Curs TWSS

Funcții

- Set de instrucțiuni
- Returnează o valoare (opțional)
- Are nume unic
- Poate avea parametri de intrare
- Există două tipuri de funcții:
    - Built-in
    - Definite de utilizatori
- func(arg1, arg2, ...)

Avantaje:

- Reutilizarea codului
- Reducerea erorilor de sintaxă și programare
- Scăderea timpului de încărcare a paginii
- Scăderea timpului de execuție a paginii
- Cod generalizat datorită argumentelor funcțiilor

Funcții built-in

- Print: *print("print is a function");*
- Afișare informații despre versiunea de PHP: *phpinfo();*

<?php
echo strrev(" .dlrow olleH");
// Reverse string
echo str_repeat("Hip ", 2);
// Repeat string
echo strtoupper("hooray!");
// String to uppercase
?>
	Hello world.
	
	Hip Hip
	
	HOORAY!

Sintagma funcției

```
function function_name([parameter [, ...]])
{
// Statements
}
```

- Numele de funcții sunt case-insensitive: PRINT, Print, PrInT
- Funcția se evaluează prima dată

```
$lowered = strtolower("aNY # of Letters and Punctuation you WANT");
echo $lowered; // any # of letters and punctuation you want
$ucfixed = ucfirst("any # of letters and punctuation you want");
echo $ucfixed; //Any # of letters and punctuation you want
```

```
print(5-8);
print(abs(5-8));
ucfirst(strtolower("aNY # of Letters and Punctuation you WANT"))
```

```
<?php
echo fix_names("WILLIAM", "henry", "gatES");
function fix_names($n1, $n2, $n3)
{
$n1 = ucfirst(strtolower($n1));
$n2 = ucfirst(strtolower($n2));
$n3 = ucfirst(strtolower($n3));
return $n1 . " " . $n2 . " " . $n3;
}
?>
// William Henry Gates
```

```
<?php
$names = fix_names("WILLIAM", "henry", "gatES");
echo $names[0] . " " . $names[1] . " " . $names[2];
function fix_names($n1, $n2, $n3)
{
$n1 = ucfirst(strtolower($n1));
$n2 = ucfirst(strtolower($n2));
$n3 = ucfirst(strtolower($n3));
return array($n1, $n2, $n3);
}
?>
```

Funcții – valori implicite pentru parametri

Setarea valorii implicite

```
<?php
echo "<h3>Use of Default Parameter<br></h3>";
function increment($num, $increment = 1)
{
$num += $increment;
echo $num;
}
$num = 4;
increment($num);
//5
increment($num, 3);
//7
?>
```

Apelul dinamic al funcțiilor

- Numele funcției se poate atribui unei variabile
- Variabila se poate trata în continuare ca o funcție

```
<?php
function sayHello()
{
print "hello<br>";
}
$function_holder = "sayHello";
$function_holder();
//hello
?>
```

Transmiterea prin referință

- Simbolul &

```
<?php
$a1 = "WILLIAM";
$a2 = "henry";
$a3 = "gatES";
echo $a1 . " " . $a2 . " " . $a3 . "<br />";
fix_names($a1, $a2, $a3);
echo $a1 . " " . $a2 . " " . $a3;
function fix_names(&$n1, &$n2, &$n3)
{
$n1 = ucfirst(strtolower($n1));
$n2 = ucfirst(strtolower($n2));
$n3 = ucfirst(strtolower($n3));
}
?>
```

Scopul variabilelor

- Variabilele definite în afara funcțiilor se numesc variabile globale

```
<?php
function myfunction()
{
$no1=10; //local variable
}
$no1=20; // global variable
myfunction();
echo $no1;
//20
?>
```

- Pentru a accesa sau schimba variabilele globale în interiorul funcției se folosește vectorul $GLOBALS[] (șir asociativ în care numele variabilelor sunt cheile șirului)

```php
<?php
function myfunction()
{
$GLOBALS[“no1”]=10; //local variable
}
$no1=20; // global variable
myfunction();
echo $no1;
//10
?>
```

```php
<?php
function myfunction()
{
$no1=10; $no2=20;
echo $no1+$no2."<br>";
echo $no1+$GLOBALS["no2"];
}
$no2=40;
add();
//30, 50
?>
```

- Folosirea cuvântului cheie **global** permite importarea variabilelor globale în scopul funcției

```php
<?php
function myfunction()
{
global $no1; $no1=10;
}
$no1=20;
myfunction();
echo $no1;
//10
?>
```

