EXPERIMENT - 9

WHAT IS INTROSPECTION

- IN (PHP) INTROSPECTION IS THE ABILITY TO EXAMINE AN OBJECT'S CHARACTERISTICS SUCH AS ITS NAME, PARENT CLASS, PROPERTIES, METHODS, INTERFACES

FUNCTIONS OF INTROSPECTION

	FUNCTION NAME	DESCRIPTION
(1)	class_exist ()	- CHECKS WHETHER A CLASS HAS BEEN DEFINED OR NOT - RETURNS A BOOLEAN VALUE RESPECTIVELY
(2)	get_class ()	- RETURNS THE NAME OF THE CLASS OF AN OBJECT - RETURNS A STRING
(3)	get_parent_class()	- RETURNS THE PARENT CLASS FOR AN OBJECT OR CLASS - RETURNS STRING OR FALSE

(4)	is_subclass_of()	- CHECKS IF OBJECT HAS THIS CLASS AS ONE OF ITS PARENTS OR IMPLEMENTS IT - RETURNS A BOOLEAN
(5)	get_declared_classes()	- RETURNS AN ARRAY WITH THE NAME OF ALL DEFINED CLASSES
(6)	get_class_methods()	- RETURNS AN ARRAY WITH THE NAME OF ALL METHODS IN A CLASS
(7)	get_class_vars()	- RETURNS THE PROPERTIES OF A CLASS IN AN ARRAY
(8)	interface_exists()	- CHECKS IF AN INTERFACE HAS BEEN DEFINED
(9)	method_exists()	- CHECKS IF A CLASS METHOD EXISTS OR NOT

EXPLAIN SERIALIZATION

- SERIALIZING MEANS CONVERTING AN OBJET INTO BYTESTREAM REPRESENTATION THAT CAN BE STORED IN A FILE.

- IT IS A TECHNIQUE USED BY PROGRAMMERS TO PRESERVE THEIR DATA INTO A FORMAT THAT CAN RESTORED LATER.

METHODS

① serialize ()

- GENERATES A STORABLE REPRESENTATION OF A VALUE

SYN function serialize (

mixed $value

): string

value → VALUE TO BE SERIALIZED

- RETURNS A STRING CONTAINING A BYTE STREAM REPRESENTING VALUE.

(2) unserialize ()

— CREATES A (PHP) VALUE FROM A STORED REPRESENTATION

SYN function unserialize (

string $data,
array $options = []

): mixed

data → SERIALIZED STRING
options → OPTIONAL, ASSIGN ARRAY TYPE

— RETURNS A MIXED VALUE

PRODUCT	PROCESS	TOTAL	SIGN

EXPERIMENT - 10

[HTML CONTROLS]

(a) Text Input

— USED FOR SINGLE LINE TEXT INPUTS

```
<input
        type = "text"
        name = "name"
        placeholder = "Enter name here"
>
```

input — TAG
type — INPUT TYPE - (text)
name — ASSIGN A NAME
placeholder — PLACEHOLDER TEXT

(b)

```
< input
        type = "password"
        name = "password"
        placeholder = "Enter password"
>
```

input — TAG
type — INPUT TYPE — (password)
name — ASSIGN A NAME
placeholder — PLACEHOLDER TEXT

(c)

```
< input
        type = "email"
```

name = "email"

placeholder = "Enter mail"

>

input — TAG
type — INPUT TYPE - (email)
name — ASSIGN A NAME
placeholder — PLACEHOLDER TEXT

(d)

<input

type = "date"

name = "dob"

>

input — TAG
type — INPUT TYPE - (date)
name — ASSIGN A NAME

(e)

<button

type = "submit"

> SUBMIT </button>

type — SUBMIT THE FORM WHEN CLICKED

(f)

<input type="checkbox" name = "game[]" value = "GTA">
<label for = "GTA" > GTA5 </label>
<input type = "checkbox" name = "game[]" value = "RDR">
<label for = "RDR"> RDR 2 </label>

input — TAG
type — INPUT TYPE — (checkbox)
name — games[] — STORE MULTIPLE VALUES
value — VALUE TO SUBMIT
label — TAG

