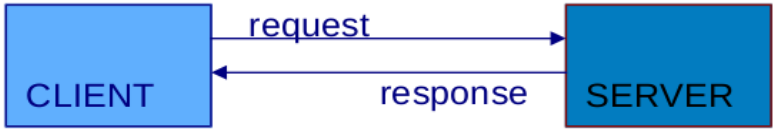
PHP –
Hypertext Preprocessor

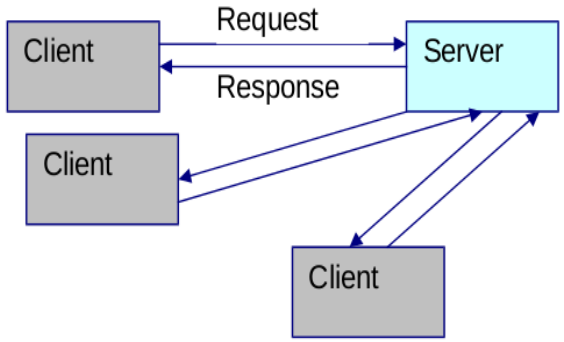
Caracteristici ale limbajului PHP:

- 1994 - Rasmus Lerdorf -**Personal Home Page**
- datele sunt interpretate (preprocesate) de către serverul Web înainte ca acesta să genereze cod HTML.
- *simplitate*
- *uşor de folosit* (C)
- *eficienţă (POO)*
- *gratuit*: PHP reprezintă un program Open Source
- http://www.php.net/
- http://php.net/manual/ro/