Funcții – returnarea variabilelor globale

```php
<?php
$a1 = "WILLIAM";
$a2 = "henry";
$a3 = "gatES";
echo $a1 . " " . $a2 . " " . $a3 . "<br />";
fix_names();
echo $a1 . " " . $a2 . " " . $a3;
function fix_names()
{
global $a1; $a1 = ucfirst(strtolower($a1));
global $a2; $a2 = ucfirst(strtolower($a2));
global $a3; $a3 = ucfirst(strtolower($a3));
```

```
}
?>
```

- Variabilele statice pot fi inițializate, această inițializare având loc prima dată când declarația statică a fost întâlnită

```
<?php
function display()
{
Static $no=10;
echo $no;
$no++;
}
display();
display();
// 10, 11
?>
```

Includerea de fișiere

- INCLUDE

```
<?php
include "library.php";
// Your code goes here
?>
```

```
<?php
include_once "library.php";
// Your code goes here
?>
```

- Include inserează codul din fișierul specificat ori de câte ori folosim funcția.
- Dacă folosim funcția include_once, prima dată verifică dacă deja a fost inserată codul, și dacă nu, numai atunci îl inserează

- REQUIRE

```
<?php
require "library.php";
// Your code goes here
?>
```

```
<?php
require_once "library.php";
// Your code goes here
?>
```

- Funcția include dacă nu reușește să insereze codul din fișier, merge în continuare. Dacă avem neapărat nevoie, ca fișierul să fie inserat, folosim funcția require sau require_once, care în cazul că nu are succes, se oprește și afișează o eroare.

Funcții – compatibilitate

- PHP-ul este un open source, și se dezvoltă continuu. În cazul în care vrem să știm dacă o funcție există în versiunea php de pe server, putem folosi funcția function_exists()

```
<?php
if (function_exists("array_combine"))
{
echo "Function exists";
}
else
{
echo "Function does not exist - better write our own";
}
?>
```

PHP – Obiecte

Avantaje:

- Reutilizarea codului
- Reducerea erorilor de sintaxă și programare
- Scăderea timpului de încărcare a paginii
- Scăderea timpului de execuție a paginii
- Cod generalizat datorită argumentelor funcțiilor
- Întreținerea facilă
- Abstractizare

Obiecte:

- Încorporează una sau mai multe funcții într-o singură structură numită clasă

Obiecte – teminologie

- Manipularea proprietăților se face doar prin intermediul metodelor
- Metode = interfața obiectelor
- Moșteniri

Obiecte – declarare clasă

- Cuvânt cheie: **class**

<?php
$object = new User;
print_r($object);
class User
{
public $name, $password;
function save_user()
{
echo "Save User code goes here";
}
}
?>
	User Object
	(
	[name] =>
	[password] =>
	)

Obiecte – creare

- Cuvânt cheie: **new**

$object = new User; $temp = new User('name', 'password');	User Object ([name] => ‘name’ [password] => ‘password’)

Obiecte – accesare

```
<?php
$object = new User;
print_r($object); echo "<br />";
$object->name = "Joe";
$object->password = "mypass";
print_r($object); echo "<br />";
$object->save_user();
class User
{
public $name, $password;
function save_user()
{
echo "Save User code goes here";
}
}
?>
```

```
User Object
(
[name] =>
[password] =>
)
User Object
(
[name] => Joe
[password] => mypass
)
Save User code goes here
```

Obiecte - clonare

```
<?php
$object1 = new User();
$object1->name = "Alice";
$object2 = $object1;
$object2->name = "Amy";
echo "object1 name = " . $object1->name .
"<br />";
echo "object2 name = " . $object2->name;
class User
{
public $name;
}
?>
```

```
object1 name = Amy
object2 name = Amy
```

În caz de copiere, dacă modificăm proprietățile obiectului nou, se modifică și cel original. În cayul când nu vrem să se întâmple acest lucru, folosim clonarea:

```
<?php
$object1 = new User();
$object1->name = "Alice";
$object2 = clone $object1;
$object2->name = "Amy";
echo "object1 name = " . $object1->name .
"<br />";
echo "object2 name = " . $object2->name;
class User
{
public $name;
} ?>
```

```
object1 name = Alice
object2 name = Amy
```

Obiecte - constructori

```php
<?php
class User
{
function User($param1, $param2)
{
// Constructor statements go here
}
}
?>
```

```php
<?php
class User
{
function __construct($param1,
$param2)
{
// Constructor statements go here
}
}
?>
```

Obiecte – destructori

```php
<?php
class User
{
function __destruct()
{
// Destructor code goes here
}
}
?>
```

