LANGUAGE ELEMENTS, CALLED CORE EXTENSIONS, THAT ARE PART OF THE PHP CORE

    - THEY INCLUDE SPECIFIC ARRAYS, CLASSES, OBJECTS, ETC.

- **USERLAND RULES**

    - USERLAND REFERS TO THOSE APPLICATIONS THAT RUN IN THE USER SPACE (VS. THE KERNAL)

- **USERLAND RULES (CONTINUED)**
  - SELECT RULES: (SEE THE COMPLETE LISTING IN THE PHP MANUAL)

    ### GLOBAL NAMESPACE CONSTRUCTS:
    - FUNCTIONS
    - CLASSES
    - INTERFACES
    - CONSTANTS (OTHER THAN CLASS)
    - VARIABLES (DEFINED OUTSIDE OF FUNCTIONS OR METHODS)

    ### INTERNAL NAMING:
    - FUNCTIONS USE UNDERSCORES BETWEEN WORDS
    - CLASSES USE THE CAMELCASE RULE
    - THE DOUBLE UNDERSCORE PREFIX IS RESERVED, AND REFERS TO ELEMENTS CONSIDERED "MAGICAL"

CONFIGURATION 🔍 FOCUS

- **DEFINITION:**

    - CONFIGURATION FILES ESTABLISH THE INITIAL SETTINGS FOR APPLICATIONS, AS WELL AS SERVERS AND OPERATING SYSTEMS

- **PHP.INI:**

    - CONFIGURATION FILE FOR PHP

    - FILE RUN UPON SERVER STARTING OR UPON INVOCATION (`CGI` / `CLI`)

    - SEARCH ORDER:

        `sapi` MODULE > `phprc` VARIABLE > `Registry` KEYS > `HKEY_LOCAL_MACHINE\software\php` > `Working` DIRECTORY (NOT `CLI`) > `Directory` (SERVER OR PHP) > `WIN` DIRECTORY

- **.USER.INI:**

    - PHP SUPPORTS `.htaccess`-TYPE INI FILES(V5.3)
        - PROCESSED BY CGI/FASTCGI SAPI
        - MUST USE `PHP_INI_PERDIR` OR `PHP_INI_USER`

    - PHP SEARCHES FOR THESE INI FILES IN ALL DIRECTORIES

    - CONTROLLED BY DIRECTIVES `user_ini.filename`, `user.cache_ttl`
        - FILE NAMED BY `user_ini.filename` (`default = user.ini`)
        - FILE READING FREQUENCY DEFINED BY `user.cache_ttl`

- **SETTINGS**

    - CAN DEFINE VERSION/S OF `PHP` IN `INI` FILE

    - GENERALLY, USE `ini_set`() WITHIN THE PHP SCRIPT; SOME SETTINGS REQUIRE `php.ini` OR `httpd.conf`

    - APACHE: CHANGE CONFIG SETTINGS USING DIRECTIVES IN APACHE CONFIG FILES AND `.htaccess`; REQUIRES `AllowOverride` [`Options/All`] PRIVILEGES

PERFORMANCE 🔍 FOCUS

- **FACTORS AFFECTING PERFORMANCE (TWO MAJOR AREAS)**

  - REDUCED MEMORY USAGE

  - RUN-TIME DELAYS

- **GARBAGE COLLECTION**

  - CLEARS CIRCULAR-REFERENCE VARIABLES ONCE PREREQUISITES ARE MET, VIA ROOT-BUFFER FULL OR CALL TO THE FUNCTION `gc_collect_cycles()`

  - GARBAGE COLLECTION EXECUTION HINDERS PERFORMANCE

REFERENCES:

http://php.net/manual/en/language.basic-syntax.php

http://us.php.net/manual/en/language.operators.php

http://www.php.net/manual/en/language.variables.basics.php

http://www.php.net/manual/en/language.control-structures.php

http://www.php.net/manual/en/language.constants.php

http://www.php.net/manual/en/language.namespaces.rationale.php

http://www.php.net/manual/en/configuration.file.php

http://php.net/manual/en/features.gc.performance-considerations.php

TEST YOUR KNOWLEDGE : QUESTIONS

1

What is the output of the following code?

```
$a = 1;
    ++$a;
    $a *= $a;
    echo $a--;
```

A: 4

B: 3

C: 5

D: 0

E: 1

2

When PHP is running on a command line, what super-global will contain the command line arguments specified?

A: $_SERVER

B: $_ENV

C: $GLOBALS

D: $_POST

E: $ ARGV

3

Function world() is defined in the namespace 'myapp\utils\hello'. Your code is in namespace 'myapp'.

What is the correct way to import the hello namespace so you can use the world() function?

A: use hello

B: use utils\hello

C: use myapp\utils\hello

D: use myapp\utils\hello\world;

4 **What is the output of the following script?**

```php
<?php
function fibonacci ($x1, $x2)
{
        return $x1 + $x2;
}

$x1 = 0;
$x2 = 1;

for ($i = 0; $i < 10; $i++) {
    echo fibonacci($x1, $x2) . ',';
}
?>
```

A: 1,2,3,4,5,6,7,8,9

B: 1,2,3,4,5,6,7,8,9,10,

C: 1,2,3,5,8,13,21,34,55,89,

D: 1,1,1,1,1,1,1,1,1,1,

5 **Which PHP functions may be used to find out which PHP extensions are available in the system? (Choose 2)**

A: extension_loaded()

B: get_extension_funcs()

C: get_loaded_extensions()

D: phpinfo()

6 **What is the name of the error level constant that is used to designate PHP code that will not work in future versions?**

????

7

Your PHP script is repeatedly parsing 50KB of data returned from a remote web service into browser-readable HTML.

Users complain that the script takes a long time to run. Which of the following measures usually leads to the best results? (Choose 2)

A: **Install a bytecode cache**

B: **Install an SSD drive on the server**

C: **Cache the data returned by the web service locally**

D: **Upgrade to the latest version of PHP**

8

What will the following code produce?

```php
<?php

define('CONSTANT',1);
define('_CONSTANT',0);

define('EMPTY','');

if (!empty(EMPTY)) {
    if (!((boolean) _CONSTANT)) {
        print "One";
    }
}
else if(constant('CONSTANT') == 1){
    print "Two";

}
```

A: **One**

B: **Two**

C: **Syntax Error**

TEST YOUR KNOWLEDGE : ANSWERS

1 A: 4

2 A: $_SERVER

3 C: use myapp\utils\hello

4 D: 1,1,1,1,1,1,1,1,1,1,

5 C: get_loaded_extensions()
D: phpinfo()

6 E_DEPRECATED

7 C: Cache the data returned by the web service locally
D: Upgrade to the latest version of PHP

8 C: Syntax Error

DATA TYPES: SNAPSHOT TOPICS...

XML Basics

XML Extension

SimpleXML

Xpath

Web Services Basics

SOAP

REST

JSON & AJAX

Date & Time

DOM

DATA TYPES & FORMATS 🔍 FOCUS

XML BASICS

- **DEFINITION**
    - XML IS ACRONYM FOR E<u>X</u>TENSIBLE <u>M</u>ARKUP <u>L</u>ANGUAGE
    - DATA FORMAT ("UNIVERSAL") USED FOR STRUCTURED DOCUMENT EXCHANGE

XML EXTENSION

- **EXTENSION ALLOWS FOR PARSING OF XML DOCUMENTS**
    - CREATE XML PARSERS (+ PARAMS) AND DEFINE CORRESPONDING HANDLERS

        `xml_parser_create()`   ... AND ...

        `xml_parser_create_ns()`   FOR PARSER WITH NAMESPACE SUPPORT

        `xml_set_element_handler()`   SEE OTHER FUNCTIONS IN PHP MANUAL

- **CHARACTER ENCODINGS**
  - **SOURCE ENCODING :**
    - CONDUCTED AT TIME OF PARSING
    - CANNOT BE CHANGED DURING PARSER LIFETIME
    - TYPES:
      - UTF-8 (PHP USES THIS TYPE FOR INTERNAL DOCUMENT REPRESENTATION; BYTES UP TO 21)
      - US-ASCII (SINGLE BYTE)
      - ISO-8859-1 (SINGLE BYTE; DEFAULT)

  - **TARGET ENCODING :**
    - CONDUCTED AT TIME OF PHP PASSING DATA TO XML HANDLERS
    - TARGET ENCODING INITIALLY SET TO SAME AS SOURCE ENCODING
    - CAN BE CHANGED AT ANY TIME
  - CHARACTERS NOT CAPABLE OF SOURCE ENCODING CAUSE AN ERROR
  - CHARACTERS NOT CAPABLE OF TARGET ENCODING ARE DEMOTED (TO "?")

- **REQUIRES THE LIBXML EXTENSION (ENABLED BY DEFAULT IN PHP)**
  - FUNCTIONS PART OF EXPAT LIBRARY ALSO ENABLED BY DEFAULT

- **SET OF PREDEFINED ERROR CODE CONSTANTS AVAILABLE**

  - AVAILABLE WHEN DYNAMICALLY LOADED AT RUNTIME OR WHEN COMPILED INTO PHP

  - PARTIAL LIST

    ```
    XML_ERROR_**
    ```

    ```
                     _SYNTAX

                     _INVALID TOKEN

                     _UNKNOWN_ENCODING
    ```

    ```
    XML_OPTION_**
    ```

    ```
                     _OPTION_CASE FOLDING

                     _SKIP_WHITE
    ```

SimpleXML

- **DEFINITION**
  - "SIMPLE" ACCESS TO XML DATA FROM PHP

- **CONCEPT: OOP ACCESS FOR XML DATA**
  - ELEMENTS BECOME OBJECT PROPERTIES
  - ATTRIBUTES CAN BE ACCESSED VIA ASSOCIATIVE ARRAYS

- **FUNCTIONS:**

```
$xml = simplexml_load_string('<?xml...');
$xml = simplexml_load_file('file.xml');
$xml = new SimpleXMLElement('<?xml...');
```

- **CLASS: (EXAMPLES)**

  CREATES A SIMPLEXMLELEMENT OBJECT

```
SimpleXMLElement::construct()
```

  IDENTIFIES AN ELEMENT'S ATTRIBUTES

```
SimpleXMLElement::attributes()
```

  RETRIEVES AN ELEMENT'S NAME

```
SimpleXMLElement::getName()
```

  FINDS CHILDREN OF GIVEN NODE

```
SimpleXMLElement::children()
```

  COUNTS THE NUMBER OF CHILDREN OF AN ELEMENT

```
SimpleXMLElement::count()
```

  RETURNS A WELL-FORMED XML STRING BASED ON A SIMPLEXML ELEMENT

```
SimpleXMLElement::asXML()
```

DOM

- **DEFINITION**

    - DOM EXTENSION PERMITS MANIPULATING OF XML DOCUMENTS WITH ITS API AND PHP 5+

- **REQUIRES THE LIBXML EXTENSION (ENABLED BY DEFAULT IN PHP)**

    - FUNCTIONS PART OF EXPAT LIBRARY ALSO ENABLED BY DEFAULT

- **ENCODING:**

    - USES UTF-8 ENCODING

- **SIMPLEXML AND DOM**

    - `simplexml_import_dom()` CONVERTS A DOM NODE INTO A SIMPLEXML OBJECT

    - `dom_import_simplexml()` CONVERTS A SIMPLEXML OBJECT INTO A DOM (DOCUMENT OBJECT MODEL)

- **SET OF PREDEFINED CONSTANTS AVAILABLE**

    - AVAILABLE WHEN EXTENSION DYNAMICALLY LOADED AT RUNTIME OR WHEN COMPILED INTO PHP

    - PARTIAL LIST (*SEE PHP MANUAL FOR FULL LIST*)

        `XML_ELEMENT_NODE`   DEFINES NODE AS A DOMELEMENT

        `XML_TEXT_NODE`   DEFINES NODE AS A DOMTEXT

XPATH

- **DEFINITION**

    - QUERY LANGUAGE USED TO SELECT NODES WITHIN AN XML DOCUMENT

    - ALSO CAN BE USED TO COMPUTE VALUES USING DATA WITHIN AN XML DOCUMENT

    - KEYWORD: `xpath()`

        - EXECUTES THE QUERY

        - EX: `$xml->xpath('//');`

- **SIMPLEXML AND XPATH**

    - RUNS AN XPATH QUERY ON XML DATA

    - FORMAT: `array SimpleXMLElement::xpath(string $path)`

SOAP

- **DEFINITION**

    - DERIVED ACRONYM FOR SIMPLE OBJECT ACCESS PROTOCOL

    - VERSIONS 1.0 AND 1.1 RELEASED BY THE INDUSTRY; POPULARITY LED TO CONTROL BY W3C WITH VERSION 1.2

    - EXTENSION USED TO WRITE SOAP SERVERS AND CLIENTS

- **REQUIRES THE LIBXML EXTENSION (ENABLED BY DEFAULT IN PHP)**

- **RUNTIME CONFIGURATION**

    - SOAP CACHE FUNCTIONS ARE AFFECTED BY `php.ini` SETTINGS (`soap.wsdl_cache_*`)

- **SET OF PREDEFINED CONSTANTS AVAILABLE**

    - AVAILABLE WHEN DYNAMICALLY LOADED AT RUNTIME OR WHEN COMPILED INTO PHP

    - PARTIAL LIST (ALL INTEGERS)

`SOAP_1_1`	1
`SOAP_1_2`	2
`SOAP_ENCODED`	1
`SOAP_LITERAL`	2
`SOAP_AUTHENTICATION_`	0/1
`SOAP_ENC_*`	300/301
`SOAP_CACHE_*`	0/1/2/3
`SOAP_PERSISTENCE_*`	1/2
`SOAP_RPC`	1

- **SOAP FUNCTIONS**

  `is_soap_fault`   CHECKS IF A SOAP CALL HAS FAILED

  `use_soap_error_handler` INDICATES WHETHER TO USE AN ERROR
                            HANDLER

REST

- **DEFINITION**
    - REST IS ACRONYM FOR REPRESENTATIONAL STATE TRANSFER
    - DESIGN STANDARD (NOT AN EXTENSION); SET OF 4 ARCHITECTURAL PRINCIPLES FOR DESIGNING WEB PAGES AND SERVICES
        - USES ONLY HTTP
        - STATELESS
        - EXPOSES URIS
        - TRANSFERS XML, JSON, OR BOTH
    - CONCEPTUALLY A NETWORK OF PAGES, ACCESSED BY USERS THROUGH A SET OF LINKS

- **DATA TYPES SUPPORTED INCLUDE:**
    - ASCII STRINGS
    - INTEGERS
    - BOOLEANS

- **REST USES HTTP "VERBS":**

    | GET | LIST (WITHOUT IDENTIFIER) |
    | GET | RESOURCE (WITH IDENTIFIER) |
    | POST | CREATE |
    | PUT | UPDATE (WITH IDENTIFIER) |
    | DELETE | DELETE (WITH IDENTIFIER) |

- **REST AND REQUEST HEADERS**

    - TWO CONCEPTS:

        CONTENT-TYPE: WHAT IS BEING PROVIDING

        ACCEPT: WHAT IS EXPECTED IN RESPONSE

    - STATUS CODES:

        `201` = CREATED

        `400` = BAD REQUEST / FAILED VALIDATION

        `401` = UNAUTHORIZED

        `204` = NO CONTENT (*USEFUL WITH* DELETE)

        `500` = APPLICATION ERROR

    - `ext/curl` IS A COMMON WAY OF SENDING MORE COMPLEX HEADER REQUESTS FROM A PHP SCRIPT

- **REST AND RESPONSE HEADERS**

    - TWO CONCEPTS:

        `Content-Type`: WHAT IS BEING RETURNED

        `Accept`: WHAT AND WHEN TO CACHE

- **CONTEXT SWITCHING**

    - REFERS TO THE ACT OF PROVIDING DIFFERENT OUTPUT BASED ON CRITERIA FROM THE REQUEST

    - THE PROCESS INSPECTS THE HTTP REQUEST HEADERS AND/OR THE REQUEST URI, AND VARIES THE RESPONSE APPROPRIATELY

- **CONTEXT SWITCHING (CONTINUED)**

  - COMMONLY USED FOR:

    - PROVIDING DIFFERENT OUTPUT FOR REQUESTS ORIGINATED VIA `XMLHttpRequest`

    - PROVIDING DIFFERENT OUTPUT BASED ON ACCEPT HTTP HEADERS (EX: REST ENDPOINTS)

      PROVIDING ALTERNATE LAYOUTS/CONTENT BASED ON BROWSER DETECTION

JSON & AJAX

- **DEFINITION**

    - JSON IS AN ACRONYM FOR JAVASCRIPT OBJECT NOTATION

    - DATA-INTERCHANGE FORMAT

    - EXTENSION LOADED IN PHP BY DEFAULT

- **SET OF PREDEFINED CONSTANTS AVAILABLE**

    - AVAILABLE WHEN DYNAMICALLY LOADED AT RUNTIME OR WHEN COMPILED INTO PHP

    - PARTIAL LIST (ALL INTEGER)

`JSON_ERROR_NONE`	CONFIRMS WHETHER ERROR OCCURRED OR NOT
`JSON_ERROR_SYNTAX`	INDICATES SYNTAX ERROR
`JSON_ERROR_UTF8`	AIDS IN DETECTING ENCODING ISSUES
`JSON_FORCE_OBJECT`	AIDS IN ENSURING THE RECEIVING END GETS AN OBJECT WHEN AN EMPTY PHP ARRAY IS PASSED

- **FUNCTIONS**

    DECODES A JSON STRING
    ```
    json_decode($json, $assoc = false, $depth)
    ```

    RETURNS THE JSON REPRESENTATION OF A VALUE
    ```
    json_encode($value, $options)
    ```

    RETURNS THE LAST ERROR OCCURRED
    ```
    json_last_error
    ```

    WHERE `$assoc:` INDICATES WHETHER OBJECTS SHOULD BE CONVERTED INTO ASSOCIATIVE ARRAYS (BOOLEAN)

    `$value:` CAN BE OF ANY TYPE EXCEPT A RESOURCE

    `$options:` PARAM ADDED WITH V5.3

DATE & TIME

- ## DEFINITION

  - FUNCTIONS THAT RETRIEVE THE DATE AND TIME FROM THE PHP SERVER

  - FLEXIBLE DATE AND TIME FORMATTING DUE TO FACT THEY ARE STORED AS A 64-BIT NUMBER

  - FUNCTION VALUES REFLECT LOCALE SET ON SERVER, AS WELL AS SPECIAL DATE ADJUSTMENTS LIKE DAYLIGHT SAVINGS TIME, LEAP YEAR

- ## RUNTIME CONFIGURATION

  - `date.*` FUNCTIONS ARE AFFECTED BY PHP.INI SETTINGS
    `date.default_latitude; date.timezone`

- ## SET OF PREDEFINED CONSTANTS AVAILABLE

  - `DateTime` CONSTANTS PROVIDE STANDARD DATE FORMATS, IN CONJUNCTION WITH A DATE FUNCTION LIKE `date()`

- ## DATETIME CLASS

  - CONSTANTS: FORMAT (EXAMPLES)
    ```
    const string DateTime::*
       ::COOKIE = l, d-M-y H:i:s T ; MONDAY, 15-AUG-05 15:52:01 UTC
       ::RSS = D, d M Y H:i:s O ;    MON, 15 AUG 2005 15:52:01 +0000
    ```

  - METHODS: FORMAT (EXAMPLES)
    ```
    public __construct([[string $time = "now" [,
    DateTimeZone $timezone = NULL ]])

    public DateTime add(DateInterval $interval)

    public DateTime setDate(int $year, int $month, int
    $day)
    ```

- STATIC METHODS: FORMAT (OOP-STYLE EXAMPLES)

ADD A SPECIFIED AMOUNT OF TIME TO A DATETIME OBJECT

```
public DateTime DateTime::add(DateInterval $interval)
```

RETURN A NEW DATETIME OBJECT (INSTANTIATION)

```
public  DateTime::__construct()([string $time = "now"
[, DateTimeZone $timezone = NULL ]])
```

RETURN A DATETIME OBJECT IN A SPECIFIC FORMAT

```
public static  DateTime DateTime::createFromFormat(
string $format, string $time [, DateTimeZone
                              $timezone])
```

RETURN A DATE FORMATTED ACCORDING TO A GIVEN FORMAT

```
public string DateTime::format(string $format)
```

RETURN THE DIFFERENCE BETWEEN TWO DATETIME OBJECTS

```
public DateInterval DateTime::diff( DateTime
$datetime2 , bool $absolute = false])
```

RETURN THE UNIX TIMESTAMP

```
public int DateTime::getTimestamp(void)
```

ALTER THE CURRENT TIMESTAMP

```
public DateTime DateTime::modify(string $modify)
```

IN EACH CASE, THE METHOD RETURNS THE OBJECT ON SUCCESS, FALSE ON FAILURE

`$timezone`: WHEN SET TO NULL, RETURNS THE CURRENT TIME

`$format`: THE PARAMETER MUST BE IN A FORMAT ACCEPTED BY DATE()

`$modify`: DATE / TIME STRING IN VALID FORMATS (ADD/SUBTRACT)

REFERENCES:

http://us2.php.net/manual/en/book.xml.php

http://us2.php.net/manual/en/book.dom.php

http://php.net/manual/en/refs.webservice.php

http://us.php.net/manual/en/book.json.php

http://php.net/datetime

http://php.net/simplexml

http://php.net/domxpath

http://php.net/soap

TEST YOUR KNOWLEDGE : QUESTIONS

1

What is wrong with this XML document?

```xml
<?xml version="1.0" encoding="UTF-8"?>
<node>
    <?var type="string" ?>
    <leaf>Value</leaf>
</node>
<?xml version="1.0" encoding="UTF-8"?>
<node>
    <?var type="string" ?>
    <leaf>Value</leaf>
</node>
```

A: The encoding is only required for non-western languages

B: <?var is not a valid node type

C: <?var is missing a closing tag

D: Nothing

2

Which of the following is a feature that does NOT exist in the DateTime extension?

A: The ability to list timezones by continent

B: The ability to modify date data

C: The ability to generate dates between two date periods

D: The ability to parse dates in the cookie format

E: None of the above

3

What is the output of the following code?

```php
1  <?php
2
3  $doc = new DOMDocument();
4  $doc->loadXML('<root />');
5  $el = $doc->createElement('test', 'some value');
6
7  echo $doc->saveXML();
```

A: <?xml version="1.0"?>

B: <?xml version="1.0"?>
    <test>some value</test>

C: <?xml version="1.0"?>
    <root><test>some value</test>
    </root>

D: <?xml version="1.0"?>
    <root test="some value"/>

4

What is the name of the method that allows xpath expressions in SimpleXML?

A: The is no such method

B: xPath

C: xpathExpression

D: xpath

5 **What is the XSL extension in PHP doing?**

A: Formatting the XML output

B: Applying style sheets to the XML

C: Applying XML transformations

D: Checking the syntax of an XML document for validity

6 **What is the output of the following code?**

```php
1  <?php
2
3  $xml = '<root>
4      <parent name="Peter">
5          <child age="20">James</child>
6          <child age="5">Leila</child>
7      </parent>
8      <parent name="Anna">
9          <child age="10">Dido</child>
10         <child age="11">George</child>
11     </parent>
12 </root>';
13
14 $xmlElement = new SimpleXMLElement($xml);
15 $teens = $xmlElement->xpath('*/child[@age>9]');
16 print $teens[1];
```

A: It will print James

B: It will print nothing

C: It will cause a Run-time error

D: It will print Dido

7

Which web services are supported natively in PHP? (Choose two)

A: SOAP

B: REST

C: XML-RPC

D: Corba

8

Which of the following is true about SOAP and PHP?

A: It uses the JSON data format in PHP

B: Only Soap Clients can be created in PHP

C: Every PHP class can be used automatically as a soap service by adding a special parameter to the URL

D: Soap Clients in PHP are hiding the complexity of sending a request to a remote Soap Server and processing the response

9

What is the purpose of this HTTP request?

PUT /user/123
<?xml ...?

A: Add user to collection 123

B: Replace user 123

C: Replace the collection 123

D: Update the user in collection 123

10

What is JSON?

A: A way of serializing any PHP type in order to exchange it with different programming languages and systems

B: A portable XML representation of the data using PHP's serialize($value, true)

C: A format to represent any PHP type, except a resource, that can be used later on in JavaScript or other languages

TEST YOUR KNOWLEDGE : ANSWERS

1   D: Nothing

2   E: None of the above

3   A: `<?xml version="1.0"?>`
         `<root/> **`

4   D: xpath

5   C: Applying XML transformations

6   D: It will print Dido

7   A: SOAP   *and*   C: XML-RPC

8   D: Soap Clients in PHP are hiding the complexity of sending a request to a remote Soap Server and processing the response

9

B: Replace user 123

10

C: A format to represent any PHP type, except a
resource, that can be used later on in JavaScript

STRINGS: SNAPSHOT 📷 TOPICS...

Quoting

Matching

Extracting

Searching

Replacing

Formatting

PCRE

HEREDOC & NOWDOC

Encodings

STRINGS & PATTERNS 🔍 FOCUS

- **DELIMITED BY SINGLE OR DOUBLE QUOTES**
    - DOUBLE QUOTES OFFER MORE OPTIONS, INCLUDING SPECIAL CHARACTERS

- **HEREDOC SYNTAX**
    - DELIMITS STRINGS WITHOUT USING QUOTES (SO NO NEED TO ESCAPE)
    - START WITH <<< AND AN IDENTIFIER; END WITH SAME IDENTIFIER
    - DO NOT INDENT ENDING IDENTIFIER OR ADD ANY CHARS

- **NOWDOC SYNTAX**
    - SIMILAR TO HEREDOC, BUT NO PARSING IS CONDUCTED
    - SAME <<< IDENTIFIER, BUT THE IDENTIFIER SHOULD BE ENCLOSED IN SINGLE QUOTES

- **SUBSTRINGS**
    - USE THE `substr(`*`string, start, length`*`)` FUNCTION
    - RETURNS A SUBSTRING POSITION OF FIRST OCCURRENCE OR `FALSE`

- **LOCATING STRINGS:**
    - USE THE `strpos(`*`file, string, start`*`)` FUNCTION
    - RETURNS THE POSITION OF FIRST OCCURRENCE OR `FALSE`

- **COMPARING STRINGS**

`==`	SETS UP COMPARISON, INCLUDING DATA TYPE CONVERSION
`===`	SETS UP COMPARISON, INCLUDING DATA TYPE CHECK
`strcasecmp()`	CASE-*IN*SENSITIVE COMPARISON
`strcmp()`	CASE-SENSITIVE COMPARISON
`similar_text()`	SIMILARITY OF TWO STRINGS... RETURNS THE NUMBER OF MATCHING CHARS

  ```
  echo simlar_text("cat", "can"); //2
  ```

  `levenshtein()` LEVENSHTEIN DISTANCE BETWEEN STRINGS... DEFINED AS MINIMUM NUMBER OF CHARS NEEDED TO REPLACE, INSERT, OR DELETE TO TRANSFORM STRING 1 > STRING 2

  ```
  echo levenshtein("cat", "can"); //1
  ```

- **COUNTING STRINGS:**

  - **NUMBER OF CHARACTERS**

    USE THE `strlen(`*`string`*`)` FUNCTION

  - **NUMBER OF WORDS**

    USE `str_word_count(`*`string`*`);` `str_word_count(`*`strings`*`,` *`true`*`)` YIELDS AN ARRAY WITH ALL SINGLE WORDS

- **PHONETIC FUNCTIONS**

`soundex()`	SOUNDEX VALUE OF A STRING
`metaphone()`	METAPHONE KEY OF A STRING

  BASED ON ENGLISH PRONUNCIATION RULES, SO MORE PRECISION THAN THE `soundex()` FUNCTION , BUT OF LIMITED USE WITH GLOBAL SITES

- **STRINGS AND ARRAYS:**
  - `explode(`*`split string, string`*`)` CONVERTS A STRING INTO AN ARRAY
  - `implode(`*`glue string, string`*`)` CONVERTS AN ARRAY INTO A STRING

- **FORMATTING OUTPUT**

  `printf()`   PRINTS A FORMATTED STRING

  `sprintf()`   RETURNS A FORMATTED STRING

  `vprintf()`   PRINTS A FORMATTED STRING, PLACEHOLDER VALUES SUPPLIED AS AN ARRAY

  `vsprintf()`   RETURNS A FORMATTED STRING, PLACEHOLDER VALUES SUPPLIED AS AN ARRAY

  `fprintf()`   SENDS A FORMATTED STRING TO A RESOURCE

- **FORMATTING CHARACTERS (PARTIAL LISTING)**

  `%b` (BINARY)

  `%d` (DECIMAL)   `%nd` (N IS THE NUMBER OF DIGITS)

  `%f` (FLOAT)   `%.nf` (N IS THE NUMBER OF DECIMAL PLACES)

  `%o` (OCTAL)

  `%e` (SCIENTIFIC NOTATION)

  `%s` (STRING)

- **REGULAR EXPRESSIONS**
  - DESCRIBE A PATTERN
  - TWO KINDS IN PHP: `POSIX-RegEx` (NOT COVERED BY EXAM) AND `PCRE` (PERL COMPATIBLE REGULAR EXPRESSION)
  - DELIMITER
    - USUALLY `"/"`, `"#"`, OR `"!"`
    - USED AT BEGINNING AND END OF EACH PATTERN
  - LITERALS ARE ANY CHARACTERS

  - BOUNDARIES (EXAMPLES)

`^`	START OF A LINE
`$`	END OF A LINE
`\A`	START OF A STRING
`\Z`	END OF A STRING

  - CHARACTER CLASSES DELIMITED WITH `[ ]`
    - BUILT-IN CHARACTER CLASSES; CAPITALIZATION INDICATES ABSENCE (EXAMPLE)