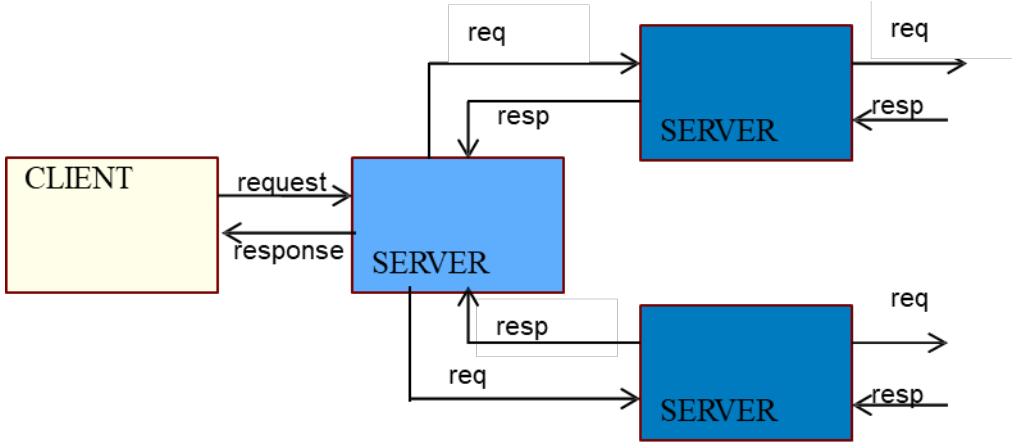
Arhitectura Client-Server

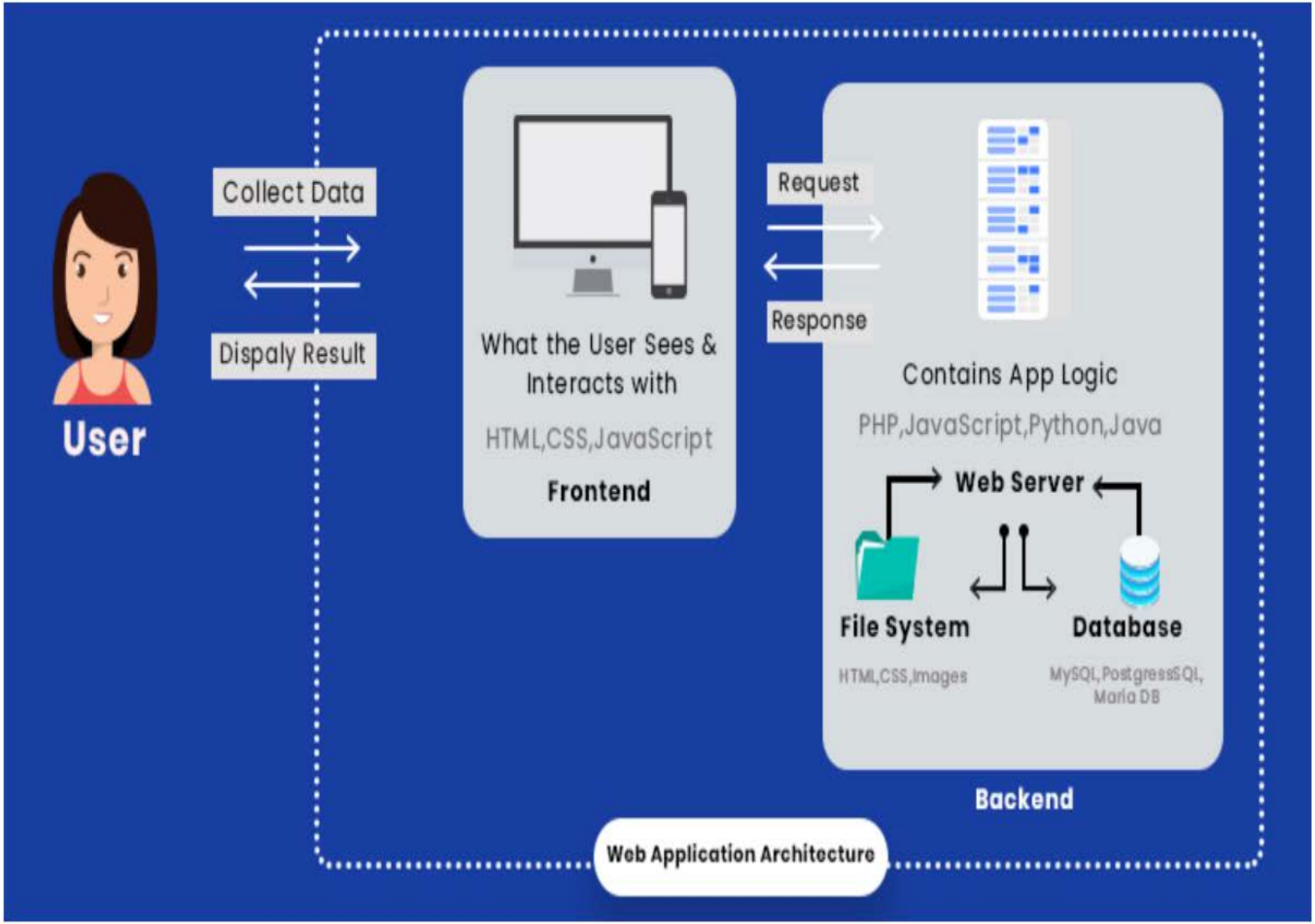


Client-server paradigm

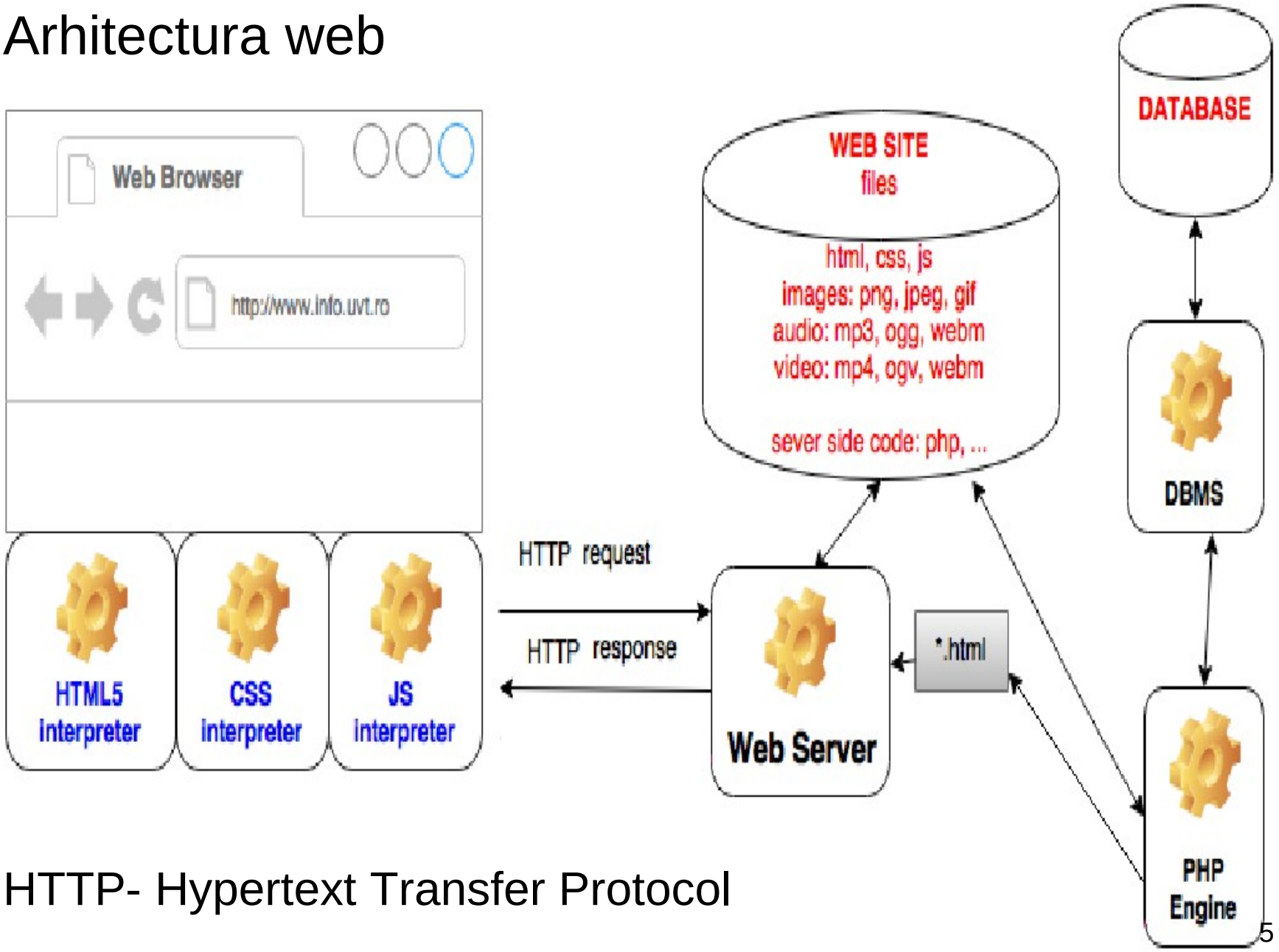


Architecture with one server and more clients



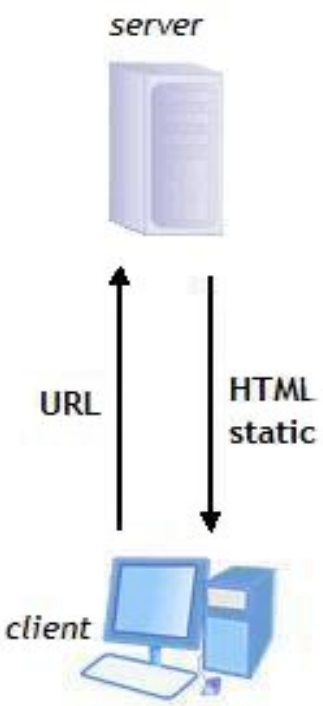

User
Collect Data
Dispaly Result
What the User Sees & Interacts with
HTML,CSS,JavaScript
Frontend
Request
Response
Contains App Logic
PHP,JavaScript,Python,Java
Web Server
File System
HTML,CSS,Images
Database
MySQL,PostgressSQL, Maria DB
Backend
Web Application Architecture

Arhitectura web



HTTP- Hypertext Transfer Protocol

- Deservirea unei pagini statice, fără intervenția interpretorului PHP



- Deservirea unei pagini dinamice, modificată de PHP in momentul cererii

Funcționalitățile PHP

- PHP generază pagini cu un conținut dinamic.
- PHP creează, deschide, citește, scrie și șterge pagini pe un server.
- PHP colectează datele din formulare.
- PHP poate modifica conținutul unei baze de date.