Obiecte – metode

- Declararea metodei este similara declarării funcției
- Diferențe:
  - Acces la variabila **$this** care poate fi utilizată pentru a accesa proprietățile obiectului curent

```php
<?php
class User
{
public $name, $password;
function get_password()
{
return $this->password;
}
}
?>
```

```php
$object = new User;
$object->password = "secret";
echo $object->get_password();
```

Obiecte – metode statice

```
<?php
User::pwd_string();
class User
{
static function pwd_string()
{
echo "Please enter your password";
}
}
?>
```

Obiecte – declararea proprietăților

Definirea implicită a proprietăților

```
$object1 = new User();
$object1->name = "Alice";
echo $object1->name;
class User {}
?>
```

Declararea proprietăților

```
class Test
{
public $name = "Paul Smith"; // Valid
public $age = 42; // Valid
public $time = time(); // Invalid - calls a function
public $score = $level * 2; // Invalid - uses an expression
}
```

Obiecte – declararea constantelor

Declararea constantelor în interiorul unei clase

```
<?php
Translate::lookup();
class Translate
{
const ENGLISH = 0;
const SPANISH = 1;
const FRENCH = 2;
const GERMAN = 3;
// ...
function lookup()
{
echo self::SPANISH;
}
}
?>
```

Obiecte - Scopul metodelor și proprietăților

- **Public** – folosirea cuvintelor cheie **var** sau **public** sau la declararea implicită a variabilelor, metolor sau proprietăților
- **Protected** – aceste proprietăți sau metode pot fi referențiate doar de către metodele definite în aceeași clasă sau în subclase
- **Private** – aceste proprietăți sau metode pot fi referențiate doar de către metodele definite în aceeași clasă (nu și în subclase)

Schimbarea scopului metodelor și proprietăților

```
<?php
class Example
{
var $name = "Michael"; // Same as public but deprecated
public $age = 23; // Public property
protected $usercount; // Protected property
private function admin() // Private method
{
// Admin code goes here
}
}
?>
```

Obiecte - Metode și proprietăți statice

Definirea unei clase cu proprietăți statice

```
<?php
$temp = new Test();
echo "Test A: " . Test::$static_property . "<br />";
echo "Test B: " . $temp->get_sp() . "<br />";
echo "Test C: " . $temp->static_property . "<br />";
class Test
{
static $static_property = "I'm static";
function get_sp()
{
return self::$static_property;
}
}
?>
```

```
Test A: I'm static
Test B: I'm static
Notice: Undefined property: Test::$static_property
Test C:
```

Obiecte – moșteniri

- O nouă clasă poate fi derivată dintr-o clasă deja definită
- Avantaj: reutilizabilitatea codului

Relația de moștenire și extinderea unei clase

```
<?php
$object = new Subscriber;
$object->name = "Fred";
$object->password = "pword";
$object->phone = "012 345 6789";
$object->email = "fred@blog.com";
$object->display();
class User
{
public $name, $password;
function save_user()
{
echo "Save User code goes here";
}
}
```

```
class Subscriber extends User
{
public $phone, $email;
function display()
{
echo "Name: " . $this->name . "<br />";
echo "Pass: " . $this->password . "<br />";
echo "Phone: " . $this->phone . "<br />";
echo "Email: " . $this->email;
}
}
?>
```

```
Name: Fred
Pass: pword
Phone: 012 345 6789
Email: fred@blog.com
```

Obiecte – operatorul părinte

Suprascrierea unei metode și folosirea operatorului "părinte"

```
<?php
$object = new Son;
$object->test();
$object->test2();
class Dad
{
function test()
{
echo "[Class Dad] <br />";
}
}
```

```
class Son extends Dad
{
function test()
{
echo "[Class Son] <br />";
}
function test2()
{
parent::test();
}
}
?>
```

```
[Class Son]
[Class Dad]
```

```
self::method();
parent::method();
```

Obiecte - constructor în subclasă

Apelul constructorului din clasa părinte

```
<?php
$object = new Tiger();
echo "Tigers have...<br>";
echo "Fur: " . $object->fur . "<br />";
echo "Stripes: " . $object->stripes;
class Wildcat
{
public $fur; // Wildcats have fur
function __construct()
{
$this->fur = "TRUE";
}
}
```

```
class Tiger extends Wildcat
{
public $stripes;
// Tigers have stripes
function __construct()
{
parent::__construct(); // Call parent
constructor first
$this->stripes = "TRUE";
}
}
?>
```

```
Tigers have...
Fur: TRUE
Stripes: TRUE
```