`\d`	DIGIT
`\D`	NO DIGIT

  - "GREEDINESS"
    - MAXIMUM MATCH IS RETURNED
    - USUALLY NEED TO USE PARENTHESES WITH ALTERNATIVES

- **REGULAR EXPRESSIONS (CONTINUED)**

  - QUANTIFIERS (EXAMPLES)

    *   ANY NUMBER OF TIMES

    +   ANY NUMBER OF TIMES, BUT AT LEAST ONCE

    ?   0 OR 1

    *COMBINATION OF ? WITH * OR + MAKES NON-GREEDY*

  - PATTERN MATCHING

    - USE THE `preg_match(`*`pattern, string`*`)` FUNCTION
    - RETURNS NUMBER OF MATCHES
    - OPTIONAL THIRD PARAM DEFINES MATCH
    - `preg_match_all()` RETURNS ALL MATCHES
    - RETURNS ALL MATCHES IN AN ARRAY

  - REPLACING
    `preg_replace(`*`search pattern, replace pattern, string`*`)`

- **ENCODINGS**

  - SOME LANGUAGE CHARACTER SETS CAN BE REPRESENTED WITH SINGLEBYTE ENCODINGS (BASED ON 8-BIT VALUES; EX: LATIN-BASED LANGUAGES) AND OTHERS REQUIRE MULTIBYTE ENCODINGS BECAUSE OF THEIR COMPLEXITY (EX: CHINESE LOGOGRAPHIC CHARACTER SET)

  - OPERATING WITH STRINGS IN MULTIBYTE ENCODING REQUIRES USING SPECIAL FUNCTIONS (`mbstring`) OR THE CHARACTERS WILL DISPLAY INCORRECTLY

- **ENCODINGS (CONTINUED)**
  - EXISTING APPLICATIONS BUILT IN A SINGLEBYTE ENVIRONMENT, THAT UTILIZE FUNCTIONS LIKE `substr()` AND `strlen()`, WILL NOT WORK PROPERLY IN MULTIBYTE ENVIRONMENTS
  - NEED TO EMPLOY FUNCTION OVERLOADING, TO CONVERT SINGLEBYTE FUNCTION AWARENESS TO A MULTIBYTE EQUIVALENT, SUCH AS `mb_substr()` AND `mb_strlen()`
  - MBSTRING MODULE:
    - HANDLES CHARACTER ENCODING CONVERSION
    - DESIGNED FOR UNICODE-BASED (UTF-8, UCS-2) AND SOME SINGLE-BYTE ENCODINGS (MANUAL HAS COMPLETE LIST...SEE REFS)
    - MODULE MUST BE ENABLED USING THE `configure` OPTION (NOT A DEFAULT EXTENSION)
    - `mb_check_encoding()` WILL VERIFY WHETHER THE STRING IS VALID FOR THE SPECIFIED ENCODING

REFERENCES:

http://us3.php.net/manual/en/book.strings.php

http://www.php.net/manual/en/mbstring.supported-encodings.php

TEST YOUR KNOWLEDGE : QUESTIONS

1

What is a good rule to follow when quoting string data?

> **A:** Use double quotes because you might want to use variable interpolation at a later time
>
> **B:** Use single quotes unless you are using variable interpolation because single quotes are faster
>
> **C:** Use single quotes unless you have a ' in your string or you are doing variable interpolation because it declares whether you want variables to be interpolated

2

What is will the output of this code be?

```
echo strcmp(12345, '12345');
```

> **A:** Less than zero because (int)12345 is less than (string)'12345'
>
> **B:** Zero because (int)12345 is equal to (string)'12345'
>
> **C:** Greater than zero because (int)12345 is greater than (string)'12345'

3

Given a string '$str = '12345';' what is the pattern required to extract each digit individually?

A: $result = sscanf($str, '%d');

B: $result = sscanf($str, '%d%d%d%d%d');

C: $result = sscanf($str, '%1d%1d%1d%1d%1d');

4

What will the following code print out?

```
$str = 'abcdef';
if (strpos($str, 'a')) {
    echo "Found the letter 'a'";
} else {
    echo "Could not find the letter 'a'";
}
```

????

5

What will this code do?

```
$var = 2;
$str = 'aabbccddeeaabbccdd';
echo str_replace('a', 'z', $str, $var);
```

A: Replace all of the 'a' characters with 'z' characters and put the replacement count in $var

B: Replace up to 2 of the 'a' characters with a 'z' character

C: 2 is a flag which, when passed to str_replace, will remove all characters except those listed

6

What will the following code print?

```php
$str = printf('%.1f', 5.3);
echo 'Zend PHP Certification ';
echo $str;
```

A: Zend PHP Certification 5.3

B: Zend PHP Certification

C: 5.3Zend PHP Certification 3

7

What will this code output?

```php
$data = 'hệɩɩọ ₩ọṛɪɖ';
$matches = array();
preg_match('/./u', $data, $matches);

echo $matches[0];
```

A: An unprintable character because PHP does not understand UTF-8

B: ĥ , because PCRE can understand UTF-8

C: Nothing, because PHP does not understand UTF-8

8

What is the key difference between HEREDOC and NOWDOC?

A: NOWDOC allows you to use block delimiters with a single
quote

B: HEREDOC terminates a block starting at the first
character, but NOWDOC allows you to indent the end of
the block

C: NOWDOC does not parse for variable interpolation, but

9

What will the following code print?

```php
$a = 'ĥ℮ʟʟℚ ₩ℴℙℐɖ';
echo strlen($a);
```

A: 1, since the space is the only ASCII character in the
string

B: 26, since PHP does not natively understand UTF-8
encoding

C: 10, since it only counts the first byte of a UTF-8 encoded
character

TEST YOUR KNOWLEDGE : ANSWERS

1   **C:** Use single quotes unless you have a ' in your string or you are doing variable interpolation because it declares whether you want variables to be interpolated

2   **B:** Zero because (int)12345 is equal to (string)'12345'

3   **C:** $result = sscanf($str, '%1d%1d%1d%1d%1d');

4   Could not find the letter 'a'

5   **A:** Replace all of the 'a' characters with 'z' characters and put the replacement count in $var

6   **C:** 5.3Zend PHP Certification 3

7   **B:** ḧ , because PCRE can understand UTF-8

8   **C:** NOWDOC does not parse for variable interpolation, but HEREDOC does

9   **B:** 26, since PHP does not natively understand UTF-8 encoding

ARRAYS: SNAPSHOT TOPICS...

- Enumerated Arrays
- Associative Arrays
- Multi-dimensional Arrays
- Array Iteration
- Array Functions
- SPL / Objects as Arrays

ARRAYS - FOCUS

- **ARRAY DEFINITION**
    - WAY OF ORDERING DATA BY ASSOCIATING VALUES TO KEYS
    - UNIQUE KEYS ARE ASSOCIATED WITH A SINGLE VALUE, OR SET OF VALUES
    - ARRAYS CAN BE NESTED, SO THAT A VALUE IN ONE ARRAY ACTUALLY REPRESENTS A COMPLETE OTHER ARRAY (MULTI-DIMENSIONAL ARRAYS)

- **CREATING ARRAYS**
    - **INDEXED NUMERICALLY** (INDEXED ARRAY)
        - EX: `$x = array('a', 'b', 'c');`
        - EX: `$x = array(0 => 'a', 1 => 'b', 2 => 'c');`
    - **INDEXED WITH STRINGS** (ASSOCIATIVE ARRAY)
      ```
      $x = array(
          'XML' => 'eXtensible Markup Language'
      );
      ```

- **FILLING ARRAYS**
    - `range()` CREATES AN ARRAY WITH VALUES FROM AN INTERVAL
        - DEFAULT STEP IS "`1`"
        - EX: `$x = range(1.2, 4.1) // == array(1.2, 2.2, 3.2)`

- **SPLITTING ARRAYS**
    - `array_slice(`*`array, offset`*`)` RETURNS PART OF AN ARRAY
        - OPTIONAL 3RD PARAM = LENGTH
        - OPTIONAL 4TH PARAM = MAINTAIN INDICES (BOOLEAN)
    - NEGATIVE OFFSET MEANS COUNT FROM THE END OF THE ARRAY

- **SPLITTING ARRAYS (CONTINUED)**
    - NEGATIVE LENGTH EXCLUDE ELEMENTS x POSITIONS FROM THE END OF THE ARRAY
        - Ex: `$x = array(1,2,3,4,5)`
          `$y = array_slice($x, -4, -1); //== array(2,3,4);`

- **ADDING ELEMENTS**
    - `array_push()` ADDS 1 OR MORE ELEMENTS TO THE END OF AN ARRAY
    - ARRAY IS PROVIDED BY REFERENCE
    - RETURN VALUE IS THE NEW NUMBER OF ARRAY ELEMENTS
        - Ex: `$x = array(1,2,3);`
          `$n = array_push($x,4,5); // $n == 5`
        - ALTERNATIVE: `$n[] = 4;   $n[] = 5;`
    - `array_unshift()` ADDS 1 OR MORE ELEMENTS TO THE BEGINNING OF AN ARRAY
    - ALREADY EXISTING ELEMENTS ARE MOVED TOWARDS THE END
    - RETURN VALUE IS THE NEW NUMBER OF ARRAY ELEMENTS
        - Ex: `$x = array(3,4,5);`
          `$n = array_unshift($x,1,2); // $n == 5`

- **REMOVING ELEMENTS**
    - `array_pop()` REMOVES 1 ELEMENT AT THE END OF AN ARRAY
    - ARRAY IS PROVIDED BY REFERENCE
    - RETURN VALUE IS THE REMOVED ELEMENT
        - Ex: `$x = array(1,2,3);`
          `$n = array_pop($x); // $n == 3`

    - `array_shift()` REMOVES 1 ELEMENT AT THE BEGINNING OF AN ARRAY
    - REMAINING ELEMENTS ARE MOVED TOWARDS THE FRONT
    - RETURN VALUE IS THE REMOVED ELEMENT
        - Ex: `$x = array(1,2,3);`
          `$n = array_shift($x); // $n == 1`

- **LOOPING ARRAYS**
    - `for` LOOP AND INDICES
        - Ex:
          ```
          for ($i = 0; $i < count($a); $i++) {
                echo $a[$i];
          }
          ```

    - `foreach` LOOP AND VALUES
        - Ex:
          ```
          foreach ($a as $value) {
                echo $value . '<br />';
          }
          ```

    - `foreach` LOOP AND KEYS AND VALUES
        - Ex:
          ```
          foreach ($a as $key => $value) {
                echo "$key: $value<br />";
          }
          ```

    - `array_walk()` PROVIDES ACCESS TO ALL ARRAY ELEMENTS
        - A CALLBACK FUNCTION IS USED FOR EACH ELEMENT

- **CHECKING FOR ARRAY VALUES**
    - `array_key_exists($key, $array)` = DETERMINES WHETHER THERE IS AN INDEX `$key` IN THE ARRAY `$array`
    - `in_array($element, $array)` = DETERMINES WHETHER THERE IS AN ELEMENT `$element` IN THE ARRAY `$array`
    - `array_keys()` IS AN ARRAY OF ALL ARRAY INDICES
    - `array_values()` IS AN ARRAY OF ALL ARRAY VALUES

- **SORTING ARRAYS**

  - `sort($a)` SORTS VALUES ALPHABETICALLY

  - THE SECOND PARAMETER INDICATES THE SORT MODE

    - `SORT_LOCALE_STRING` SORTS ACCORDING TO LOCALE SETTINGS

    - `SORT_NUMERIC` NUMERIC SORTING

    - `SORT_REGULAR` "NORMAL" SORTING (DEFAULT)

    - SORT_STRING SORTING AS STRINGS

- **OTHER SORTING FUNCTIONS**

  `rsort()` LIKE `sort()`, BUT IN REVERSE

  `asort()` SORTS ASSOCIATIVE ARRAYS (MAINTAINS KEY-VLAUE)

  `arsort()` LIKE `asort()`, BUT IN REVERSE

  `ksort()` SORTS BY KEYS

  `krsort()` LIKE `ksort()`, BUT IN REVERSE

  `usort()` USER-DEFINED SORT

- **NATURAL SORTING**

  - `natsort()` RETURNS RESULTS BASED ON HOW A HUMAN WOULD SEE ORDER (*9.PHP > *10.PHP > *11.PHP) "NATURAL" STRING SORTING VS. (*10.PHP > *11.PHP > *9.PHP) "NORMAL" STRING SORTING

- **MERGING ARRAYS**

  - `array_merge($x, $y)` CREATES AN ARRAY CONTAINING THE ELEMENTS OF BOTH ARRAYS, X AND Y

- **COMPARING ARRAYS**

    - `array_diff($x, $y)` COMPARES THE TWO ARRAYS, X AND Y

    - RETURN VALUE IS AN ARRAY WITH ALL ELEMENTS IN $x NOT IN $y

    - RELATED FUNCTIONS:

`array_diff_assoc()`	COMPARES VALUES AND KEYS
`array_diff_key()`	COMPARES ONLY KEYS
`array_diff_uassoc()`	LIKE `array_diff_assoc()` BUT WITH USER-DEFINED COMPARE FUNCTION
`array_diff_ukey()`	LIKE `array_diff_key()` BUT WITH USER-DEFINED COMPARE FUNCTION

- **SPL - OBJECTARRAY CLASS**

    - CLASS ALLOWS OBJECTS TO FUNCTION AS ARRAYS

`ArrayObject::STD_PROP_LIST`	PROPERTIES ARE RETAINED WHEN ACCESSED AS A LIST (EX: `var_dump`, `foreach`)
`ArrayObject::ARRAY_AS_PROPS`	ENTRIES CAN BE ACCESSED AS PROPERTIES (EX: READ/WRITE)

    - RELATED ARRAYOBJECTS (SELECTION):

`ArrayObject::append`	APPENDS A VALUE
`ArrayObject::asort`	SORTS THE ENTRIES BY VALUE
`ArrayObject::natsort`	SORTS ACCORDING TO A "NATURAL ORDER"... SEE `natsort()` ABOVE

REFERENCES:

http://php.net/manual/en/ref.array.php

TEST YOUR KNOWLEDGE : QUESTIONS

1

What is the output of the following code ?

```php
<?php
    $a = array(1, 2, 3);
    foreach ($a as $x) {
        $x *= 2;
    }
    echo $a[0] * $a[1] * $a[2];
    ?>
```

????

2

What is the output of the following code ?

```php
<?php
    $a = array(1,2,4,8);
    $b = array(0,2,4,6,8,10);
    echo count(
        array_merge(
            array_diff($a, $b),
            array_diff($b, $a)
        )
    );
?>
```

????

3

What is the output of the following code ?

```php
<?php
    $a = array(
        "1" => "A", 1 => "B", "C", 2 => "D");
    echo count($a);
?>
```

????