- Programele PHP execută trei categorii de operaţii elementare:

1. Obţin date de la un utilizator.
2. Execută prelucrări ale datelor, respectiv obţin accesul la datele stocate în fişiere şi baze de date şi le manipulează.
3. Afişează date astfel încât un utilizator să le poată vizualiza.

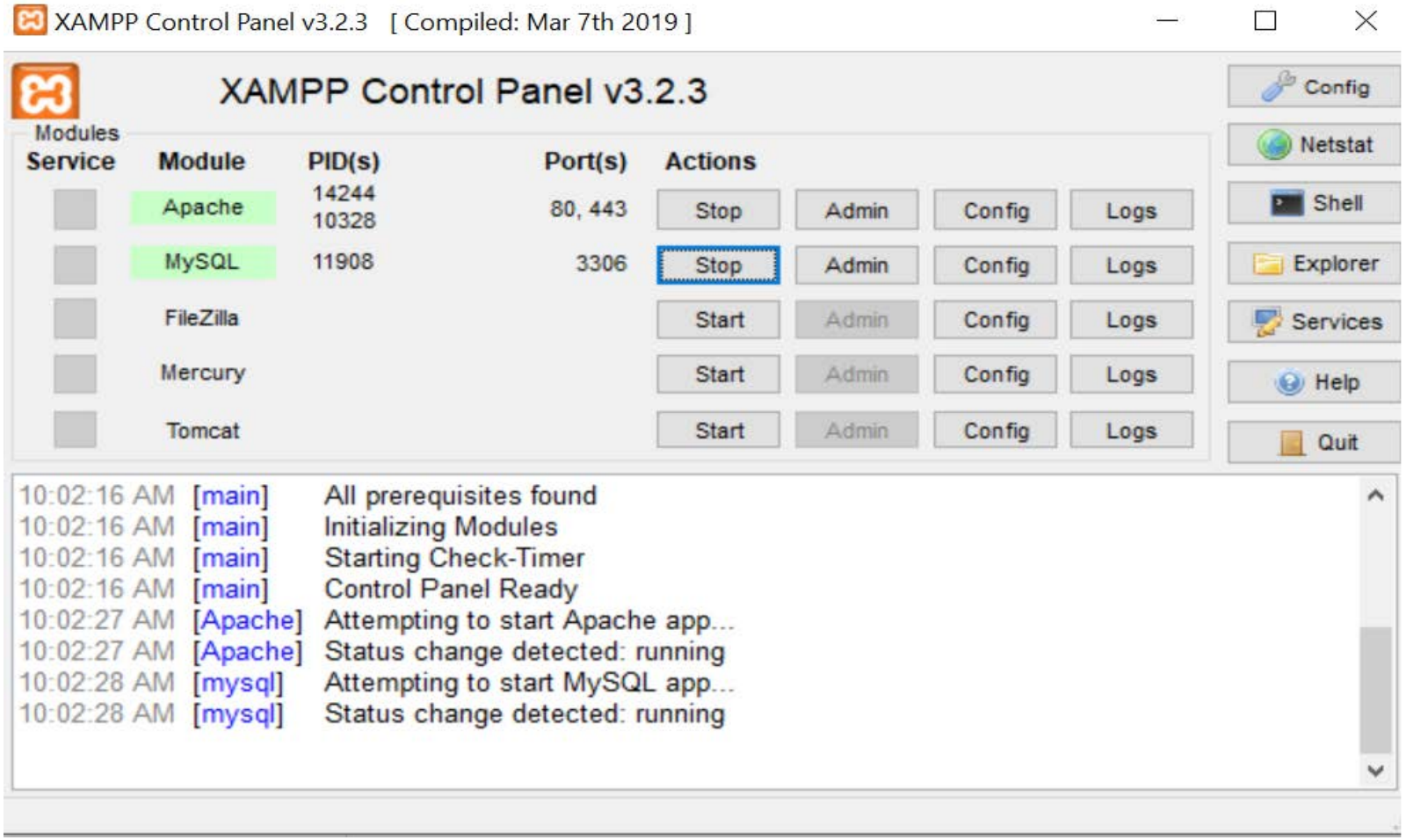
Infrastructura

Comunitatea PHP oferă soluții software în cadrul GNU (General Public License).

- WAMP Server
- LAMP Server
- MAMP Server
- XAMPP Server

Toate aceste tipuri de software se configurează automat în interiorul sistemului de operare, după instalare, având PHP, MySQL, Apache și fișierul de configurare de bază al sistemului de operare, nu trebuie să se configureze manual.

XAMPP – soluție cross-platform (Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P)) este un pachet de programe free software, open source și cross-platform web server, care constă în Apache HTTP Server, MySQL database și interpretoare pentru scripturile scrise în limbajele de programare PHP și Perl



Securitatea

- Folosit pentru medii de dezvoltare.
- Nu este adecvat pentru mediile de producție.

Securitatea lipsește în XAMPP:

- Administratorul MySQL (root) nu are parolă.
- PhpMyAdmin este accesibil prin rețea.
- Pagina demo XAMPP este accesibilă prin rețea.
- Îmbunătățiți securitatea utilizând un router sau un firewall.
- „Consola de securitate XAMPP” permite atribuirea parolelor.
- Directorul principal pentru toate documentele WWW este \xampp\htdocs
- Un fișier "test.html" din acest director, poate fi accesat cu URL-ul "http://localhost/test.html "

Crearea unui script PHP

1. **<?php**

...comenzi PHP...