(9)
```
<input type="radio" name="gender" value="Male">
<label for="male">Male</label>
<input type="radio" name="gender" value="Dishwasher">
<label for="dishwasher">Dishwasher</label>
```

INPUT — TAG
label — TAG
type — INPUT TYPE — (radio)
name — ASSIGN NAME
value — VALUE TO SUBMIT

PROCESS	PRODUCT	TOTAL	SIGN

EXPERIMENT – 11

HTML CONTROLS

(a)

```
<select name = "country" multiple>
    <option value = "US"> USA </option>
    <option value = "uk"> UK </option>
</select>
```

select — DEFINE DROPDOWN LIST
name — ASSIGN NAME
multiple — (LISTBox)
option — LIST ITEM
value — VALUE TO SUBMIT

(b)

```
<input type = "text"
list = "cities"
name = "city"
>

<datalist
    id = "cities"
>
    <option value = "NY">
    New York

    <option value = "New York">
```

```
<option value="Florida">
<option value="California">
</datalist>
```

list — ASSOCIATE INPUT WITH "datalist"
datalist — DEFINE A LIST
option — LIST ITEMS
value — VALUE TO SUBMIT

(c)
```
<button type="submit">
    SUBMIT
</button>
```

button — TAG
type — SUBMIT FORM WHEN CLICKED

PRODUCT	PROCESS	TOTAL	SIGN

EXPERIMENT - 12

WHAT IS REGEX

- A REGEX IS AN OBJECT THAT DESCRIBES A PATTERN OF CHARACTERS.

- THEY ARE USED TO PERFORM PATTERN MATCHING

MODIFIERS

1. g
   - PERFORM GLOBAL MATCH, FIND ALL MATCHES
   CASE - SENSITIVE
2. i
   - CASE INSENSITIVE, STOP AT FIRST MATCH
3. m
   - DEFAULT, CASE-SENSITIVE MATCHING IS DONE ON A SINGLE LINE.
   PERFORMS MULTI-LINE MATCHING

BRACKETS

1. [abc]

— FIND ANY CHARACTER FROM THE BRACKETS

② [^abc]

— FIND ANY CHARACTER NOT FROM THE BRACKETS

③ [0-9]

— FIND ANY CHARACTER [ANY DIGIT]

④ (x|y)

— FIND EITHER CHARACTER

META CHARACTERS

① .

— FIND ANY SINGLE CHARACTER, EXCEPT NEWLINE OR "10"

② \w

— FIND A WORD CHARACTER

③ \W

— FIND A NON WORD CHARACTER

(4) \d

— FIND A DIGIT

(5) \D

— FIND A NON DIGIT

(6) \s

— FIND A WHITESPACE CHARACTER

(7) \S

— FIND A NON WHITE SPACE CHARACTER

(8) \b

— FIND MATCH AT START OR END

[QUANTIFIERS]

(1) n +

— MATCHES ANY STRING THAT CONTAINS ATLEAST ONE OR MORE

(2) n *

— MATCHES ANY STRING THAT CONTAINS ZERO OR MORE

(3) n?

— MATCHES ANY STRING THAT CONTAINS ZERO OR ONE

(4) n{X}

— MATCHES ANY STRING THAT CONTAINS A SEQUENCE OF X 'n's.

(5) n{X,Y}

— MATCHES ANY STRING THAT CONTAINS A SEQUENCE OF X TO Y n's.

(6) n{X,}

— MATCHES ANY STRING THAT CONTAINS A SEQUENCE OF AT LEAST X n's.