4 **Which of the following will generate an E_NOTICE error assuming the following code?**

```php
1  <?php
2
3  $array = array(
4      array(
5          1, 2
6      ),
7      'a' => array(
8          'b' => 1,
9          'c'
10     )
11 );
```

A: $array[] = 1;

B: echo $array[5][2];

C: echo $array[5][2] = 2;

D: isset($array[7][3][1]);

---

5 **Read carefully: Which interface can be used to allow an object to be executed in a foreach loop?**

A: ArrayObject

B: Iterator

C: ArrayList

D: Hashtable

---

6 **Given the following PHP code, which of these answers creates a valid associative array in PHP?**

```php
1  <?php
2
3  $one = array ('one', 'two','three');
4  $two = array (1, 2, 3);
```

A: array_combine($one, two)

B: array_merge($one, two)

C: array_values($two)

D: array_flip($one)

1

6

2

4

3

2

4

```
B:   echo $array[5][2];
```

5

```
B:   Iterator
```

6

```
A:   array_combine($one, two)    and
D:   array_flip($one)
```

INPUT / OUTPUT: SNAPSHOT 📷 TOPICS...

Files

Filesystem Functions

Streams

Contexts

Reading

Writing

INPUT / OUTPUT - FOCUS

FILES and FILESYSTEM FUNCTIONS

- **TWO MAIN TYPES OF FUNCTIONS:**

  `f*():`   FUNCTIONS THAT WORK WITH A FILE RESOURCE
              EX: `fopen()`

  `file*():` FUNCTIONS THAT WORK WITH A FILENAME:
              EX: `file_get_contents()`

- **FILES WITH RESOURCES**

  - USER ASSIGNED A UNIQUE IDENTIFIER, THE "SESSION ID"

    - CREATE A FILE RESOURCE WITH `fopen()`

      - 1ST PARAMETER: `File name` (REQUIRED)

      - 2ND PARAMETER: `File mode` (REQUIRED)

    - READ WITH `fread()`

    - EX:

      ```
      $fp = fopen('file.txt', 'r');
      while (!feof($fp)) {
          echo htmlspecialchars(fread($fp, 4096));
      }
      fclose($fp);
      ```
      OR
      ```
      echo htmlspecialchars(
      fread($fp, filesize('file.txt'));
      ```

- **WRITE TO RESOURCES**
  - `fwrite()` AND `fputs()` WRITE DATA INTO A RESOURCE
    - EX:
      ```
      $fp = fopen('file.txt', 'w');
      fwrite($fp, 'data...');
      fclose($fp);
      ```

    OTHER FUNCTIONS
  - `fputcsv():` WRITES AN ARRAY IN CSV FORMAT INTO A FILE
  - `fprintf(): printf()` FOR RESOURCES

- **OUTPUT FILES**
  - `fpassthru():` OUTPUTS ALL THE DATA OF A FILE HANDLE DIRECTLY TO THE OUTPUT BUFFER; STARTS AT CURRENT FILE POSITION

    Using `fread()` PLUS ESCAPING SPECIAL CHARACTERS IS OFTEN A BETTER ALTERNATIVE

- **FILE OPERATIONS (ONLY PARTIAL LIST... SEE PHP MANUAL)**
  - **DIRECTORY**

    `chdir():`   CHANGES THE DIRECTORY

    `chroot():`   CHANGES THE ROOT DIRECTORY

    `readdir():`   READS AN ENTRY FROM THE DIRECTORY HANDLE

    `rmdir():`   DELETES A DIRECTORY

  - **FILE INFORMATION**

    `finfo_open():` CREATE A NEW FILEINFO RESOURCE

    `finfo_file():` RETURNS INFORMATION ABOUT A FILE

  - **FILESYSTEM**

    `basename():`   RETURNS FILENAME COMPONENT OF A PATH

    `chmod():`   CHANGES THE FILE MODE

    `copy():`   COPIES A FILE

    `file_exists:` CHECKS IF A FILE OR DIRECTORY EXISTS

    `fpassthru():` OUTPUTS ALL DATA OF A FILE HANDLE DIRECTLY TO THE OUTPUT BUFFER (STARTING AT THE CURRENT FILE POSITION)

    `fputcsv():`   WRITES DATA INTO A RESOURCE
    `fputs()`

    `rename():`   MOVES/RENAMES A FILE

    `unlink():`   DELETES A FILE

STREAMS

- PROVIDE A WAY OF GROUPING AND MAKING AVAILABLE OPERATIONS WHICH HAVE FUNCTIONS AND ACTIONS IN COMMON

- **PARTS OF A DATA STREAM:**

    WRAPPER

    PIPELINES

    CONTEXT

    META DATA

- **FILE WRAPPERS**

    - PROVIDES INFORMATION ON PROTOCOLS AND ENCODINGS

        - CAN BE ANY FILE WRAPPER

        - ALLOWS FOR TWO PIPELINES AT MOST - FOR READING & WRITING

    - PREFIX IN FRONT OF A FILE PATH

        ```
        file://          php://

        http://          compress.zlib://

        https://         compress.bzip2://

        ftp://           ftps://
        ```

    - CUSTOM WRAPPERS

        - `stream_wrapper_register(protocol, classname)` REGISTERS A PROTOCOL; IMPLEMENTATION IS PART OF THE CLASS

- CUSTOM WRAPPERS (CONTINUED)
  - THE CLASS IMPLEMENTS STANDARD FUNCTIONALITY LIKE READING, WRITING, OR CHANGING THE FILE POSITION
  - `php_user_filter` IS A PREDEFINED CLASS IN PHP AND IS USED IN CONJUNCTION WITH USER-DEFINED FILTERS

- **PIPELINES / TRANSPORT**
  - CODE WRAPPER COMMUNICATION
  - CONTEXT: ADDITIONAL INFORMATION FOR A STREAM (EX: HTTP HEADERS FOR HTTP STREAMS)
  - META DATA: CAN BE DETERMINED WITH `stream_get_meta_data()`

- **STREAM CONTEXTS**
  - SET OF PARAMETERS AND WRAPPER OPTIONS THAT CAN MODIFY A STREAM'S BEHAVIOR
  - CREATE CONTEXTS WITH `stream_context_create()`
    - OPTIONS CAN BE SPECIFIED WHEN THE FUNCTION IS CALLED
    - PARAMETERS CAN BE SPECIFIED WITH `stream_context_set_params()`
  - CURRENT OPTIONS FOR A GIVEN STREAM CAN BE DETERMINED BY CALLING `stream_context_get_options`

- **STREAM FILTERS**
  - CAN BE APPLIED TO STREAM DATA

    `stream_filter_append($fp, 'filtername');`
  - CAN CREATE CUSTOM FILTERS

    `stream_filter_register(filtername, classname);`
  - CLASS IMPLEMENTS THE FOLLOWING METHOD

    `function filter($in, $out, &$consumed, $closing);`

READING and WRITING

- **READ IN THE COMPLETE CONTENTS OF A FILE**

  ```
  string file_get_contents(string Åfilename [, â~~→
  Åuse_include_pathKàá→bæ [, resource Åcontext
  ÁÁ[, int ÅoffsetKËFÁYÊÁ↔^\ÁÅ↑á[→æ^]]]Ÿ)
  ```

- **READ A FILE DELIMITED BY LINE INTO AN ARRAY**

  ```
  array file(string filename [, int use_include_path])
  ```

- **READ AND OUTPUT A FILE TO THE OUTPUT BUFFER**

  ```
  int readfile (string filename [, int use_include_path])
  ```

- **WRITE DATA INTO A FILE**

  ```
  file_put_contents(string filename, mixed data [, int
  flags [, resource context]] )
  ```

- **WRITE TO RESOURCES**

  ```
  fwrite() AND fputs() ... WRITE DATA INTO A RESOURCE
  $fp = fopen('file.txt', 'w');
  fwrite($fp, 'data...');
  fclose($fp);
  ```

- **WRITE TO STREAMS**

  ```
  fprintf(): printf FOR RESOURCES
  ```

REFERENCE:

http://www.php.net/manual/en/book.stream.php

http://us2.php.net/manual/en/ref.filesystem.php

TEST YOUR KNOWLEDGE : QUESTIONS

1

Which function can be used to read and parse data from a CSV file?

????

2

What is the output of the following function call (assuming that foo.txt exists and contains text)?

```
$output = file(""foo.txt"");
```

A: A file handle that can be used in subsequent calls such as fread

B: True if the file could successfully be read, false if not

C: A string containing the contents of foo.txt

D: An array where every entry is a line from the file foo.txt

E. True if the file exists, false if not

3

What happens if you use fwrite to write data to a readonly file?

A: A PHP fatal error occurs

B: False is returned

C: An exception is thrown

D: A PHP warning occurs

4

Consider the following snippet of code. What is the name of the function that needs to be inserted in the placeholder?

```
$dh = opendir("".".");
while ($file = _____($dh)) {
    echo $file;

}
```

????

5

Which of the following is NOT a default PHP input or output stream?

A: php://stdin

B. php://stdout

C. php://stderr

D. php://input

E. php://output

F. php://error

6

Which of the following functions does not accept a stream $context parameter?

A: fopen

B: fgets

C: file_get_contents

D: file

TEST YOUR KNOWLEDGE : ANSWERS

1 | `fgetcsv`

2 | `D: An array where every entry is a line from the file foo.txt`

3 | `B: False is returned`

4 | `readdir`

5 | `F: php://error`

6 | `B: fgets`

FUNCTIONS: SNAPSHOT 📷 TOPICS...

Syntax

Arguments

Variables

References

Returns

Variable Scope

Anonymous Functions (Closures)

82

FUNCTIONS - FOCUS

- **FUNCTION DEFINITION**

  - PACKAGES OF CODE THAT SERVE AS INSTRUCTIONS TO EXECUTE AN ACTION

    - ANY VALID CODE CAN BE USED, INCLUDING FUNCTIONS AND CLASSES

  - CASE-INSENSITIVE; GLOBAL SCOPE

  - CAN BE REFERENCED BEFORE BEING DEFINED UNLESS FUNCTION CONDITIONAL

  - TYPES: BUILT-IN (PHP SUPPLIED); USER-DEFINED; EXTERNALLY PROVIDED

- **DECLARING FUNCTIONS**

  - PARAMETERS AND RETURN VALUE OPTIONAL; SET PARAM DEFAULT TO AVOID WARNING

  - EX:
    ```
    function myFunction ($p)  {
          // do something
          return $p;
    }
    $x = myFunction("ABC");  //$x == "ABC"
    $x = myFunction();  //warning!
    ```

  - EX:
    ```
    function myFunction ($p = "ABC")  {
          // do something
          return $p;
    }
    $x = myFunction("DEF");  //$x == "DEF"
    $x = myFunction();       //$x == "ABC"
    ```

- **FUNCTION ARGUMENTS**

  - `func_num_args()`   NUMBER OF PARAMETERS

  - `func_get_arg(nr)`   PARAMETER NUMBER "NR"S

  - `func_get_args()`   ALL PARAMETERS AS AN ARRAY

  - ARGUMENT LIST IS A SET OF COMMA-DELIMITED EXPRESSIONS

  - CAN PASS ARGUMENTS IN SEVERAL WAYS

    - BY VALUE (DEFAULT)

      - CREATES COPY: ARGUMENT CHANGES EXTEND ONLY WITHIN FUNCTION

    - BY REFERENCE

      - USE "`&`" TO SUPPLY PARAMETERS BY REFERENCE

    - BY DEFAULT ARGUMENT VALUES (PARAMETERS)

    - CHANGES TO ANY REFERENCE AFFECTS ALL REFERENCES

- **RETURN VALUES**

  - `return()` STATEMENT ENDS FUNCTION EXECUTION

  - WILL RETURN VALUES THAT INCLUDE ARRAYS, OBJECTS, FUNCTION REFERENCES (USING `&`)

- **VARIABLE SCOPE**

  - VARIABLES DECLARED WITHIN FUNCTIONS ONLY VISIBLE IN THAT FUNCTION

  - VARIABLES DECLARED OUTSIDE OF FUNCTIONS ARE VISIBLE EVERYWHERE OUTSIDE OF FUNCTIONS

  - VARIABLES DECLARED OUTSIDE OF FUNCTIONS CAN BE MADE VISIBLE WITHIN A FUNCTION USING `GLOBAL`

- **VARIABLE FUNCTIONS**
  - WORK LIKE VARIABLE VARIABLES
  - VARIABLES FOLLOWED BY PARENTHESES CAUSES SEARCH FOR, AND EXECUTION OF, FUNCTION WITH SAME NAME AS VARIABLE EVALUATION
  - COMMONLY USED FOR CALLBACKS, FUNCTION TABLES
  - SOME USE RESTRICTIONS WITH COMMON CONSTRUCTS
    - EX: `echo(), print()`

- **ANONYMOUS FUNCTIONS (CLOSURES)**
  - ENABLE CREATION OF FUNCTIONS WITHOUT SPECIFYING A NAME
  - IMPLEMENTED USING THE `Closure` CLASS
  - COMMONLY USED AS PARAM VALUE FOR CALLBACK FUNCTIONS, OR ALTERNATIVELY AS VARIABLE VALUES
  - TO INHERIT VARIABLES FROM PARENT SCOPE (FUNCTION IN WHICH CLOSURE WAS DECLARED), THESE VARIABLES MUST BE DECLARED IN FUNCTION HEADER

REFERENCE:

http://www.php.net/manual/en/language.functions.php

1

What is the output of the following code? (Choose 2)

```php
<?php
    function addValues() {
        $sum = 0;
        for ($i = 1; $i <= func_num_args(); $i++) {
            $sum += func_get_arg($i);
        }
        return $sum;
    }
    echo addValues(1,2,3);
?>
```

A: 5

B: 6

C: A parser error

D: A warning

2

Take a look at the following code...

```php
<?php
function myFunction($a) {
   $a++;
}
$b = 1;
myFunction($b);
?>
```

What code do you need to replace so that $b has the value 2 at the end of the script?

A: Line 02: Replace $a with &$a

B: Line 03: Replace $a++ with $a +=2;

C: Line 03: Replace $a++ with $a *=2;

D: Line 06: Replace $b with &$b

3

What is the output of the following code?

```php
<?php
$v1 = 1;
$v2 = 2;
$v3 = 3;
function myFunction() {
   $GLOBALS['v1'] *= 2;
   $v2 *= 2;
   global $v3; $v3 *= 2;
}
myFunction();
echo "$v1$v2$v3";
?>
```

A: 123

B: 246

C: 226

D: 126

4

What is the output of the following code?

```php
<?php

function increment ($val)
{
    return ++$val;
}

echo increment (1);

?>
```

????

5

What is the output of the following code?

```php
1  <?php
2
3  function func($x, $x=1, $x=2) {
4      return $x;
5  }
6
7  print func(3);
```

A: Syntax error

B: 3 will be printed

C: 2 will be printed

D: Nothing will be printed

6

What is the output of the following code?

```php
1  <?php
2
3  $x = function func($a, $b, $c) {
4      print "$c|$b|$a\n";
5  };
6
7  print $x(1,2,3);
```

A: Syntax error

B: 3|2|1

C: 1|2|3

7

What is the best way to test if $param is an anonymous function in a method?