?>

2. **<script language="php">**

...comenzi PHP...

</script>

3. **<?**

...comenzi PHP...

?>

Un fișier ce conține cod php se salvează cu extensia **.php**

Ex1.

```
<?php
    echo 'Salut, acesta este primul meu script PHP';
?>
```

Ex2.

Într-un script **PHP** se pot însera şi elemente **HTML** ca în exemplul următor:

```
<?php
echo 'php cu  <b>HTML</b> <br> <p>Salut, acesta este script PHP</p>';
?>
```

```
<?php
echo 'php cu  <b>HTML</b> <br>
   <a href="primul_script.php">
    Legatura catre primul meu program php</a>';
?>
```

Vocabularul limbajului este format din:

- cuvinte cheie
- identificatori
- separatori
- comentarii
  - pentru comentarii pe mai multe linii ale textului sursă se utilizează ***/*** ….. *****/***
  - pentru comentariu scris pe o singură linie se utilizează caracterele ***//*** sau #

Instrucțiuni uzuale de afișare în PHP: echo, print, printf

```
<?php
# cele 2 instrucțiuni de mai jos sunt echivalente
print "Text";
echo "Text";
 # o instrucțiune echo poate primi mai mulți parametri
echo "Afisez", " un text din ", 4, " bucati";
 # o singură instrucțiune print poate primi doar un parametru
print "Afisez";
print " un text din ";
print 4;
print  " bucati";
# printf este folosită pentru a formata conținutul, la fel ca in C/C++
printf("Suma %4.2f lei", 102.1234 ); // afiseaza Suma 102.12 lei
?>
```

Tipuri de date

- ***Boolean***
- ***Integer***
- ***Float***
- ***String***
- ***Array***
- ***Object***
- ***Resource***
- ***Null***

În mod normal tipul variabilelor nu este specificat explicit. Acesta va fi evaluat de către interpretorul PHP la momentul run-time (în momentul executării scriptului).

Ex.

```
<?php
$var1 = TRUE;
$var2 = 100;
$var3 = 23.88;
$var4 = "Nume";
$var5 = 'fructe';
echo gettype($var1);
echo '<br />'.gettype($var2);
echo '<br />'.gettype($var3);
echo '<br />'.gettype($var4);
echo '<br />'.gettype($var5);
?>
```

```
boolean
integer
double
string
string
```

- **Boolean**

variabila primeşte valoarea TRUE (adevărat) sau valoarea FALSE (fals):

$a = TRUE;

$a = 0;

- **Integer**

$a = 1234; //numar intreg pozitiv

$a = **0**123; //numar intreg in format octal

$a = **0x**1AB; //numar intreg in format hexazecimal

- **Float**

$a = 1.23;

$a = 1.2e3;

$a = 7E-10;

- **String**

1) **Declararea unui şir prin delimitarea cu apostrof (' ')**
2) **Specificarea unui şir prin folosirea ghilimelelor (" ")**

```
<?php
    $var = 'Acesta este un sir de test';
    echo 'Curs \'PHP\'';
    echo '<br />Vrei sa stergi C:\\*.* ?';
    echo '<br />Variabila var=$var';
    echo " <br /> Numele variabilei este \$var";
    echo " <br /> Variabila var=$var";
?>
```

Rezultatul va fi

Curs 'PHP'
Vrei sa stergi C:*.* ?
Variabila var=$var
Numele variabilei este $var
Variabila var=Acesta este un sir de test

Secvenţele escape folosite în PHP:

- \n - salt la linie nouă
- \r - retur de car (rând nou)
- \t - caracter de tabulare pe orizontală
- \\ - backslash
- \$ - simbolul dolarului
- \” - ghilimele duble

- operatorul de concatenare a şirurilor: **. (punct)**

Ex.

```
<?php
$nume = 'Popescu';
$prenume='Costel';
echo 'Numele de familie este '.$nume.' iar prenumele este ‘.$prenume;
?>
```

Rezultat

Numele de familie este Popescu iar prenumele este Costel

Obs. accesul la un caracter din șir - prin index ($nume[0] , $nume[1], ...)

3) **Sintaxa heredoc: o altă modalitate de a delimita şiruri**

- În acest caz delimitatorul este ("<<<");
- acesta trebuie urmat de un identificator unic, după care urmează şirul de caractere, iar secvenţa se încheie din nou cu identificatorul menţionat.
- Identificatorul de încheiere trebuie să se afle în prima coloană a liniei, acesta poate conține caractere alfanumerice dar neaparat trebuie să înceapă cu o literă

Ex.

```
<?php
$var1 = <<< TEXT
Exemplu de sir care foloseste delimitatorul heredoc.
TEXT;
echo $var1;
?>
```

Ex1.

```
<?php
$nume = "Stefan";
echo <<<DELIMITATOR
Salut! <br/>
Numele meu este $nume.
DELIMITATOR;
?>
```

Rezultat:

Salut!

Numele meu este Stefan!

```
<?php
$name = "Max";
$str = <<<DEMO
Hello $name! <br/>
This is a
demo message
with heredoc.
DEMO;
echo $str;
?>
```

Rezultat:

Hello Max!

This is a demo message with heredoc.