Obiecte – metode finale

- Folosirea cuvântului cheie **final**

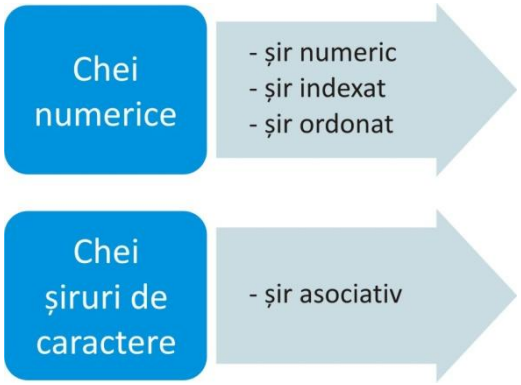
Crearea metodei finale

```
<?php
class User
{
final function copyright()
{
echo "This class was written by Joe Smith";
}
}
?>
```

Șiruri în PHP

- Șir – listă care conține mai multe elemente (valori)
- Permite stocarea, ordonarea și accesarea unor mai multe valori folosind un singur nume
- Un șir se formează din elemente de forma (cheie, valoare)
- Fiecare element al șirului se poate accesa direct prin intermediul index-ului
- Index-ul unui element din șir poate avea valoare numerică sau poate fi șir de caractere
- În mod implicit, elementele din șir sunt indexate numeric începând de la 0

- Cheia unui element – scalar (string, număr, boolean)
- Valoarea unui element – scalar (string, număr, boolean) sau non-scalar (șir sau alte elemente)



Șiruri indexate numeric

<?php $paper[] = "Copier"; $paper[] = "Inkjet"; $paper[] = "Laser"; $paper[] = "Photo"; print_r($paper); ?>	Array ([0] => Copier [1] => Inkjet [2] => Laser [3] => Photo)
<?php $paper[0] = "Copier"; $paper[1] = "Inkjet"; $paper[2] = "Laser"; $paper[3] = "Photo"; print_r($paper); ?>	Array ([0] => Copier [1] => Inkjet [2] => Laser [3] => Photo)
<?php $paper[] = "Copier"; $paper[] = "Inkjet"; $paper[] = "Laser"; $paper[] = "Photo"; for ($j = 0 ; $j < 4 ; ++$j) echo "$j: $paper[$j] "; ?>	0: Copier 1: Inkjet 2: Laser 3: Photo

Şiruri indexate prin nume

```
<?php
$paper['copier'] = "Copier & Multipurpose";
$paper['inkjet'] = "Inkjet Printer";
$paper['laser'] = "Laser Printer";
$paper['photo'] = "Photographic Paper";
echo $paper['laser'];
?>
```

Şiruri - adăugare elemente

Adăugarea elementelor într-un şir folosind cuvântul cheie array	
<?php $p1 = array("Copier", "Inkjet", "Laser", "Photo"); echo "p1 element: " . $p1[2] . " "; $p2 = array('copier' => "Copier & Multipurpose", 'inkjet' => "Inkjet Printer", 'laser' => "Laser Printer", 'photo' => "Photographic Paper"); echo "p2 element: " . $p2['inkjet'] . " "; ?>	p1 element: Laser p2 element: Inkjet Printer
echo $p1['inkjet']; echo $p2['3'];	// Undefined index // Undefined offset

Şiruri – iniţializarea unui şir

```
<?php
$scores = array(5, 10);
$padded = array_pad($scores, 5, 0);
// $padded is now array(5, 10, 0, 0, 0)
?>
```

```
<?php
$scores = array(5, 10);
$padded = array_pad($scores, -5, 0);
// $padded is now array(0, 0, 0, 5, 10)
?>
```

Șiruri - Parcurgerea șirului de caractere

```
$paper = array("Copier", "Inkjet", "Laser", "Photo");
$j = 0;
foreach ($paper as $item)
{
echo "$j: $item<br>";
++$j;
}
?>
```

```
<?php
$paper = array('copier' => "Copier & Multipurpose",
'inkjet' => "Inkjet Printer",
'laser' => "Laser Printer",
'photo' => "Photographic Paper");
foreach ($paper as $item => $description)
echo "$item: $description<br>";
?>
```

```
<?php
$paper = array('copier' => "Copier & Multipurpose",
'inkjet' => "Inkjet Printer",
'laser' => "Laser Printer",
'photo' => "Photographic Paper");
while (list($item, $description) = each($paper))
echo "$item: $description<br>";
?>
```