A: Use method_exists($param, '__invoke')

B: Use is_callable($param)

C: Use the type-hint Closure on the signature

D: Use is_executable($param)

1

```
A: 5 and D: a warning
```

2

```
A: Line 02: Replace $a with &$a
   (excluding D: because of deprecation)
```

3

```
C: 226
```

4

```
2
```

5

```
C: 2 will be printed
```

6

```
A: Syntax error
```

7

```
B: Use is_callable($param) **
```

OOP : SNAPSHOT   TOPICS...

- Instantiation
- Instance Methods & Properties
- Class Definition
- Modifiers / Inheritance Abstracts
- Interfaces
- Exceptions
- Static Methods & Properties
- Autoload
- Reflection
- Type Hinting

OOP : SNAPSHOT   TOPICS…

Class Constants

Late Static Binding

Magic Methods

SPL

OBJECT-ORIENTED PROGRAMMING   🔍   FOCUS

OBJECTS

- **CONVERTING OBJECTS TO STRINGS**

    - THE SPECIAL METHOD `__toString()` IS CALLED, IF AVAILABLE, WHENEVER A STRING IS EXPECTED - INCLUDING PRINT, STRING INTERPOLATION, OPERATION WITH STRINGS, CALLING FUNCTIONS THAT EXPECT STRINGS, ...

- **COPYING OBJECTS**

    - KEYWORD: `clone`

    - OBJECTS ARE ALWAYS PASSED BY REFERENCE

    - CLONING AN OBJECT CAUSES THE OBJECT ITSELF TO BE COPIED INSTEAD OF PASSING THE REFERENCE

        - CLONING BY DEFAULT COPIES ALL THE PROPERTIES, BUT USES ASSIGNMENT, NOT CLONE, SO CLONING IS "SHALLOW" BY DEFAULT

    - PHP EXECUTES THE SPECIAL METHOD `__clone()` UPON CLONING, IF AVAILABLE

- **SERIALIZING OBJECTS**

    - FUNCTIONS: `serialize() / unserialize()`

- **SERIALIZING OBJECTS (CONTINUED)**
  - SPECIAL METHOD `__sleep()` IS EXECTUED WITH SERIALIZATION, IF AVAILABLE
    - ALLOWS YOU TO SPECIFY WHICH PROPETIES SHOULD BE STORED (SERIALIZED) AND WHICH SHOULD NOT BE
    - CAN ALSO CREATE/CHANGE PROPERTIES FOR SERIALIZATION
  - SPECIAL METHOD `__wakeup()` IS EXECUTED WITH DESERIALIZATION, IF AVAILABLE
    - EX: TO OPEN A DATABASE CONECTION UNIQUE TO THE OBJECT

CREATING CLASSES AND INSTANTIATION

- KEYWORD: `class`

- A CLASS DEFINES THE ABSTRACT CHARACTERISTICS OF AN OBJECT, INCLUDING ITS ATTRIBUTES AND ACTIONS

- PROPERTIES AND METHODS DEFINED BY A CLASS ARE CALLED "MEMBERS"

- STRUCTURE:

  KEYWORD > CLASS NAME > { CONSTANTS, PROPERTIES & METHODS }
  WHERE PROPERTIES = CLASS VARIABLES, AND METHODS = CLASS FUNCTIONS

- CREATE AN INSTANCE OF A CLASS WITH THE KEYWORD "`new`"

  - AN OBJECT IS CREATED UNLESS IT HAS A CONSTRUCTOR DEFINED THAT THROWS AN EXCEPTION WITH AN ERROR

  - CLASSES SHOULD BE DEFINED "PRIOR" TO INSTANTIATION

    - WITH AUTOLOADING, A CLASS CAN BE DEFINED (LOADED) AT THE MOMENT IT IS REQUIRED BY THE INSTANTIATING OPERATOR

  - ASSIGNING AN EXISTING INSTANCE OF A CLASS TO A NEW VARIABLE (OR FUNCTION) RESULTS IN THE VARIABLE ACCESSING THE SAME INSTANCE AS THE ASSIGNED OBJECT

- Ex:

```
class myClass {
    // ...
}
$c = new myClass();
```

- **INHERITANCE: CLASS**
  - USE THE KEYWORD `extends` IN THE CLASS DECLARATION TO HAVE A CLASS INHERIT THE METHODS AND PROPERTIES OF ANOTHER CLASS,
    - A CLASS CAN INHERIT FROM ONLY ONE CLASS, NOT BE EXTENDED TO MULTIPLE CLASSES
    - INHERITED METHODS AND PROPERTIES CAN BE OVERRIDDEN BY REDECLARING THEM WITH SAME NAME
  - WHENEVER AN EXTENDING CLASS OVERRIDES THE PARENTS' METHOD DEFINITION, THAT METHOD WILL NOT BE AUTOMATICALLY CALLED
    - SIMILARLY FOR CONSTRUCTORS/DESTRUCTORS, OVERLOADING, AND MAGIC METHODS
    - CHILD CLASSES CANNOT OVERRIDE A PARENT PROPERTY OR METHOD USING A LOWER VISIBILITY
      - FOR EXAMPLE, IF `classA` HAS A PUBLIC METHOD CALLED `getA()`, `classB` WHICH EXTENDS `classA` CANNOT DECLARE A METHOD CALLED `getA()` AND DECLARE IT PRIVATE
    - CLASSES AND METHODS MARKED WITH FINAL CANNOT BE OVERRIDDEN
    - THE PARAMETER SIGNATURE CANNOT BE "STRICTER" THAN BEFORE or AN `E_STRICT` ERROR WILL BE THROWN (EXCEPT FOR THE CONSTRUCTOR)

- **ABSTRACT CLASSES**
  - KEYWORD: `abstract`
  - PROVIDES A SKELETON FOR A CLASS
  - MAY CONTAIN IMPLEMENTATIONS
  - ABSTRACT METHODS MUST BE IMPLEMENTED IN DERIVED CLASSES
  - VISIBILITY CAN BECOME WEAKER / MORE PERMISSIVE, BUT NOT STRONGER / LESS PERMISSIVE (EX: YOU CANNOT GO FROM PUBLIC TO PRIVATE)

INTERFACES

- KEYWORD: `interface`, `implements`

- PROVIDES METHODS TO IMPLEMENT

    - DOES NOT CONTAIN ANY IMPLEMENTATION ITSELF

- DERIVED CLASSES MAY IMPLEMENT MORE THAN ONE INTERFACE

- INTERFACES MAY INHERIT FROM OTHER INTERFACES USING THE `extends` KEYWORD

- ALL METHODS ARE ASSUMED TO BE PUBLIC IN THE INTERFACE DEFINITION - CAN BE DEFINED EXPLICITLY AS PUBLIC, OR IMPLICITLY

- WHEN A CLASS IMPLEMENTS MULTIPLE INTERFACES THERE CANNOT BE ANY NAMING COLLISION BETWEEN METHODS DEFINED IN THE DIFFERENT INTERFACES

EXCEPTIONS

- KEYWORD: `throw` ... TO LAUNCH AN EXCEPTION

- CATCH WITH: `try ... catch`

    - `Catch` MAY ALSO WAIT FOR SPECIFIC EXCEPTIONS

    - NEED TO PROVIDE THE TYPE IN THE CATCH

        - TYPE MAY BE AN EXCEPTION EXTENDED FROM ANOTHER

- CUSTOM EXCEPTIONS NEED TO EXTEND THE BASE `Exception` CLASS

CONSTRUCTORS / DESTRUCTORS

- `__construct()` IS A RESERVED METHOD NAME FOR THE CLASS CONSTRUCTOR

- `function __construct()` IS USED TO DECLARE A CONSTRUCTOR CLASS METHOD

  - THESE METHODS ARE USED WITH NEW OBJECTS AS PREPARATION FOR INITIALIZATION BEFORE USE

- `__destruct()` IS A RESERVED METHOD NAME FOR THE CLASS DESTRUCTOR

  - IF A CLASS MAINTAINS AN OPEN FILE HANDLE OR CONNECTION THROUGHOUT ITS LIFE, THEN THE `__destruct()` METHOD IS A GOOD PLACE FOR A CLOSE-TYPE OPERATION

- `__destruct()` IS CALLED WHENEVER AN OBJECT IS DESTROYED (WHEN ALL ITS REFERENCES ARE REMOVED OR THE END OF THE SCRIPT) IS REACHED

METHODS & PROPERTIES

- **PROPERTIES (VARIABLES)**

  - CLASS MEMBER VARIABLES ARE CALLED PROPERTIES OR ATTRIBUTES

  - VISIBILITY KEYWORDS: PUBLIC, PRIVATE, PROTECTED

  - DECLARED LIKE ANY VARIABLE; IF INITIALIZED, MUST BE WITH A CONSTANT VALUE

  - `nowdocs` CAN BE USED TO INITIALIZE A PROPERTY

  - CREATING A VARIABLE WITHIN THE CLASS... EX:

    ```
    class myClass {
    public $member = "ABC";
        // ...
    }
    $c = new myClass();
    echo $c->member;
    ```

- **METHODS (FUNCTIONS)**

  - SET OF PROCEDURAL STATEMENTS

  - IF VISIBILITY IS NOT EXPLICITLY DEFINED, THEN DEFAULT IS PUBLIC

  - CAN ACCESS PROPERTIES OR METHODS OF THE CURRENT INSTANCE USING `$this` (FORMAT `$this->property`), FOR NON-STATIC PROPERTIES

  - EX:
    ```
    class myClass {
        public $member = "ABC";
        function showMember() {
            echo $this->member;
        }
    }
    $c = new myClass();
    $c->showMember();
    ```

- **STATIC PROPERTIES / METHODS**
  - KEYWORD: `static`
  - OPERATOR (::)
    - TOKEN THAT PERMITS ACCESS TO THE STATIC, CONSTANT, OR OVERRIDDEN PROPERTIES / METHODS OF A CLASS
      - USE THE CLASS NAME WHENEVER REFERENCING THESE ELEMENTS OUTSIDE OF THE CLASS DEFINITION
      - SELF ALWAYS REFERS TO THE CURRENT CLASS; PARENT REFERS TO THE PARENT OF THE CURRENT CLASS (I.E., THE ONE IT EXTENDS)
      - THE STATIC CONTEXT CAN BE DEFINED AS WORKING WITH THE CLASS AND NOT AN OBJECT OF THE CLASS. IT IS USEFUL WHEN YOU ARE WORKING WITH DATA THAT IS NOT TIED TO A PARTICULAR INSTANCE OF A CLASS
  - REQUIRES DECLARATION, AS WITH ANY METHOD; OTHERWISE RESULTS IN A FATAL ERROR
  - NO INSTANTIATION REQUIRED
  - WITH V5.3, YOU CAN NOW ACCESS A STATIC CLASS METHOD USING A VARIABLE REFERENCE (EX: `ClassName::$varMethod`)

- **AUTOLOAD**
  - PHP EXECUTES THE `__autoload()` FUNCTION WHENEVER THERE IS AN ATTEMPT TO USE A CLASS OR INTERFACE THAT HAS NOT BEEN DEFINED
    - PARAM: NAME OF MISSING CLASS
  - EXCEPTIONS THROWN IN `__autoload()` CAN NOW BE CAUGHT IN A CATCH BLOCK, AS LONG AS THE CUSTOM EXCEPTION CLASS IS AVAILABLE
    - `autoload()` CAN RECURSIVELY LOAD THE CUSTOM EXCEPTION CLASS

- **AUTOLOAD (CONTINUED)**
  - `spl_autoload()` IS USED AS AN IMPLEMENTATION FOR `__autoload()`
    - CALL `spl_autoload_register()`, WHICH WILL REGISTER A FUNCTION AS AN `__autoload()` IMPLEMENTATION
    - BOOLEAN: IT PREPENDS THE AUTOLOADER ON THE AUTOLOAD STACK WHEN TRUE; APPENDS WHEN FALSE

- **REFLECTION**
  - ALLOWS FOR INTROSPECTION OF:
    - OBJECTS
    - CLASSES
    - METHODS
    - PROPERTIES
    - FUNCTIONS
    - PARAMETERS
    - EXCEPTIONS
    - EXTENSIONS
  - HELPER CLASSES FORMAT = `ReflectionXXX` (WHERE `XXX` = OBJECT, CLASS, …)

- **TYPE HINTING**
  - DATA TYPES MAY BE PROVIDED FOR FUNCTION & METHOD PARAMETERS
    - CLASSES
    - ARRAYS
  - IF THE DATA TYPE DOES NOT MATCH, A FATAL ERROR OCCURS
  - CLASS TYPE MATCHES EITHER EXACT TYPE OR ANY TYPE THAT EXTENDS OR IMPLEMENTS (IN THE CASE OF INTERFACES) THIS TYPE
  - AS LONG AS THE TYPE-HINTED CLASS EXISTS SOMEWHERE BELOW THE PASSED CLASS' HIERARCHY, IT WILL BE ALLOWED

- **CLASS CONSTANTS**
  - A SPECIAL ENTITY THAT REMAINS FIXED ON AN INDIVIDUAL CLASS BASIS, AND DOES NOT EXHIBIT THE USUAL "`$`" VARIABLE SYMBOL
    - SIMILAR IN CONCEPT TO A CONSTANT THAT IS RE-DEFINED USING `define()`
  - INTERFACES MAY ALSO INCLUDE CONSTANTS
  - WHEN CALLING A CLASS CONSTANT USING THE `$classname::CONSTANT` SYNTAX, THE CLASSNAME CAN ACTUALLY BE A VARIABLE
  - WITH V5.3, YOU CAN NOW ACCESS A STATIC CLASS CONSTANT USING A VARIABLE REFERENCE (EX: `ClassName::$varConstant`)

- **LATE STATIC BINDING**

    - SUPPORTED BEGINNING WITH PHP V5.3

    - USED FOR RETRIEVING THE CALLER CLASS INFORMATION WHEN STATIC CALL TO INHERITED METHOD IS MADE, SO THERE'S NO OBJECT AVAILABLE TO CARRY THE CLASS INFO.