- **Array**

$nume_array[cheie] = valoare;

unde cheie – de tip String sau întreg pozitiv

Ex1.

```
$a['culoare'] = 'rosu'; $a['gust'] = 'dulce';
$a['forma'] = 'rotund'; $a['nume'] = 'mar';  $a[ ] = 4;  // cheia este 0
 // este echivalent cu
$a = array( 'culoare' => 'rosu', 'gust' => 'dulce', 'forma' => 'rotund',
        'nume'  => 'mar', 4  );       // cheia este 0
```

Ex2.

```
$b[ ] = 'a'; $b[ ] = 'b'; $b[ ] = 'c';
// Va rezulta sirul array( 0 => 'a' , 1 => 'b' , 2 => 'c' ),
// sau mai simplu, sirul array('a', 'b', 'c')
```

Referirea la o variabilă de tip tablou se face cu numele tabloului şi cheie:

```
echo "Fructul ".$a['nume']." are gust ".$a['gust']."."";
//Va afisa: Fructul mar are gust dulce.
```

Tipul de date speciale

Resource

- Este un tip special de variabilă care păstrează o legătură spre resurse externe. Exemple de resurse externe: manipulatori pentru deschidere de fişiere, conectare la baze de date, compresia fisierelor, etc...

Object

- Reprezintă de fapt instanţa unei clase declarate în PHP. O clasă este o structură care conţine variabile membru şi funcţii membru.

Null

- Reprezintă variabilele care nu au încă atribuită o valoare.

- **Variabile definite de utilizator**

$var

- **Variabile predefinite**

$GLOBALS – conține referințe la variabilele globale disponibile in scriptul curent.

$_SERVER – variabile definite de server sau relative la contextul in care se execută scriptul curent

$_GET – variabile furnizate scriptului prin adresa URL

$_POST – variabile furnizate scriptului prin metoda HTTP POST (in general prin formulare)

$_FILES – furnizează scriptului fișierele uploadate

$_SESSION – variabile care sunt inregistrate in sesiunea scriptului

- ***Constante***

1. ***definite de utilizator***
2. ***predefinite***

Constante definite de utilizator

- funcția **define()**

Ex.

```
define("ZINATIONALA", "1 DEC");

echo "Ziua nationala este:".ZINATIONALA;
```

Obs. constantele pot defini doar valori ale tipurilor scalare (boolean, întreg, real, string)

Constantă predefinită	Explicaţie
__FILE__	Se utilizează cu un fişier care a fost inclus sau solicitat; când numele fişierului inclus este dat;
__LINE__	Numărul liniei din scriptul curent care este analizat. Se utilizează împreună cu un fişier care a fost inclus sau solicitat, când poziţia în fişierul inclus este dată;
PHP_VERSION	Un şir de caractere reprezentând versiunea PHP utilizată;
PHP_OS	Numele sistemului de operare în care rulează PHP;
TRUE	O valoare adevărată;
FALSE	O valoare falsă;
E_ERROR	Indică o eroare, alta decât o eroare de analizare, pentru care corectarea nu este posibilă;
E_WARNING	Indică o situaţie în care PHP ştie că ceva este greşit, dar continuă oricum;
E_PARSE	Semnalează o sintaxă invalidă în fişierul script; corectarea nu este posibilă;

Operatori

- **Operatorii aritmetici cu variabile**
- **Operatorii de atribuire**
- **Operatorii de incrementare/decrementare**
- **Operatorii de comparaţie**
- **Operatorii la nivel de bit**
- **Operatorii logici**
- **Operatorul ternar**
- **Operatorul de concatenare a şirurilor**

- **Operatorii aritmetici cu variabile**

  + Adunare $a + $b

  - Scadere $a - $b

  *** Inmultire $a * $b**

  / Impartire $a / $b

  % Modulo $a % $b

Ex:

```
<?php
$a = '12';
$b = '8';
$rezultat = $a + $b;
echo 'Rezultatul adunarii lui '.$a.' cu '.$b.' este
'.$rezultat.' ';
?>
```

- **Operatorii de atribuire**

```
$a = 3;      //$a primeste valoarea 3
$a += 5; $a=$a+5      //$a ia valoarea 8, adica 3 + 5
$c = ($b = 4) + 5;
$b=4;
$c=$b+5;
//$b ia valoarea 4, iar $c valoarea 9

$b = “Buna ";
$b .= “ziua!";      // atribuie lui $b valoarea “Buna ziua!",
 //este echivalent cu
$b = $b . "ziua!";
```

Operatorii de incrementare/decrementare

- ++$a Pre-incrementare (adună 1 la $a, apoi returnează $a)
- $a++ Post-incrementare (returnează $a, apoi adună 1 la $a)
- --$a Pre-decrementare (scade 1 din $a, apoi returnează $a)
- $a-- Post-decrementare (returnează $a, apoi scade 1 din $a)

- Operatorii de comparaţie

= =	Egal	$a = = $b
= = =	Identic	$a = = = $b
!=	Diferit	$a != $b
<>	Diferit	$a <> $b
!==	Neidentic	$a ! = = $b
<	Mai mic	$a < $b
>	Mai mare	$a > $b
<=	Mai mic sau egal	$a <= $b
>=	Mai mare sau egal	$a >= $b

Operatorii la nivel de bit

- $a & $b şi
- $a | $b sau
- $a ^ $b sau-exclusiv
- ~ $a not (negare)
- $a << $b Shift la stânga (mută biţii lui $a cu $b poziţii spre stânga)
- $a >> $b Shift la dreapta (mută biţii lui $a cu $b poziţii spre dreapta)

- Operatorii logici

! NOT	!$b
&& AND	$a && $b
|| OR	$a || $b

- Operatorul ternar

conditie?adevarat:fals

Ex.

```
<?php
 $variabila = "student";
 echo $variabila == "student" ? "variabila are valoarea
 student" : "variabila nu are valoarea student";
 ?>
```

- concatenarea şirurilor

- Folosind operatorul .