Șiruri - Parcurgerea șirurilor multidimensionale

```
<?php
$products = array(
 'paper' => array(
    'copier' => "Copier & Multipurpose",
    'inkjet' => "Inkjet Printer",
    'laser' => "Laser Printer",
    'photo' => "Photographic Paper"),
 'pens' => array(
    'ball' => "Ball Point",
    'hilite' => "Highlighters",
    'marker' => "Markers"),
 'misc' => array(
    'tape' => "Sticky Tape",
    'glue' => "Adhesives",
    'clips' => "Paperclips") );
echo "<pre>";
foreach ($products as $section => $items)
 foreach ($items as $key => $value)
    echo "$section:\t$key\t($value)<br>";
echo "</pre>"; ?>
```

```
echo $products['misc']['glue']; //Adhesives
```

Șiruri – funcții

is_array()	echo (is_array($fred)) ? "Is an array" : "Is not an array";

count()	echo count($fred);
	echo count($fred, 1); // multi-dimensional array
	// al doilea parametru 0 = se numără elementele de pe nivelul 0,
	// al doilea parametru 1 = se numără recursiv elementele de pe toate nivelele

sort()
rsort()
	- **afectează șirul de caractere dat ca parametru;**
	sort($fred);sort($fred, SORT_NUMERIC);sort($fred, SORT_STRING);
	rsort($fred, SORT_NUMERIC);rsort($fred, SORT_STRING);
	- **în cazul array-urilor asociative cheile sunt resetate și indexate numeric**

asort() **arsort()**	- **sortează un array asociativ crescător pe baza valorilor**
ksort() **krsort()**	- **sortează un array asociativ crescător pe baza cheilor**
shuffle()	- **ordine aleatoare pentru elementele șirului;** - **afectează șirul de caractere dat ca parametru;** shuffle($cards);
explode()	- **transformă șirul de caractere dat ca parametru într-un șir de obiecte;** <?php $temp = explode(' ', "This is a sentence with seven words"); print_r($temp); ?>
extract()	extract($_GET); extract($_GET, EXTR_PREFIX_ALL, 'fromget');
compact()	$fname = "Elizabeth"; $sname = "Windsor"; $address = "Buckingham Palace"; $city = "London"; $contact = compact('fname', 'sname', 'address', 'city'); print_r($contact); $j = 23; $temp = "Hello"; $address = "1 Old Street"; $age = 61; print_r (compact (explode (' ','j temp address age')));

reset()	reset($fred); // Throw away return value
	$item = reset($fred); // Keep first element of the array

end()	end($fred); $item = end($fred);
array_unique(), array_reverse(), array_merge(), array_pop(), array_push() **array_search(), array_shift(), array_slice(), array_splice(), array_sum()**	

Șiruri – iterator

Current()	Returnează elementul curent indicat de iterator
Reset()	Iteratorul va indica spre primul element din șir care este și returnat
Next()	Iteratorul va indica spre următorul element din șir care este și returnat
Prev()	Iteratorul va indica spre elementul anterior din șir care este și returnat
End()	Iteratorul va indica spre ultimul element din șir care este și returnat
Each()	current() + next() Se folosește pentru a parcurge șirul element cu element
Key()	Returnează cheia elementului curent indicat de iterator

```
<?php
reset($addresses);
while (list($key, $value) = each($addresses)) {
echo "$key is $value<BR>\n“;
} ?>
```

Șiruri – exemple funcții

Ștergerea unui element dintr-un șir

```
unset($array[‘$key’])
```

Transformarea șirului (array) în șir de caractere

```
<?php
$cars=array("tata","bmw","audi","chev");
$cars1=implode("_",$cars);
echo $cars1;
// tata_bmw_audi_chev
?>
```

Transformarea șirului de caractere în șir(array)

```
<?php
$cars="tata,bmw,audi,chev";
$cars1=explode(",",$cars);
print_r($cars1);
//Array ( [0] => tata [1] => bmw [2] => audi [3] => chev )
?>
```

Extragerea unui subșir dintr-un șir

```
$subset = array_slice(array, offset, length, preserve);
```

array	*
offset	* elementul inițial care se copiază
length	lungimea subșirului
preserve	păstrarea cheilor (true sau false)

```
<?php
$cars=array("hi"=>"tata","hello"=>"bmw","hey"=>"audi","hm"=>"chev");
$cars1=array_slice($cars,1,3);
print_r ($cars1);
```

```
//Array ( [hello] => bmw [hey] => audi [hm] => chev )
?>
```

Funcția printf()

- Funcția printf este variantă la print, cu diferența că se poate formata șirul

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

Expresie	Conversie	Exemplu
%	Afișează caracterul %	%
b	Afișează argumentul în binar	1111011
c	Afișează caracterul ASCII corespunzător argumentului	{
d	Afișează argumentul ca număr întreg cu semn	123
e	Afișează argumentul ca număr real, folosind notația științifică	1.23000e+2
f	Afișează argumentul ca număr real, în virgulă flotantă	123.000000
o	Afișează argumentul în baza 8	173
s	Afișează argumentul ca șir de caractere	123
u	Afișează argumentul în baza 10	123
X, X	Afișează argumentul în baza 16	7b, 7B