        - STORES THE CLASS NAMED IN THE LAST "NON-FORWARDING" CALL

        - STATIC METHOD CALLS CLASS EXPLICITLY NAMED (`name::xx`)

    - STATIC REFERENCES (Ex: `self::xx`) USE THE CURRENT CLASS TO WHICH THE FUNCTION BELONGS

- **MAGIC METHODS**

    - When accessing non-existent properties, PHP will execute special ("magic") functions, if available   Ex:

        `__get()`   Reads a property

        `__set()`   Writes a property

        `__isset()` Checks if the property is set

        `__unset()` Unsets or destroys a property

    - When accessing non-existent methods, PHP will execute the special `__call()` function, if available

    - With PHP v5.3, there is a new `__callStatic()` magic method, which allows the calling of non-existent static methods (must be public)

- **SPL**
  - Acronym for "Standard PHP Library"
  - Examples:
    - `ArrayIterator`
      - It creates a stand-alone iterator object over an array, which allows it to iterate over the same array multiple times and also passes the iteration state around in an object
      - Ex: current element, next element
      - Allows `foreach` access
    - `ArrayObject`
      - Interface that implements an array
      - Ex: number of elements, read/write access
      - Allows access to the object using array functions

REFERENCE:

http://www.php.net/manual/en/language.oop5.php

TEST YOUR KNOWLEDGE : QUESTIONS

1 What is the relationship between classes and objects?

> A: A class is a collection of objects
>
> B: A class is a template from which objects are made
>
> C: Objects are distinguished from one another by assigning them to a class
>
> D: Classes and objects are variable types

2 What is the output of the following code?

```php
<?php
  interface myBaseClass1 {
    public function doSomething();
    public function specialFunction1();
  }
  interface myBaseClass2 {
    public function doSomething();
    public function specialFunction2();
  }
  class myClassA implements myBaseClass1, myBaseClass2 {
    function doSomething() {
      echo '...';
    }
    function mySpecialFunction1() {
      echo '...';
    }
    function mySpecialFunction2() {
      echo '...';
    }
  }
  $a = new myClassA();
  $a->doSomething();
?>
```

> A: ...
>
> B: Parser error
>
> C: Fatal error
>
> D: None of the above

3

What is the output of the following code?

```php
<?php
  abstract class myBaseClass {
    abstract protected function doSomething();
    function threeDots() {
      return '...';
    }
  }
  class myClassA extends myBaseClass {
    protected function doSomething() {
      echo $this->threeDots();
    }
  }
  $a = new myClassA();
  $a->doSomething();
?>
```

A: ...

B: Parser error

C: Fatal error

D: None of the above

4

Which of the following statements about _exceptions_ is NOT true?

A: Only objects of class Exception and classes extending it can be thrown

B: It is recommended that catch(Exception) be the last catch clause

C: Exceptions can be re-thrown after being caught

D: Uncaught exceptions always cause fatal errors

5

Which of the following statements about <u>static functions</u> is true?

> **A: Static functions can only access static properties of the class**
>
> **B: Static functions cannot be called from non-static functions**
>
> **C: Static functions cannot be abstract**
>
> **D: Static functions cannot be inherited**

6

Which of the following statements about <u>autoloading</u> is true?

> **A: Autoloading is executed whenever the name of an unidentified class is encountered in the code**
>
> **B: Multiple autoloading functions can be defined using spl_autoload_register()**
>
> **C: Autoloading should be avoided due to high performance penalties**
>
> **D: An autoloading function should throw an**

7

Which of the following <u>CANNOT</u> be a part of the class definition?

> **A: Constant**
>
> **B: Variable**
>
> **C: Function**
>
> **D: Interface**

8 **Reflection functions CANNOT ...**

> A: Instantiate objects
>
> B: Modify static properties of the class
>
> C: Get the namespace name of a class
>
> D: Modify static variables in functions

9 **Of the following statements about typehints, which is NOT true?**

> A: Typehinted parameters can default to NULL
>
> B: A typehint class does not have to be defined when a function definition is parsed
>
> C: Objects should be of the same class to satisfy typehinting
>
> D: Typehints cannot be PHP scalar types

10 **Which is the correct syntax to define a class constant for the class MyClass?**

> A: const $NAME="value";
>
> B: Define("MyClass::NAME", "value");
>
> C: const NAME="value";
>
> D: static final $NAME='value';

11

What is the output of the following code?

```php
<?php
class Magic {
    public $a = "A";
    protected $b = array("a" => "A", "b" => "B", "c" => "C");
    protected $c = array(1,2,3);

    public function __get($v) {
        echo "$v,";
        return $this->b[$v];
    }

    public function __set($var, $val) {
        echo "$var: $val,";
        $this->$var = $val;
    }
}
$m = new Magic();
echo $m->a.",".$m->b.",".$m->c.",";
$m->c = "CC";
echo $m->a.",".$m->b.",".$m->c;
?>
```

A: A, Array, Array, A, Array, Array, CC

B: b, c, A, B, C, c: CC, b, c, A, B, C

C: a, b, c, A, B, C, c: CC, a, b, c, A, B, C

D: b, c, A, B, C, c: CC, b, c, A, B, CC

12

Which statement about SPLObjectStorage class is NOT true?

A: It uses objects as indexes

B: It can be used to implement sets of objects

C: It allows arbitrary data to be associated with an object

D: It permits the serialization of any object

TEST YOUR KNOWLEDGE : ANSWERS

1   B: A class is a template from which objects are made

2   C: Fatal Error

3   C: Fatal Error

4   D: Uncaught exceptions always cause fatal errors

5   A: Static functions can only access static properties of the class

6   B: Multiple autoloading functions can be defined using spl_autoload_register()

7   D: Interface

8   D: Modify static variables in functions

9   C: Objects should be of the same class to satisfy typehinting

10   C: const NAME="value";

11   B: b, c, A, B, C, c: CC, b, c, A, B, C

12   D: It permits the serialization of any object

13   B: B

14   C: 12123

DATABASES: SNAPSHOT TOPICS...

- SQL
- Joins
- Analyzing Queries
- Prepared Statements
- Transactions
- PDO

DATABASES: SNAPSHOT

DATABASES - FOCUS 🔍

- **DEFINITION**

  - WAY OF STORING AND RETRIEVING DATA EFFICIENTLY

- **KEYS**

  - **PRIMARY KEY:** COLUMN OF UNIQUE VALUES THAT DESCRIBE AN ENTRY IN THE DATA TABLE

  - **FOREIGN KEY**: PRIMARY KEY FROM ANOTHER TABLE; ENABLES RELATIONAL DATABASES

- **SQL**

  - **CREATE A DATABASE**:

    ```
    CREATE TABLE tbl (
        id INT NOT NULL PRIMARY KEY,
        field1 VARCHAR(100),
        field2 CHAR(32) NOT NULL
    )
    ```

    - NOTE: NULL IS NOT THE SAME AS THE NUMBER `"0"`, `"false"`, OR AN EMPTY STRING... IT REPRESENTS "NO VALUE" OR "MISSING VALUE"

  - **READ DATA**:

    ```
    SELECT field1, field2 FROM tbl
        WHERE field3 = 'value'
    ```

  - **INSERT DATA**:

    ```
    INSERT INTO tbl
        (field1, field2, field3) VALUES
        ('value1', 2, 'value3')
    ```

- **SQL (CONTINUED)**
  - **UPDATE DATA:**
    ```
    UPDATE tbl
        SET field1 = 'value1', field2 = 'value2'
        WHERE field3 = 'value3'
    ```

  - **DELETE DATA:**
    ```
    DELETE FROM tbl WHERE field1 = 'value1'
    DROP TABLE tbl
    DROP DATABASE tbl
    ```

- **SORTING (ORDER BY)**
  - **ORDER BY ASCENDING (ASC) OR DESCENDING (DESC)**
    ```
    SELECT * FROM tbl ORDER BY col DESC
    ```

- **GROUPING (GROUP BY)**
  - IN GENERAL, THE COLUMNS USED TO GROUP BY MUST BE INCLUDED IN THE SELECT LIST
    ```
    SELECT col1, col2 FROM tbl
        GROUP BY col1
    ```

- **AGGREGATION**

AVG()	AVERAGE VALUE
COUNT()	NUMBER OF ELEMENTS
DISTINCT COUNT()	NUMBER OF DISTINCT ELEMENTS
MIN()	MINIMAL VALUE
MAX()	MAXIMUM VALUE
SUM()	SUM OF VALUES

- **JOINS**

  ### INNER JOIN

  EX: RETURNS ALL ENTRIES IN TAB1 AND TAB2 LINKED USING THE PRIMARY/FOREIGN KEY, AND THAT FULFILL THE WHERE CLAUSE IN TAB1

  ```
  SELECT * FROM tab1
      INNER JOIN tab2
      ON tab1.primkey = tab2.forkey
      WHERE tab1.col1 = 'value1'
  ```

  ### LEFT JOIN

  EX: ALL DATA FROM THE "LEFT" TABLE IS USED, EVEN IF THERE IS NO MATCH IN THE "RIGHT" TABLE

  ```
  SELECT * FROM tab1
      LEFT JOIN tab2
      ON tab1.primkey = tab2.forkey
      WHERE tab1.col1 = 'value1'
  ```

  ### RIGHT JOIN

  EX: ALL DATA FROM THE "RIGHT" TABLE IS USED, EVEN IF THERE IS NO MATCH IN THE "LEFT" TABLE

  ```
  SELECT * FROM tab1
      RIGHT JOIN tab2
      ON tab1.primkey = tab2.forkey
      WHERE tab2.col1 = 'value1'
  ```

- **PREPARED STATEMENTS**
  - SIMILAR IN CONCEPT TO TEMPLATES - CONTAIN COMPILED CODE USED TO RUN COMMON SQL OPERATIONS
  - ADVANTAGES:
    - QUERY ONLY PARSED ONCE, BUT ALLOWS FOR MULTIPLE EXECUTIONS, WITH SAME OR DIFFERENT PARAMETERS (PERFORMANCE CONSIDERATION)
    - RELATED PARAMETERS DO NOT NEED TO BE QUOTED (SECURITY CONSIDERATION)
  - ONLY FEATURE PDO WILL EMULATE FOR ADAPTERS THAT DO NOT SUPPORT PREPARED STATEMENTS

- **TRANSACTIONS**
  - COMBINES INDIVIDUAL SQL OPERATIONS INTO ONE
  - USUALLY START WITH `BEGIN` OR `BEGIN TRANSACTION`
  - EXECUTE THE TRANSACTION USING COMMIT
  - CANCEL THE TRANSACTION USING `ROLLBACK`

- **PDO (PHP Data Objects extension)**
    - Provides interface for accessing databases - a data-access abstraction layer
        - Can use the same functions to manipulate databases, regardless of DB type
        - NOT for data type or SQL abstraction
    - Must use database-specific PDO adapters to access a DB server
        - Database adapters implementing PDO interfaces expose database-specific features as regular extension functions
    - Runtime Configuration Options:
        - `pdo.dsn.*` in `php.ini`
        - `PDO::setAttribute()`
    - Set of predefined class constants available
    - Error settings available: `Silent`, `Warning`, and `Exception`

CONNECTIONS

- Connections are made by creating an instance of the PDO class, *not* by creating instances of `PDOStatement` or `PDOException`
- Ex: Connecting to MySQL

```php
<?php
$dbh = new
PDO('mysql:host=localhost;
    dbname=test', $user, $pass);
?>
```

QUERIES

`PDO::query()`

EXECUTES A SQL STATEMENT, IN A SINGLE FUNCTION CALL, AND RETURNS THE RESULTING VALUES AS A `PDOStatement` OBJECT

NEED TO RETRIEVE ALL DATA IN THE RESULT SET BEFORE CALLING QUERY FUNCTION AGAIN

FETCH

`PDOStatement->setFetchMode`

SETS THE DEFAULT FETCH MODE (EX: `FETCH_COLUMN`)

TRANSACTIONS

`PDO::beginTransaction()`

TURNS OFF AUTOCOMMIT MODE FOR CHANGES MADE TO THE DATABASE

`PDO::commit()`

CALL TO END TRANSACTION AND COMMIT CHANGES

`PDO::rollBack()`

CALL TO REVERSE ALL CHANGES MADE TO THE DATABASE AND REACTIVATE AUTOCOMMIT MODE

- **PDOSTATEMENT**
  - ONLY VALUES CAN BE BOUND (*NOT* ENTITIES, SUCH AS TABLE NAMES AND COLUMN NAMES)
  - ONLY SCALARS CAN BE BOUND TO THE VALUES (NOT ARRAYS OR NULLS)

`PDO::prepare()` AND `PDOStatement::execute()`

`PDO::prepare()` IS USED TO PREP THE OBJECT , WHILE `PDOStatement::execute()` IS USED TO ISSUE THE STATEMENT

IF USING PARAMS, MUST EITHER PASS AN ARRAY OF INPUT PARAM VALUES, OR CALL `PDOStatement::bindParam()` TO BIND THE PARAM MARKERS TO THE CORRESPONDING VARIABLES

`PDOStatement::bindParam()`
BINDS THE VARIABLES IN A SQL STATEMENT TO THE CORRESPONDING PARAMETER MARKERS

`PDOStatement::bindValue()`
BINDS THE VARIABLE IN A SQL STATEMENT AS A REFERENCE

EVALUATED ONLY WHEN `PDOStatement::execute()` CALLED

`PDOStatement::closeCursor()`

FREES ANY RESOURCES TIED TO THE `PDOStatement` OBJECT

APPROPRIATE FOR A SINGLE ISSUE OF A `SELECT` STATEMENT

`PDO::exec()`

EXECUTES A SQL STATEMENT IN A SINGLE FUNCTION CALL, AND RETURNS THE NUMBER OF ROWS (NOT THE DATA) AFFECTED BY THE STATEMENT

APPROPRIATE FOR MULTIPLE CALLS TO A `SELECT` STATEMENT

REFERENCES:

http://php.net/manual/en/refs.database.php
http://dev.mysql.com/doc/
http://us.php.net/manual/en/intro.pdo.php

TEST YOUR KNOWLEDGE : QUESTIONS

1

Given the following table called "names" ...

```
pos    |    name    |    email
----   |    ------- |    ------------------
-2     |    anna    |    anna@example.com
-1     |    betty   |    betty@example.com
NULL   |    clara   |    clara@example.com
1      |    demi    |    demi@example.com
2      |    emma    |    emma@example.com
3      |    gabi    |    gabi@example.com
```

... how many rows will be returned from the following query?

SELECT * FROM names WHERE pos < 10

A: 3
B: 4
C: 5
D: 6

2

Given the following table called "names"...

```
id   |   name
---  |   -------
1    |   anna
2    |   betty
3    |   clara
4    |   demi
5    |   emma
```

... and the following table called "emails"

```
id   |   email
---  |   -----------------
1    |   anna@example.com
3    |   clara@example.com
5    |   emma@example.com
7    |   gabi@example.com
9    |   julia@example.com
```

... how many rows will be returned from the following query?

```
SELECT names.name, emails.email
 FROM names
JOIN emails ON emails.id = names.id
```

A: 3
B: 5
C: 9
D: 10

3

Given the following table called "names"...

```
id  |  name    |  email
--- | ------- | ------------------
1   |  anna    |  anna@example.com
2   |  betty   |  betty@example.com
3   |  clara   |  clara@example.com
4   |  anna    |  anna@example.com
5   |  betty   |  betty@example.com
6   |  clara   |  clara@example.com
```

.
... what will the COUNT() value be when the following PHP code runs? (Assume PDO connection is valid)

```php
$pdo = new PDO(...);
$sql = "SELECT :cols FROM names WHERE name = :name";
$stmt = $pdo->prepare($sql);
$stmt->bindValue(':cols', 'COUNT(id)');
$stmt->bindValue(':name', 'anna');
$stmt->execute();
```

4

Given the following table called "names" ...

```
id |  name    |  email
-- | ------- | ------------------
1  |  anna    |  anna@example.com
2  |  betty   |  betty@example.com
3  |  clara   |  clara@example.com
```

... and the following PDO code (assume PDO connection is valid)...

```php
$pdo = new PDO(...);
$pdo->begin();
$pdo->query("INSERT INTO NAMES (name, email) VALUES
        ('demi', 'demi@example.com')");
$stmt = $pdo->query('SELECT COUNT(*) FROM names');
$count1 = $stmt->fetchColumn();
$pdo->rollBack();
$stmt = $pdo->query('SELECT COUNT(*) FROM names');
$count2 = $stmt->fetchColumn();
```

... what is the value of `$count2` **?**

5 **Given the following table called "names" ...**

```
id | name    | email
-- | ------- | ------------------
1  | anna    | anna@example.com
2  | betty   | betty@example.com
3  | clara   | clara@example.com
```

A: 'anna'

B: 'betty'

C: 'clara'

D: NULL

... what is the value of $name at the end of the following PHP code? (Assume PDO connection is valid)

```php
$pdo = new PDO(...);
$name = null;
$stmt = $pdo->prepare('SELECT * FROM names WHERE name =
                :name');
$stmt->bindValue(':name', 'anna');
$stmt->execute();
while ($row = $stmt->fetch()) {
    var_dump($name);
}
```

TEST YOUR KNOWLEDGE : ANSWERS

1

```
5
```

2

```
3
```

3

```
NONE - the prepared statement is invalid
```

4

```
3
```

5

```
NULL
```

SECURITY : SNAPSHOT   📷   TOPICS...