Ex.

```
<?php
$var1 = 'Ionescu';
echo 'Numele candidatului este '.$var1;
$var2 = 'Candidat: ';
$var2 .= $var1;
echo "<br />$var2";
?>
```

Rezultat:

Numele candidatului este Ionescu

Candidat: Ionescu

Conversia între tipuri de date

Conversie fortata	Rezultat
(int), (integer)	Conversie fortata la întreg
(real), (double), (float)	Conversie fortata la dublu
(string)	Conversie fortata la sir
(array)	Conversie fortata la tablou (matrice)
(object)	Conversie fortata la obiect

Functie	Operatie
doubleval(), floatval()	- Trateaza argumentul ca fiind de tip double.
intval()	- Trateaza argumentul ca fiind de tip întreg.
strval()	- Trateaza argumentul ca fiind de tip string

- Nici conversia normală si nici cea forțată nu modifică tipul unei variabile.
- Ambele mecanisme determină tratarea variabilelor doar în expresia respectivă ca și cum ar fi de alt tip.
- Totuși, modificarea tipului unei variabile este posibilă prin utilizarea funcției **settype()**.

```
$x = 1.5;
settype($x,"integer");
echo $x; // Va afisa valoarea 1
```

Instrucţiuni

- if
- elseif
- switch
- while
- do while
- for
- foreach
- break
- continue
- return

Ex.

```
<?php
$numar = 88;
echo $numar;
if ($numar > 100 )
    echo " Acesta este un numar mai mare decat 100";
elseif ($numar > 10)
        echo " Acesta este un numar mai mic decat 100,
dar mai mare decat 10";
elseif ($numar > 1)
        echo " Acesta este un numar mic";
else
        echo " Acesta este un numar foarte mic";
?>
```

Utilizarea tablourilor

- **Crearea**
  - Tablourile pot fi create folosind 2 metode:
    1 - utilizând direct instrucţiunile de atribuire
    2 - folosind sintaxa array()

Utilizarea tablourilor

Ex.

$clasa[1] = "geometrie"; $clasa[2] = "algebra";

$preferinte[Nelu] = "cirese";
$preferinte[Radu] = "mere";
$preferinte[Gabi] = "pere";

$b["disciplina"]="informatica";

$b["limbajul"]="PHP";

$b["anul"]="1";

```
$limbaje = array("Perl", "PHP", "Python");

$limbaje = array(10=>"Perl", "PHP", "Python");

$limbaje = array("PHP"=>"Ridicat",
"Python"=>"Mediu", "Perl"=>"Redus");

$b=array(
  "disciplina"=>"informatica",
  "limbajul"=>"PHP",
  "anul"=>"1");        //crearea vectorului asociativ
```

- **Accesul la elementele unui tablou**

Exemplu:

```
$limbaje[0] = „Perl";
$limbaje[2] = „Python";

$b["disciplina"]="informatica";
$b["limbajul"]="PHP";
$b["anul"]="1";
```

Parcurgerea iterativă a unui tablou

```
<?php
    $limbaje = array(0=>"Perl", 1=>"PHP", 2 =>"Python");
    $limita = count($limbaje);
    for ($i = 0; $i < $limita; $i++) {
            echo "<br />$i => $limbaje[$i]";
    }
?>
```

Rezultat

0 => Perl
1 => PHP
2 => Python

- **Parcurgerea iterativă a unui tablou non-secvential**

```
foreach (tablou as $cheie => $valoare)
        {
                //      corp
        }
```

Ex1.

```
<?php
   $limbaje = array(10=>"Perl", 20=>"PHP", 21=>"Python");
   foreach ($limbaje as $index => $limbaj)
   {
    // parcurge iterativ tabloul
          echo "<br />$index =>$limbaj";
   }
?>
```

10 =>Perl
20 =>PHP
21 =>Python

Ex2.

```
<?php
$b=array(
  "disciplina"=>"informatica",
  "limbajul"=>"PHP",
  "anul"=>"1");//crearea vectorului asociativ
echo "<br>Elementele vectorului asociativ <br>";
foreach ($b as $i=>$denumire)
  echo "<br>".$i."=".$denumire."  ";
?>
```

Elementele vectorului asociativ

disciplina=informatica
limbajul=PHP
anul=1

Tablouri multi-dimensionale

Ex3.

```
<?php
    $multiDimArray[“firstLine”] = array(1=>10, 2=>20, "a"=>"alpha");
    $multiDimArray["nextLine"] = array(1=>20, 2=>40, "b"=>"beta");
    echo "<br />".$multiDimArray["firstLine"][1];
    echo "<br />".$multiDimArray["nextLine"][1];
    echo "<br />".$multiDimArray["firstLine"][2];
    echo "<br />".$multiDimArray["nextLine"][2];
    echo "<br />".$multiDimArray["firstLine"]["a"];
    echo "<br />".$multiDimArray["nextLine"]["b"];
?> echo "<br />
```

Obs. Definire echivalentă (un array cu 2 linii şi 3 coloane)

```
$multiDimArray = array("firstLine"=>array(1=>10, 2=>20, "a"=>"alpha"),
"nextLine"=>array(1=>20, 2=>40, "b"=>"beta"));
```

Lucrul cu funcţii listă

- **current()** - returneaza valoarea curentă a tabloului
- **next()** - referire la următorul element
- **prev()** - referire la elementul anterior
- **key()** - returnează cheia asociată elementului curent
- **each()** - returnează perechea "cheie-valoare" care se află la poziția curentă
- **list() -** permite atribuirea de valori la mai multe variabile în cadrul unei instrucțiuni

list($var1, $var2, ……, $varn) = valoare_tablou;

- **sort(tablou)** - sortare în funcție de valoare
- **rsort(tablou)** – sortare in ordine inversă în funcție de valoare
- **asort()** ; **arsort()**;
- **ksort()** - sortare în funcție de cheie, in ordine ascendentă;
- **krsort()**

- Perechea "cheie-valoare" este returnată sub forma unui tablou asociativ cu patru elemente, după cum urmează:

Cheie	Valoare
0	- Componenta cheie a perechii cheie-valoare curenta
1	- Componenta valoare a perechii cheie-valoare curenta
key	- Componenta cheie a perechii cheie-valoare curenta
value	- Componenta valoare a perechii cheie-valoare curenta