Printf – exemple

```
printf("My name is %s. I'm %d years old, which is %X in hexadecimal",'Simon', 33, 33);
//My name is Simon. I'm 33 years old, which is 21 in hexadecimal
```

```
printf("<font color='#%X%X%X'>Hello</font>", 65, 127, 245);
//<font color='#417FF5'>Hello</font>
```

```
printf("<font color='#%X%X%X'>Hello</font>", $r-20, $g-20, $b-20);
```

```
printf("The result is: $%.2f", 123.42 / 12);
//The result is $10.29
```

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

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

Printf – conversie

Start conversie	Caracter de ”umplutură”	Numărul de caractere de umplutură	Numărul total de caractere	Precizia de afișare	Expresia de conversie	Exemple
%		15			f	10.285000
%	0	15		.4	f	000000000010.29
%	#	15		.2	f	########10.2850
%					S	[House]
%	-		10		S	[House]
%		‘#	8	.4	s	[####Hous]

```
<?php
echo "<pre>"; // Enables viewing of the spaces
$h = 'House';
printf("[%s]\n", $h); // Standard string output
printf("[%10s]\n", $h); // Right justify with spaces
printf("[%-10s]\n", $h); // Left justify with spaces
printf("[%010s]\n", $h); // Zero padding
printf("[%'#10s]\n\n", $h); // Use the custom padding character '#‘

$d = 'Doctor House';
printf("[%10.8s]\n", $d); // Right justify, cutoff of 8 characters
printf("[%-10.6s]\n", $d); // Left justify, cutoff of 6 characters
printf("[%-'@10.6s]\n", $d); // Left justify, pad '@', cutoff 6 chars
?>
```

```
[House]
[    House]
[House    ]
[00000House]
[#####House]


[  Doctor H]
[Doctor    ]
[Doctor@@@@]
```

Sprintf

- Este ca și printf, cu diferența, că nu afișează rezultatul pe ecran, ci îl putem salva într-o variabilă.

```
$hexstring = sprintf("%X%X%X", 65, 127, 245);
$out = sprintf("The result is: $%.2f", 123.42 / 12);
echo $out;
```

PHP – Date & Time

- timestamp (numărul de secunde trecute de la 1 Ianuarie 1970)

Date & Time – exemple

```
echo time();
// timestamp curent
```

```
echo time() + 7 * 24 * 60 * 60;
// timestamp peste o saptamana
```

```
echo mktime(0, 0, 0, 1, 1, 2000);
// creare timestamp dintr-o dată
// (ora, minut, secunda, luna, zi, an)
```

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

```
echo date("l F jS, Y - g:ia", time());
\\Thursday April 15th, 2010 - 1:38pm
```

Date & Time – format

Caracter	Descriere	Exemplu
Day		
d	Day of the month, 2 digits with leading zeros	*01* to *31*
D	A textual representation of a day, three letters	*Mon* through *Sun*
j	Day of the month without leading zeros	*1* to *31*
l	A full textual representation of the day of the week	*Sunday* through *Saturday*
N	ISO-8601 numeric representation of the day of the week (added in PHP 5.1.0)	*1* (for Monday) through *7* (for Sunday)
S	English ordinal suffix for the day of the month, 2 characters	*st*, *nd*, *rd* or *th*. Works well with *j*
w	Numeric representation of the day of the week	*0* (for Sunday) through *6* (for Saturday)
z	The day of the year (starting from 0)	*0* through *365*
Week		
W	ISO-8601 week number of year, weeks starting on Monday	Example: *42* (the 42nd week in the year)
Month		
F	A full textual representation of a month, such as January or March	*January* through *December*
m	Numeric representation of a month, with leading zeros	*01* through *12*
M	A short textual representation of a month, three letters	*Jan* through *Dec*
n	Numeric representation of a month, without leading zeros	*1* through *12*
t	Number of days in the given month	*28* through *31*

Year		
L	Whether it's a leap year	*1* if it is a leap year, *0* otherwise.
Y	A full numeric representation of a year, 4 digits	Examples: *1999* or *2003*
y	A two digit representation of a year	Examples: *99* or *03*
Time		
a	Lowercase Ante meridiem and Post meridiem	*am* or *pm*
A	Uppercase Ante meridiem and Post meridiem	*AM* or *PM*
g	12-hour format of an hour without leading zeros	*1* through *12*
G	24-hour format of an hour without leading zeros	*0* through *23*
h	12-hour format of an hour with leading zeros	*01* through *12*
H	24-hour format of an hour with leading zeros	*00* through *23*
i	Minutes with leading zeros	*00* to *59*
s	Seconds, with leading zeros	*00* through *59*