(7) n$

— MATCHES ANY STRING WITH "n" AT THE END OF IT

(8) ^n

- MATCHES ANY STRING WITH "n" AT THE BEGINNING OF IT

[METHODS]

① preg_match()

- RETURNS "1" IF PATTERN IS FOUND OR "0"

pattern — REGEX
input — STRING IN WICH TO SEARCH
matches — OPTIONALS, ARRAYS WHERE FOUND
flags — OPTIONALS, OPTIONS FOR THE ARRAY
offset — OPTIONAL, TILL WHERE TO SEARCH IN THE STRING

② preg_match_all()

- RETURNS NUMBER OF TIMES THE PATTERN FOUND IN STRING, MAYBE "0"

PRODUCT	PROCESS	TOTAL	SIGN

EXPERIMENT – 13

WHAT IS COOKIE

- A COOKIE IS A SMALL FILE THAT THE SERVER EMBED'S IN THE USER'S PC.

- A COOKIE IS SENT EVERY TIME THE SAME PC REQUESTS A PAGE WITH A BROWSER.

ATTRIBUTES

(1) NAME & VALUE —
NAME — REQUIRED, COOKIE NAME
VALUE — OPTIONAL, COOKIE VALUE

(2) SECURE — OPTIONAL, IF TRUE THEN COOKIE IS SET WHEN THERE IS AN SECURE HTTP CONNECTION, DEFAULT FALSE

(3) DOMAIN — OPTIONAL, IF SPECIFIED THEN HOST NAME OF SERVER IS USED AS THE DEFAULT VALUE

(4) PATH — OPTIONAL, SPECIFY THE URL OR PATH FOR WHICH THE COOKIE IS VALID. DEFAULT IS THE CURRENT FOLDER

(5) HTTPONLY — OPTIONAL, IF SET TO TRUE, THEN COOKIE IS ACCESSIBLE VIA HTTP PROTOCOL

⑥ EXPIRES — WHEN TO KILL THE COOKIE, OPTIONAL, DEFAULT IS "0".

COOKIE METHODS

① setCookie (name, value, expire, path, domain, secure, httponly)

— USED TO CREATE A COOKIE

② TO MODIFY THE COOKIE, JUST SET THE COOKIE AGAIN USING "setCookie()"

— THE VALUE IS AUTOMATICALLY ENCODED WHEN SENDING THE COOKIE.

③ TO DELETE A COOKIE, USE "setCookie()" WITH AN EXPERATION DATE IN THE PAST.

WHAT IS SESSION

— SESSION IS A WAY TO STORE INFO AND SHARE IT ACROSS DIFF. WEB PAGES.

— THE SESSION IS NOT STORED ON USER'S PC

— HOLD INFO. OF ONE USER AND CAN BE ACCESSED FROM ANY WHERE

- THESE LAST TILL THE USER CLOSES THE BROWSER.

[METHODS]

① session_start()

— USED TO START A SESSION

— SET WITH GLOBAL VARIABLE $_SESSION

— "session_start()" MUST BE THE FIRST THING IN YOUR DOCUMENT

② session_unset()

— USED TO REMOVE ALL SESSION VARIABLES.

③ session_destroy()

— USED TO DESTROY session

PRODUCT	PROCESS	TOTAL	SIGN

EXPERIMENT - 14

MAIL METHOD

mail ()

- ALLOWS USER TO SEND EMAILS DIRECTLY FROM SCRIPT

SYNTAX

```
function mail (
        string $to ,
        string $subject ,
        string $message ,
        array|string $add_headers = [] ,
        string $add_parms = ''
): bool
```

to → THE RECIPIENTS EMAIL ADDRESS

subject → SUBJECT OF THE EMAIL TO BE SENT

message → Message TO BE SENT

add_headers → STRING OR ARRAY TO BE INSERTED FROM THE END OF THE EMAIL. IT IS SET IN THE "php.ini" FILE

add_params → USED TO PASS ADDITIONAL PARAMETERS THAT CAN BE USED TO PASS ADDITIONAL FLAGS AS THE COMMAND LINE OPTIONS DEFINED BY "sendmail_path"

→ RETURNS A BOOLEAN VALUE

Eg.

```
$sub = "My mail";
$msg = "Hello World";
$rec = "abc@example.com";

if (mail ($rec, $sub, $msg)) {
    echo "Sent";
}
else
    echo "Failed";
}
```

PRODUCT	PROCESS	TOTAL	SIGN