Ex.

```
<?php
    $limbaje = array(10=>"Perl",
    20=>"PHP", 21=>"Python");
    $fiecare = each($limbaje);
    $zero = $fiecare[0];
    $unu = $fiecare[1];
    $cheie = $fiecare['key’];
    $valoare = $fiecare['value’];
    echo "<br />zero=$zero";
    echo "<br />unu=$unu";
    echo "<br />cheie=$cheie";
    echo "<br />valoare=$valoare";
?>
```

Rezultat:

zero=10
unu=Perl
cheie=10
valoare=Perl

Funcţii

- **abs(x)** - Returnează valoarea absolută a lui 'x'
- **ceil(x)** - Returnează valoarea 'x', rotunjită la întregul imediat superior
- **floor(x)** - Returnează valoarea 'x', rotunjită la întregul imediat inferior
- **max(x,y,...)** - Returnează valoarea maximă a unui set de valori
- **min(x,y,...)** - Returnează valoarea minimă a unui set de valori
- **pow(x,n)** - Returnează numarul 'x', ridicat la puterea 'n'
- **strftime(f)** - Returnează data curentă, formatată conform conținutului parametrului 'f'
- **sqrt(x)** - Returnează rădacina pătrată a lui 'x'

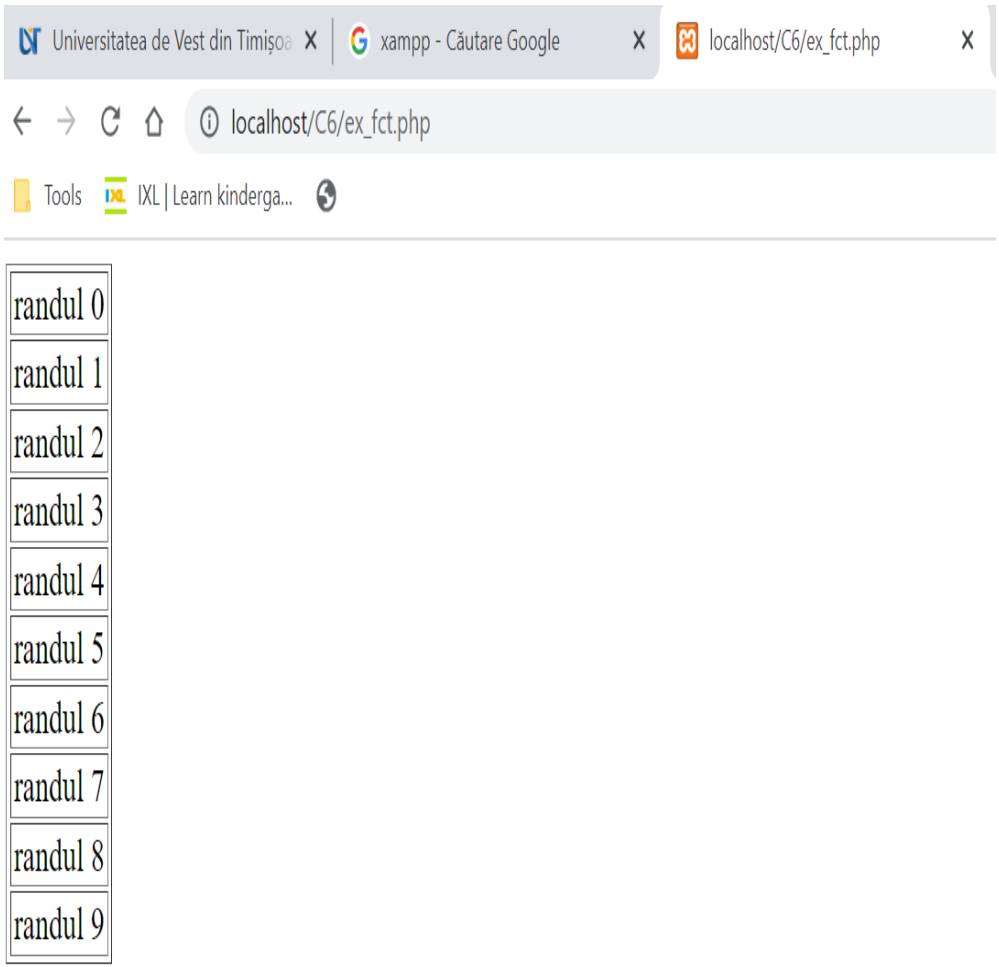
Funcţii

```
function nume_functie(nume_argument) {
        // corpul functiei
   }
```

```
<?php
function arie($lungime, $latime) {
        return $lungime * $latime;
}
$rezultat = arie(5,3);
echo "Aria este : $rezultat";
?>
```

Aria este : 15

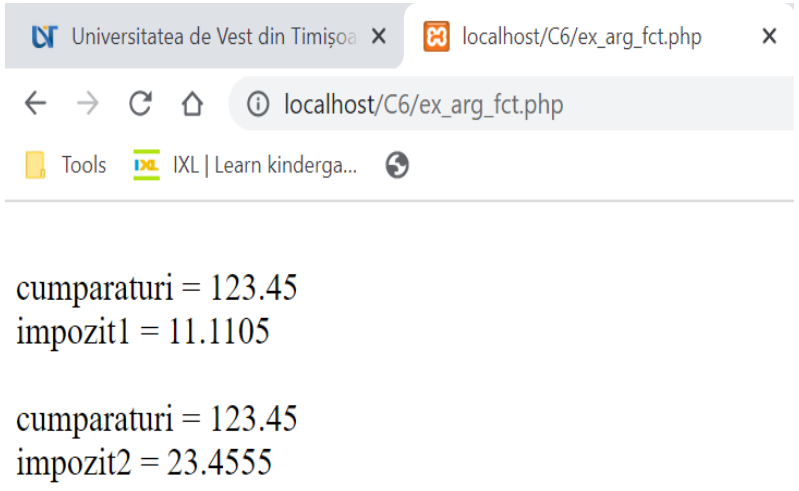
```php
<?php
function tabel($lim) {
echo "<table border=\"1\">\n";
for ($i=0; $i<=$lim; $i++) {
echo "<tr><td>randul
".$i."</td></tr>\n";
}
echo "</table>";
}
tabel(9); //tabel cu 10 randuri
?>
```



Definirea argumentelor prestabilite

PHP permite definirea de funcţii cu argumente prestabilite.