Date & Time – constant

DATE_ATOM	Y-m-d\TH:i:sP	2012-08-16T12:00:00+0000
DATE_COOKIE	l, d-M-y H:i:s T	Thursday, 16 Aug 2012 12:00:00 UTC
DATE_RSS	D, d M Y H:i:s T	Thu, 16 Aug 2012 12:00:00 UTC
DATE_W3C	Y-m-d\TH:i:sP	2012-08-16 T12:00:00+0000

Date & Time – checkdate

```
<?php
$month = 9; // September (only has 30 days)
$day = 31; // 31st
$year = 2012; // 2012
if (checkdate($month, $day, $year))
  echo "Date is valid";
else
  echo "Date is invalid";
?>
```

Lucrul cu fișiere în PHP

- Manipularea fișierelor în PHP
  - Creare
  - Upload
  - Editare
- Erori uzuale
  - Umplerea hard-ului cu fișiere care nu sunt necesare
  - Ștergerea conținutului unui fișier

Lucrul cu fișiere – funcții

file_exists()
- Verifică existența unui fișier

fopen()
- deschide fișier
- creează fișier

fwrite()
- Scrie în fișier

fclose()
- Închide un fișier

Verificare existență fișier

```
if (file_exists("testfile.txt")) echo "File exists";
```

Creare fișier

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

Lucrul cu fișiere – fopen

Mod	Acțiune
`r`	Citește de la începutul fișierului
`r+`	Citește de la începutul fișierului și permite scrierea
`w`	Scrie la începutul fișierului și șterge restul conținutului
`w+`	Scrie la începutul fișierului și permite citirea
`a`	Scrie la sfârșitul fișierului
`a+`	Scrie la sfârșitul fișierului și permite citirea

Lucrul cu fișiere – funcții

fread()
- Citește din fișier o secvență de caractere

fgets()
- Citește din fișier o linie

fgetc()
- Citește din fișier un singur caracter

feof()
- Sfârșitul unui fișier

filesize()
- Returnează dimensiunea unui fișier

Citirea unui fișier linie cu linie folosind fgets

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

```
Line 1
```

Citirea unui fișier porțiune cu porțiune folosind fread

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

```
Lin
```

Lucrul cu fișiere – copy

copy()
- Copiază un fișier

Copierea unui fișier

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

```
<?php
if (!copy('testfile.txt', 'testfile2.txt'))
  echo "Could not copy file";
else
  echo "File successfully copied to 'testfile2.txt'";
?>
```

Lucrul cu fișiere – rename

rename()

- Redenumește un fișier

Mutarea unui fișier

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

Lucrul cu fișiere – unlink

unlink()

- Șterge un fișier

Ștergerea unui fișier

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

Lucrul cu fișiere – fseek

fseek()

- Mutarea pointer-ului în fișier

SEEK_END	Setează pointer-ul la finalul fișierului
SEEK_SET	Setează pointer-ul în poziția specificată
SEEK_CUR	Setează pointer-ul la n poziții de poziția curentă

Lucrul cu fișiere – actualizarea unui fișier

Actualizarea unui fișier

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

```
Line 1
Line 2
Line 3
Line 1
```

Lucrul cu fișiere - Blocarea fișierelor pentru accesări multiple

Actualizarea unui fișier cu file lock

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

Lucrul cu fișiere - Citirea unui întreg fișier

Citirea unui întreg fișier

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

```
<?php
echo file_get_contents("http://oreilly.com");
?>
```

Lucrul cu fișiere - Upload

Encoding: *multipart/form-data*

```
<?php
echo <<<_END
<html><head><title>PHP Form Upload</title></head><body>
<form method='post' action='upload.php' enctype='multipart/form-data'>
Select File: <input type='file' name='filename' size='10' />
<input type='submit' value='Upload' />
</form>
_END;
if ($_FILES)
{
$name = $_FILES['filename']['name'];
move_uploaded_file($_FILES['filename']['tmp_name'], $name);
echo "Uploaded image '$name'<br /><img src='$name' />";
}
echo "</body></html>";
?>
```