Configuration

Session Security

Cross-Site Scripting

Cross-Site Request Forgeries

SQL Injection

Remote Code Injection

Email Injection

Input Filtering

Escape Output

Encryption, Hashing Algorithms

FUNCTIONS

OOP

DATABASES

SECURITY

WEB FEATURES

SECURITY : SNAPSHOT   TOPICS...

File Uploads

Data Storage

SSL

SECURITY 🔍 FOCUS

CONFIGURATION

- **GENERAL SETTINGS**
    - `register_globals` SET TO `off`
    - `display_error` SET TO `off`, `log_errors` SET TO `on`
    - `allow_url_include` SET TO `off`
    - `error_reporting = E_ALL & ~E_DEPRECATED`

- **USING PHP AS A CGI BINARY**

    PHP HAS BUILT-IN SAFEGUARDS AGAINST COMMON ATTACK SCHEMES INVOLVING INTERPRETERS, ALONG WITH CONFIGURABLE SETTINGS FOR ADDED SECURITY:
    - **ACCESSING SYSTEM FILES**: PHP DOES NOT INTERPRET COMMAND LINE ARGUMENTS PASSED BY THE INTERPRETER TO THE CGI INTERFACE
    - **ACCESSING PRIVATE DOCUMENTS**: RUNTIME DIRECTIVES `cgi.force_redirect`, `doc_root`, AND `user_dir` CAN BE USED TO OVERCOME SECURITY VULNERABILITIES IN SERVER SETUPS WHEN DEALING WITH RESTRICTED DIRECTORIES.
    - **ACCESSING PUBLIC FILES**: THE OPTION `--enable-force-cgi-redirect` CAN BE ADDED TO THE CONFIGURE SCRIPT FOR SERVERS THAT DO NOT ALLOW REDIRECTS OR DO NOT HAVE A WAY TO CONFIRM A REQUEST HAS BEEN SAFELY REDIRECTED.
    - **DIRECTLY CALLING PHP**: THE CONFIGURATION DIRECTIVE `cgi.force_redirect` BLOCKS THE ABILITY TO CALL PHP DIRECTLY FROM A URL; DIRECTIVE WILL ALLOW PHP TO PARSE ONLY IF IT HAS BEEN REDIRECTED (APACHE WEB SERVER)
    - **PARSER:** OPTIONALLY, PLACE THE PHP PARSER BINARY OUTSIDE OF THE WEB TREE

- **USING PHP AS A CGI BINARY (CONTINUED)**

  - **ACTIVE CONTENT(SCRIPTS, EXECUTABLES):** SET UP A SEPARATE SCRIPT DIRECTORY FOR EXECUTABLES, TO AVOID SECURITY ISSUES DUE TO DISPLAYING ACTIVE CONTENT AS HTML DOCUMENTS. SET THE DOCUMENT ROOT USING THE DIRECTIVE `doc_root` IN THE CONFIG FILE OR SET THE ENVIRONMENT VARIABLE `PHP_DOCUMENT_ROOT`; FILES WILL BE OPENED WITH THE `doc_root` AND PATH INFO IN THE REQUEST

    ANOTHER OPTION IS TO UTILIZE `user_dir`, WHICH WHEN UNSET CAUSES A REQUESTED FILE TO OPEN UNDER THE DOC_ROOT AND NOT THE USER'S HOME DIRECTORY (FILE FORMAT `~user/document.php`).

- **USING PHP INSTALLED AS AN APACHE MODULE**

  PHP IN THIS CONFIGURATION WILL INHERIT THE PERMISSIONS STRUCTURE OF THE APACHE SERVER. COMMON SECURITY STEPS TO TAKE INCLUDE:

  - SET THE APACHE AUTHORIZATION (VS. USING DEFAULT 'NOBODY' SETTING)

  - CREATE AN ACCESS MODEL USING `.htaccess` FILES, LDAP, ...

  - DO NOT ENDOW THE USER WITH ROOT PERMISSION (PERMIT SUDO'ING, CHROOT'ING); INSTEAD, USE `open_basedir` TO CONTROL DIRECTORY USE

- **FILESYSTEM SECURITY**

  - ONLY ALLOW LIMITED PERMISSIONS TO THE APACHE WEB USER BINARY

  - CHECK ALL VARIABLES SUBMITTED

- **ERROR HANDLING**

  - DISPLAY ERRORS ONLY IN A DEVELOPMENT ENVIRONMENT; IN PRODUCTION, `display_errors = off` and `log_errors = on`

  - USE HIGH ERROR REPORTING SETTINGS `error_reporting = E_ALL | E_STRICT`

SESSION SECURITY

- **DESCRIPTION: SESSION HIJACKING**
  - OCCURS WHEN THE SESSION ID IS STOLEN
  - SESSION ID IS THE SOLE AUTHENTICATION TOKEN FOR THE WEB SITE; THEREFORE, THE USER'S SESSION IS HIJACKED

- **DESCRIPTION: SESSION FIXATION**
  - OCCURS WHEN USER GETS A "FIXED" SESSION ID (USUALLY VIA A SPECIALLY-CRAFTED URL)

- **COUNTER-MEASURES**
  - REGENERATE THE SESSION ID UPON LOGIN, BEFORE AUTHENTICATION
  - USE SSL ENCRYPTION FOR THE LOGIN, OR ASSIGN A HIDDEN KEY (NOT AS GOOD)
  - CHECK THAT THE IP ADDRESS REMAINS THE SAME (ALTHOUGH NOT ALWAYS RELIABLE)
  - CHANGE SESSION ID PRIOR TO "CRITICAL" OPERATIONS USING `session_regenerate_id()`
  - USE SHORT SESSION TIMEOUT
  - PROVIDE USER LOGOUT
  - DESTROY THE ORIGINAL SESSION BY PASSING TRUE `session_regenerate_id(true)`
  - USE PHP CONFIGURATION SETTING `session.use_only_cookies`

CROSS-SITE SCRIPTING

- **DESCRIPTION**

  - INJECTION OF HTML, CSS, OR SCRIPT CODE INTO A PAGE

    Ex: `<script>alert(document.cookie)</script>`

  - INSERTION COULD BE PERMANENT (INCORPORATED) OR VIA A LINK

  - JAVASCRIPT IS PARTICULARLY DANGEROUS BECAUSE OF ITS ABILITY TO:

    - REDIRECT THE USER

    - MODIFY THE PAGE

    - READ OUT COOKIES

- **COUNTER-MEASURES**

  - ESCAPE DATA BEFORE OUTPUTTING IT

    - `htmlspecialchars()`

    - `htmlentities()`

    - `strip_tags()`

  - BLACKLIST IS NOT A SUFFICIENT APPROACH

CROSS-SITE REQUEST FORGERIES

- **DESCRIPTION**

  - CREATES HTTP REQUESTS

  - ATTACKER EMPLOYS USER'S BROWSER TO EXECUTE REQUESTS ON THE ATTACKER'S BEHALF

  - RELIES ON WEB SITE TRUST OF LOGGED-IN USERS

  - ATTACKS USUALLY INVOLVES TRICKING A USER INTO TRANSMITTING 'BAD' HTML WITH A REQUEST, WHICH THEN RETURNS SENSITIVE DATA TO THE ATTACKER

    EXECUTED VIA IFRAMES OR VIA `XMLHttpRequest` REQUESTS OR `<script>, <object>, <embed>, <img>, …`
    EX:

    ```
    <form name="myForm">
    <input type="hidden" name="item_id" value="123" />
    <input type="hidden" name="quantity" value="1" />
    </form>
    <script>document.forms['myForm'].submit();</script>
    ```

- **COUNTER-MEASURES**

  - USE UNIQUE TOKEN IN THE FORM

  - REQUIRE RE-LOGIN BEFORE SENSITIVE (EX: FINANCIAL) OPERATIONS

SQL INJECTION

- **DESCRIPTION**

    - SQL CODE IS INJECTED INTO THE SQL QUERY

    - ALLOWS ATTACKER TO DO ALMOST ANYTHING THE DATABASE USER IS PERMITTED

    - RESULTING SQL STATEMENT ALWAYS RETURNS ALL THE DATA FROM THE 'USERS' TABLE

    - EX:

        ```
        $sql = "SELECT * FROM users WHERE
        username='$user' AND password='$pass'";
        $user and $pass contain the value ' OR ''='
        ```

    - FURTHER ATTACK POSSIBILITIES: INSERT DATA, DELETE DATA, READ DATA, DENIAL OF SERVICE...

- **COUNTER-MEASURES**

    - USE PREPARED STATEMENTS WHEN SUPPORTED BY THE DATABASE

    - USE DATABASE-SPECIFIC ESCAPING FUNCTIONS

        - EX: `mysqli_real_escape_string()`

    - `addslashes()` IS NOT A SUFFICIENT APPROACH

REMOTE CODE INJECTION

- REMOTE CODE INJECTIONS RUN THE ATTACKER'S CODE ON A USER'S MACHINE, OFTEN BY EXPLOITING THE FUNCTIONALITY OF THE `include` OR `require` FUNCTIONS

- EVEN IF A URL DOES NOT HAVE AN `include` FILE SPECIFIED, THE SITE IS STILL VULNERABLE IF IT USES AN EVAL FUNCTION THAT CONTAINS USER-SUPPLIED DATA

- OTHER VULNERABLE FUNCTIONS: `preg_replace` WITH THE /e PATTERN MODIFIER; `create_function()`

- **INCLUDE FILES ATTACKS**
  - OCCUR WHEN INCLUDING AND EXECUTING FILES
    - POSSIBLE FROM REMOTE SERVERS
    - INCLUDES REMOTE CODE EXECUTION

- **COUNTER-MEASURES**
  - CHECK DATA AGAINST A WHITELIST
  - REMOVE PATHS USING `basename()`
  - Set `allow_url_fopen = Off` in php.ini
    - HELPS SOMEWHAT BUT NOT SUFFICIENT, AS SOME ATTACK VECTORS REMAIN OPEN

- **DYNAMIC DATA CALL ATTACKS**
  - CODE INJECTION CAN OCCUR WHEN USING DYNAMIC DATA IN CALLS TO `system()` AND RELATED

- **COUNTER-MEASURES**
  - DO NOT USE `system()`
  - `escapeshellargs()` TO ESCAPE ARGUMENTS
  - `escapeshellcmd()` TO ESCAPE COMMANDS

EMAIL INJECTION

- **EMAIL / SMTP**
  - MAKE SURE NOT TO PROVIDE OPEN RELAYS
  - OPEN THE SMTP PORT ONLY IF ESSENTIAL
  - COULD UTILIZE "TARPITS", TO DELAY INCOMING CONNECTIONS AS A MEANS OF DISSUADING ATTACKS
    - POSSIBLE MECHANISM BECAUSE USER DOES NOT EXPECT AN IMMEDIATE RESPONSE WHEN USING EMAIL (WOULD NOT WORK WITH HTTP REQUESTS)

INPUT FILTERING

- **CHARACTER SET**
  - **Risk:**
    - ATTACK VECTORS MAY EMPLOY A NON-STANDARD CHAR SET (EX: UTF-8 ENCODED) THAT MAY BE MISSED BY FILTERING, BUT EXECUTED BY THE BROWSER
  - **Counter:**
    - USE THE SAME CHAR SET FOR FILTERING AS THE TARGET PROCEDURE
    - CONVERT CHARSETS PRIOR TO FILTERING
    - USE FILTERS NATIVE TO THE DATABASE (EX: DB QUOTING FUNCTIONS)
      ```
      content-Type: text/html; charset="UTF-8"
      ```

ESCAPING OUTPUT

- ONE OF TWO FUNDAMENTAL SECURITY RULES: (1) FILTER AND VALIDATE ALL INPUT; (2) ESCAPE OUTPUT

- ALWAYS ESCAPE USER-SUPPLIED CONTENT GIVEN IN HTML (UNLESS DATA HAS BEEN PREVIOUSLY FILTERED WITH A RESTRICTIVE FILTER)

- NEVER RELY ON CLIENT SIDE (JAVASCRIPT) FILTERING

- FUNCTIONS USED TO ESCAPE DATA BEFORE OUTPUTTING:

  ```
  htmlspecialchars()
  htmlentities()
  strip_tags()
  ```

ENCRYPTION, HASHING ALGORITHMS

- **PASSWORD SECURITY**
  - DO NOT SAVE PASSWORDS IN CLEARTEXT
  - USE HASH VALUES
    - `md5()`   32 CHARACTERS, HEXADECIMAL
    - `sha1()`   40 CHARACTERS, HEXADECIMAL
      - CANNOT BE REVERSED, BUT VULNERABLE TO A BRUTE-FORCE ATTACK
  - DO NOT USE HARD-CODING UNLESS VALUES ALSO HASHED

FILE UPLOADS

- `$_FILES` IS FILLED WITH USER-SUPPLIED DATA, AND THEREFORE POSES RISK
    - **RISK**: FILE NAME CAN BE FORGED
    - **COUNTER**: USE CHECKS AND `basename()`

    - **RISK**: `MIME` TYPE CAN BE FORGED
    - **COUNTER**: IGNORE

    - **RISK**: TEMP FILE NAME CAN BE FORGED UNDER CERTAIN CONDITIONS
    - **COUNTER**: USE `*_uploaded_file()` FUNCTIONS (`* = is, move`)

DATA STORAGE

- **DATABASE CONNECTIONS**
    - IF USING SQL TO MAKE CONNECTIONS, THE CODE IS SUBJECT TO SQL INJECTIONS (SEE SQL INJECTION SECTION)

- **DATABASE DESIGN**
    - EMPLOY PRINCIPLE OF LIMITED RIGHTS - ASSIGN ONLY THOSE PRIVILEGES THAT ARE NEEDED BY USER
    - NO EXPOSURE TO THE INTERNET UNLESS ESSENTIAL
    - ISOLATE DATABASES WITH SENSITIVE INFORMATION TO SEPARATE NETWORK SEGMENTS
    - CONTROL OUTGOING TRAFFIC OF WEB SERVERS
    - CHANGE STANDARD PASSWORDS AND ENCRYPT
    - READ THE LOGS

SSL

- SECURE SOCKET LAYER (SSL) ENCRYPTION PROTECTS DATA AS IT IS COMMUNICATED FROM THE CLIENT TO THE SERVER

  - SSH (SECURE SHELL PROTOCOL) ENCRYPTS THE NETWORK CONNECTION BETWEEN THE CLIENT AND THE DATABASE SERVER

- ENCRYPTION OF THE DATABASE DATA MAY BE AUGMENTED WITH THE USE OF THE PHP EXTENTIONS MCRYPT AND MHASH

  - ENCRYPT DATA BEFORE INSERTION AND DECRYPT WITH RETRIEVAL

- DATA THAT DOES NOT NEED TO BE PROCESSED CAN BE STORED AS A HASHED VALUE

REFERENCE:

http://www.php.net/manual/en/security.php

http://httpd.apache.org/docs/2.2/misc/security_tips.html

http://www.linuxsecurity.com/content/section/9/161/

TEST YOUR KNOWLEDGE : QUESTIONS

1

What is the recommended setting for error_reporting for production servers?

A: E_ALL & ~E_DEPRECATED

B: E_ALL & ~E_NOTICE

C: E_STRICT

D: OFF

2

How can you make it harder for JavaScript code to read out session IDs? (Choose 2)

A: Use the session_regenerate_id() function

B: Use the session_set_cookie_params() function

C: Use the session.cookie_httponly php.ini setting

D: Use the session.use_only_cookies php.ini setting

3

Which of the following measures provides good protection against Cross-Site Request Forgery?

A: Relying on HTTP POST only

B: Relying on the HTTP referer header

C: Relying on a one-time token

D: Relying on the user agent

4 Which potential security vulnerability is/vulnerabilities are in the following code?

```php
<?php
    echo htmlspecialchars($_GET['name']);
?>
<a href="<?php echo $_SERVER['PHP_SELF'] ?>">Reload</a>
```

A: Cross-Site Scripting (XSS)

B: Cross-Site Request Forgeries (CSRF)

C: Provoking an error message

D: None of the above

5 Which of PHP's database extensions does not support prepared statements?

A: ext/mysqli

B: ext/oci8

C: ext/pgsql

D: ext/sqlite

6 Which function does NOT provide ANY protection from remote command injection?

A: escapeshellcmd()

B: escapeshellarg()

C: htmlspecialchars()

D: strip_tags()

7

Your PHP application sends an email with data provided by the user, using PHP's mail() function. How can an attacker inject a custom BCC header to that email?

> A: Adding "\rBcc: email@example.com" to the subject
>
> B: Adding "\nBcc: email@example.com" to the mail body
>
> C: Adding "\r\nBcc: email@example.com" to the sender's address
>
> D: None of the above

8

Which of the following data may be altered by the user and should be filtered

> A: Querystring data
>
> B: HTTP referer
>
> C: Browser identification string
>
> D: All of the above

9

What is the output of the following code?

```
<CODE>
echo strlen(sha1('0', true));
</CODE>
```

> ???

10 Escaping output may help protect from which common security vulnerabilities? (Choose 2)

A: Clickjacking

B: Cross-Site Scripting

C: Cross-Ste Request Forgery

D: SQL Injection

11 What does the max_file_uploads configuration option contain?

A: The maximum number of file uploads per session

B: The maximum number of file uploads per request

C: The maximum number of file uploads per user

D: The maximum number of file uploads before the web service process is restarted

12 You are writing a PHP application that is used by thousands of people. You need to store database credentials in a secure fashion, but also want to make sure that the application can be easily deployed. What is the best way to achieve that?

A: In a .txt file inside the web folder

B: In an .inc file inside the web folder

C: In a .php file inside the web folder

D: In a .php file outside the web folder

13

What is the safest way to transport a password entered in a web form to the server?

A: Use JavaScript to hash the value, then send it to the server

B: Use JavaScript to encrypt the value, then sed it to the server

C: Use an HTTPS connection to the server

D: Use HTTP-only cookies

TEST YOUR KNOWLEDGE : ANSWERS

1
A: E_ALL & ~E_DEPRECATED

2
B: Use the session_set_cookie_params() function

C: Use the session.cookie_httponly php.ini setting

3
C: Relying on a one-time token

4
A: Cross-Site Scripting (XSS)

C: Provoking an error message

5
D: ext/sqlite

6
D: strip_tags()

7
D: None of the above

TEST YOUR KNOWLEDGE : ANSWERS

8

D: All of the above

9

20

10

B: Cross-Site Scripting (XSS)

D: SQL Injection

11

B: The maximum number of file uploads per request

12

C: In a .php file inside the web folder

13

C: Use an HTTPS connection to the server

WEB FEATURES: SNAPSHOT   TOPICS...

Sessions

Forms

GET and POST Data

Cookies

HTTP Headers and Codes

HTTP Authentication

FUNCTIONS

OOP

DATABASES

SECURITY

WEB FEATURES

WEB FEATURES - FOCUS

SESSIONS

- **DEFINITION**

  - WAY OF PRESERVING DATA ACROSS A SERIES OF WEB SITE ACCESSES BY THE USER

    - SESSION SUPPORT IS ENABLED BY DEFAULT

    - CONFIGURATION OPTIONS SET IN PHP.INI

    - SID(*STRING*) IS A PRE-DEFINED CONSTANT FOR THIS EXTENSION

- **SESSION ID**

  - USER ASSIGNED A UNIQUE IDENTIFIER, THE "SESSION ID"

    - SESSION ID IS STORED IN A COOKIE ON THE CLIENT OR IN THE URL

  - SITE ACCESS BY USER TRIGGERS SESSION ID CHECK THROUGH ONE OF THESE MECHANISMS:

    AUTOMATICALLY ... IF `session.auto_start = 1`

    UPON REQUEST ... USING `session_start()`

- **VARIABLES:**

  - `$_SESSION` IS AVAILABLE AS A GLOBAL VARIABLE

- **SECURITY MEASURES**

  - ENABLE `session.use_only_cookies` FOR DATA PROTECTION

- ## SESSION FUNCTIONS (PARTIAL LIST)

  session_cache_expire   RETURNS CURRENT CACHE EXPIRE

  session_destroy   DESTROYS ALL DATA REGISTERS TO A SESSION

  session_id   GET/SET CURRENT SESSION ID

  session_start   INITIALIZE SESSION DATA

FORMS (PHP and HTML)

- ### DEFINITION

    - WAY OF COLLECTING DATA ONLINE FROM USER ACCESSING A WEB SITE

- ### FORM ELEMENTS

    - AUTOMATICALLY AVAILABLE TO PHP SCRIPTS

    - DOTS AND SPACES IN VARIABLE NAMES CONVERTED TO UNDERSCORES

        - Ex: FORM FIELD `"foo.x"` BECOMES `$_GET["foo_x"]` OR `$_POST["foo_x"]`

    - FORM DATA CAN BE MADE INTO AN ARRAY USING THE FOLLOWING SYNTAX

        - `<input name="FormArray[]" />`

        - GROUP ELEMENTS BY ASSIGNING THE SAME ARRAY NAME TO DIFFERENT ELEMENTS; CAN SPECIFY KEYS

- ### SUPERGLOBAL ARRAYS

    - `$_POST` SUPERGLOBAL CONTAINS ALL POST DATA; PAIRED WITH `POST` METHOD

    - `$_GET` SUPERGLOBAL CONTAINS ALL GET DATA

    - `$_REQUEST` IS INDEPENDENT OF DATA SOURCE, AND MERGES INFORMATION FROM `GET`, `POST`, AND `COOKIES`

        - ALTERNATIVE: `IMPORT_request_variables()` IMPORTS `GET`/`POST`/`COOKIE` DATA

- **ENCODING / DECODING**
  - IMPLEMENT AT KEY STAGES IN FORM SUBMISSION PROCESS
    - HTML INTERPRETATION: `htmlspecialchars()` FUNCTION ENCODES SPECIAL CHARACTERS IN DATA, AS A SECURITY MEASURE
    - URL: ENCODE DATA WITH `urlencode()` TO INTERPRET AS ONE ITEM

- **FILE UPLOADING**
  - `POST` METHOD ALLOWS FOR BOTH TEXT AND BINARY FILE UPLOAD
    - USED IN CONJUNCTION WITH AUTHENTICATION AND FILE FUNCTIONS
    - FORM MUST CONTAIN ATTRIBUTE `enctype='multipart/form-data` FOR UPLOADS TO WORK
  - GLOBAL `$_FILES` ARRAY/S WILL CONTAIN ALL UPLOADED FILE INFORMATION
    `$_FILES['filename'][....]`

`['name']`	CLIENT-SIDE FILE NAME
`['type']`	MIME TYPE
`['size']`	FILE SIZE
`['error']`	ERROR CODE FOR UPLOAD
`['tmp_name']`	TEMPORARY FILENAME OF FILE IN WHICH THE UPLOADED FILE WAS STORED ON THE SERVER

COOKIES

- **DEFINITION**

  - WAY OF STORING DATA IN A BROWSER TO ID / TRACK A USER

- **USING COOKIES**

  - CREATE (SET) COOKIES WITH THE `setcookie()` OR `setrawcookie()` FUNCTION

    - MUST BE CALLED BEFORE SENDING ANY OUTPUT TO BROWSER

    - CAN DELAY SCRIPT OUTPUT USING OUTPUT BUFFERING, TO ALLOW TIME TO DECIDE TO SET COOKIES OR SEND HEADERS

  - `setcookie()` PARAMS ARE DEFINED ACCORDING TO SPECIFICATIONS:

NAME=*VALUE*	STRING
expire=*DATE*	OPTIONAL; DEFAULT IS SESSION END
domain=*DOMAIN_NAME*	CHECK ON DOMAIN ATTRIBUTES OF COOKIES AGAINST HOST INTERNET DOMAIN NAME
path=*PATH*	SPECIFIES URLS IN A DOMAIN FOR WHICH COOKIE IS VALID
secure	COOKIE ONLY TRANSMITTED VIA SECURE CHANNELS (HTTPS); BOOLEAN
http_only	COOKIE ONLY MADE ACCESSIBLE VIA HTTP PROTOCOL; BOOLEAN

  - ACCESS WITH `$_COOKIE` OR `$_REQUEST` SUPERGLOBALS

    - COOKIE DATA FROM THE CLIENT IS AUTOMATICALLY SENT TO `$_COOKIE`, IF PARAMS OF `variables_order()` INCLUDE "C" (ENVIRONMENT/GET/POST/Cookie/SERVER)

    - WILL OVERWRITE ITSELF IF NAME, PATH, AND DOMAIN ARE IDENTICAL

- **USING COOKIES (CONTINUED)**

    - COOKIES ARE PART OF THE HTTP HEADER

    - AS WITH SESSIONS, MULTIPLE VALUES CAN BE ASSIGNED TO AN ARRAY

        - TO ASSIGN ALL VALUES TO ONLY ONE COOKIE, CAN USE `serialize()` OR `explode()` WITH FIRST VALUE

- **HTTP HEADERS AND CODE**

    `header()` SETS AN HTTP HEADER

    PARAMS: LOCATION, OVERWRITE, STATUS CODE

    ```
    header('Location: http://www.zend.com/',
     false,default: 200');
    ```

    `headers_list()` LIST OF HEADERS SENT OR TO BE SENT; INDEXED ARRAY

    `headers_sent()` BOOLEAN CONFIRMATION OF WHETHER HEADERS SENT OR NOT

    `header_remove()` REMOVES PREVIOUSLY SET HEADER

    - HTTP HEADERS: INCLUDES A SET OF METHODS... EXAMPLES:

    `isError` BOOLEAN; CHECK IF STATUS CODE IS AN ERROR (CLIENT 4XX OR SERVER 5XX)

    `isSuccessful` BOOLEAN; CHECKS IF STATUS CODE IS SUCCESSFUL (2XX)

    `setHeader` BOOLEAN; DEFAULT VALUE FOR "LAST-MODIFIED" HEADER IS CURRENT DATA / TIME

    OTHER HTTP HEADER CODES:

    1XX   INFORMATIONAL

    3XX   REDIRECTION

- **HTTP AUTHENTICATION**

  - SPECIFIC HOOKS ONLY AVAILABLE WHEN RUNNING THE APACHE MODULE

  - CAN USE `header()` FUNCTION TO SEND A MESSAGE TO THE CLIENT BROWSER TO CAUSE A USERNAME + PASSWORD WINDOW TO DISPLAY

  - UPON ENTRY, A PHP SCRIPT RUNS WITH SET VARIABLES IN THE `$_SERVER` ARRAY

`PHP_AUTH_USER`	USER
`PHP_AUTH_PW`	PASSWORD
`AUTH_TYPE`	AUTHENTICATION TYPE

  - `PHP_AUTH` VARIABLES ARE NOT SET IF EXTERNAL AUTHENTICATION IS ENABLED FOR A PAGE, AND SAFE MODE IN GENERAL, FOR PASSWORD PROTECTION

  - **BASIC ACCESS AUTHENTICATION SCHEME :**

    - ORIGINALLY REFERRED TO AS HTTP/1.0 , NOW 1.1

    - PRESCRIPTS:

      USERNAME APPENDED WITH COLON ":" BEFORE TRANSMISSION

      STRING IS THEN BASE64 ENCODED (TO DEAL WITH NON-HTTP COMPATIBLE CHARACTERS)

REFERENCE:

http://us.php.net/manual/en/intro.session.php

http://php.net/manual/en/tutorial.forms.php

http://www.php.net/manual/en/language.variables.external.php

TEST YOUR KNOWLEDGE : QUESTIONS

1

What is the default timeout of a PHP session cookie?

> A: Depends on the web server
>
> B: 10 minutes
>
> C: 20 minutes
>
> D: Until the browser is closed

2

If a form's action attribute is set to an empty string, where is data usually sent to?

> A: /
>
> B: the current URI
>
> C: index.php
>
> D: the default page of the current directory

3

Which HTTP method is commonly used for file uploads?

> A: CONNECT
>
> B: GET
>
> C: OPTIONS
>
> D: POST

4

How many HTTP requests are required to determine, without JavaScript, whether a client supports cookies or not?

A: 0

B: 1

C: 2

D: Impossible to achieve without JavaScript

5

Which class of HTTP status codes is used for error conditions?

A: 1XX

B: 3XX

C: 5XX

6

Which encryption method is used when using HTTP Basic authentication?

A: None

B: Hashing

C: Asymmetric-key encryption

D: Symmetric-key encryption

1 D: Until the browser is closed

2 B: The current URI

3 D: POST

4 C: 2

5 C: 5XX

6 A: None