Lucrul cu fișiere – Upload $_FILES

Element	Conținut
$_FILES['file']['name']	Numele fișierului uploadat ***Smiley.jpg***
$_FILES['file']['type']	Tipul fișierului uploadat ***Image/jpeg***
$_FILES['file']['size']	Dimensiunea fișierului în octeți
$_FILES['file']['tmp_name']	Numele temporar al fișierului stocat pe server
$_FILES['file']['error']	Codul de eroare returnat în urma upload-ului
application/pdf, image/gif, multipart/form-data, text/xml, application/zip, *image/jpeg, text/css, video/mpeg* *audio/mpeg, image/png, text/html, video/mp4* *audio/x-wav, image/tiff, text/plain, video/quicktime*	

Lucrul cu fișiere – Upload validare

```
<?php
echo <<<_END
<html><head><title>PHP Form Upload</title></head><body>
<form method='post' action='upload2.php' enctype='multipart/form-data'>
Select a JPG, GIF, PNG or TIF File:
<input type='file' name='filename' size='10' />
<input type='submit' value='Upload' /></form>
_END;
if ($_FILES)
{
 $name = $_FILES['filename']['name'];
 switch($_FILES['filename']['type'])
 {
  case 'image/jpeg': $ext = 'jpg'; break;
  case 'image/gif': $ext = 'gif'; break;
  case 'image/png': $ext = 'png'; break;
  case 'image/tiff': $ext = 'tif'; break;
  default: $ext = ''; break;
 }
 if ($ext)
 {
  $n = "image.$ext";
  move_uploaded_file($_FILES['filename']['tmp_name'], $n);
  echo "Uploaded image '$name' as '$n':<br />";
  echo "<img src='$n' />";
 }
 else echo "'$name' is not an accepted image file";
}
else echo "No image has been uploaded";
echo "</body></html>";
?>
```

Lucrul cu fișiere – Upload validare

```
$name = ereg_replace("[^A-Za-z0-9.]", "", $name);
$name = strtolower(ereg_replace("[^A-Za-z0-9.]", "", $name));
```

PHP - Apeluri sistem exec

Listarea conținutului unui director

```
<?php
$cmd = "dir"; // Windows
// $cmd = "ls"; // Linux, Unix & Mac
exec(escapeshellcmd($cmd), $output, $status);
if ($status) echo "Exec command failed";
else
{
echo "<pre>";
foreach($output as $line) echo "$line\n";
}
?>
```

Cuprins

Funcții 1
Funcții built-in 1
Sintagma funcției 1
Funcții – valori implicite pentru parametri 2
Apelul dinamic al funcțiilor 3
Transmiterea prin referință 3
Scopul variabilelor 3
Funcții – returnarea variabilelor globale 4
Includerea de fișiere 5
Funcții – compatibilitate 6
PHP – Obiecte 7
Obiecte – teminologie 7
Obiecte – declarare clasă 7
Obiecte – creare 7
Obiecte – accesare 8
Obiecte - clonare 8
Obiecte - constructori 9
Obiecte – destructori 9
Obiecte – metode 9
Obiecte – metode statice 10
Obiecte – declararea proprietăților 10
Obiecte – declararea constantelor 10
Obiecte - Scopul metodelor și proprietăților 11
Obiecte - Metode și proprietăți statice 11
Obiecte – moșteniri 12
Obiecte – operatorul părinte 12
Obiecte - constructor în subclasă 13
Obiecte – metode finale 13
Șiruri în PHP 14
Șiruri indexate numeric 14

Șiruri indexate prin nume 15
Șiruri - adăugare elemente 15
Șiruri – inițializarea unui șir 15
Șiruri - Parcurgerea șirului de caractere 16
Șiruri - Parcurgerea șirurilor multidimensionale 16
Șiruri – funcții 17
Șiruri – iterator 18
Șiruri – exemple funcții 18
Funcția printf() 20
Printf – exemple 20
Printf – conversie 21
Sprintf 21
PHP – Date & Time 22
Date & Time – exemple 22
Date & Time – format 22
Date & Time – constant 23
Date & Time – checkdate 23
Lucrul cu fișiere în PHP 24
Lucrul cu fișiere – funcții 24
Lucrul cu fișiere – fopen 24
Lucrul cu fișiere – funcții 25
Lucrul cu fișiere – copy 25
Lucrul cu fișiere – rename 26
Lucrul cu fișiere – unlink 26
Lucrul cu fișiere – fseek 26
Lucrul cu fișiere – actualizarea unui fișier 26
Lucrul cu fișiere - Blocarea fișierelor pentru accesări multiple 27
Lucrul cu fișiere - Citirea unui întreg fișier 27
Lucrul cu fișiere - Upload 27
Lucrul cu fișiere – Upload $_FILES 28
Lucrul cu fișiere – Upload validare 28
Lucrul cu fișiere – Upload validare 29
PHP - Apeluri sistem exec 29