La apelul funcţiei care are un argument prestabilit, dar nu este furnizată nici o valoare pentru argumentul respectiv, argumentul ia valoarea prestabilită specificată la început.




```
<?php
function impozit_vanzari($cantitate, $rata = 0.19) {
        return $cantitate * $rata;
}
$cumparaturi = 123.45;
echo "<br />cumparaturi = $cumparaturi";
$impozit1 = impozit_vanzari($cumparaturi, 0.09);
echo "<br />impozit1 = $impozit1";
$cumparaturi = 123.45;
echo "<br /><br />cumparaturi = $cumparaturi";
$impozit2 = impozit_vanzari($cumparaturi);
echo "<br />impozit2 = $impozit2";
?>
```

- O instrucțiune **return** determină terminarea execuției functiei care o conține
- **Terminarea execuției unui script- functia exit()**
- **Funcții recursive**

```
<?php
function f($nr) {
        $nr++;
        if ($nr<8) {
                return f($nr);
        }
        return $nr;
}
$x = f(3);
echo $x;
?>
```

Observatie

- O functie utilă, recomandată a fi folosită în scripturi este funcţia **isset()** - este cel mai des folosită cu "if()".
- **isset()** preia ca argument de obicei o variabilă şi arată dacă aceasta a fost sau nu setată.

Ex1.

- echo “Variabila \$nr este setata? ”.**isset($nr) ;**

Ex2.

```
<?php
   if (isset($_GET['id'])) {
         // Se executa codul dorit
   echo “\$id are valoarea “.$_GET['id'];
   }
?>
```

- Variabilele funcțiilor in PHP:

- **Variabile globale**
- **Variabile locale**

Variabilele globale sunt create în exteriorul functiei, în timp ce variabilele locale sunt create în interiorul unei functii.

- Variabilele globale nu pot fi accesibile din interiorul corpului unei funcții
- accesul la o variabilă globală în cadrul unei funcții, se poate realiza prin folosirea instructiunii **GLOBAL**.

GLOBAL variabila1, variabila2, variabila3;

```
<?php
$var1 = 135;
$var2 = 250;
function Suma() {
        return $var1 + $var2;
}
echo "Suma este ". Suma();
?>
```

NULL

```
<?php
$var1 = 135;
$var2 = 250;
function Suma() {
        GLOBAL $var1, $var2;
        return $var1 + $var2;
}
echo "Suma este ". Suma();
?>
```

385

$GLOBALS este o variabilă predefinită, este de fapt un array ; elementele acestei matrice au cheia egală cu numele variabilelor declarate și valoarea egală cu cea a variabilelor declarate.
$GLOBALS este o variabilă superglobală, ea va fi recunoscută in orice script. Ex. $GLOBALS['var1']

```
<?php
$x = 135;
$y = 250;
function suma() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}
 suma();
echo $z;
?>
```

```
<?php
function v_local() {
        $x = 5;
        echo "<br />In corpul functiei x = $x";
}
$x = 2;
echo "<br />In corpul scriptului x = $x";
v_local();
echo "<br />In corpul scriptului x = $x";
?>
```

Rezutat:

In corpul scriptului x = 2
In corpul functiei x = 5
In corpul scriptului x = 2

Variabile statice
static $var1, $var2;

```
<?php
function v_local() {
        $x=0;
        $x = $x + 1;
        echo "<br /> x = $x";
}
function v_static() {
        STATIC $x;
        $x = $x + 1;
        echo "<br /> x = $x";
}
v_local();
v_local();
v_local();
echo "<br /> apel functie cu
    x static de 3 ori";
v_static();
v_static();
v_static();
?>
```

Transferul argumentelor unei funcții: prin valoare și referință(**&**)

```
<?php
function p_valoare($a) {
        $a = $a+10;
         echo "<br /> in functia p_valoare, valoarea este  $a";
}
function p_referinta(&$a) {
        $a = 2;
        echo "<br /><br /> in functia p_referinta, valoarea este  $a";
}
$b = 0;
echo "<br /> b = $b";
p_valoare($b);
echo "<br /> la iesirea din functie b = $b";
$b = 0;
p_referinta($b);
echo "<br /> la iesirea din functie b = $b";
?>
```

b = 0
in functia p_valoare, valoarea este 10
la iesirea din functie b = 0
in functia p_referinta, valoarea este 2
la iesirea din functie b = 2

Lucrul cu functii create /definite in script

call_user_func("func", arg)

- Apelează funcția "func" dându-i valoarea argumentului "arg". Returnează rezultatul funcției apelate sau FALSE.

Ex.

```
<?php
function func($arg1, $arg2) {
  return "$arg1 - $arg2";
}
// Apeleaza func()
echo call_user_func("func", 'Curs', 'PHP');
// Sau o metoda mai avansata, prin crearea si apelarea functiei in
//acelasi timp (PHP 5.3 +)
echo call_user_func(function($arg) { echo "Tutoriale $arg"; }, 'php');
?>
```

ex_fct_call1.php

call_user_func_array("func", array_arg)

- Apelează o funcție ("func") dându-i valoarea argumentelor dintr-un Array (array_arg). Returnează rezultatul funcției apelate sau FALSE.

Ex.

```
<?php
// Functie creata standard
function func1($arg1, $arg2) {
  return $arg1 + $arg2;
}
// Functie creata in variabila (PHP
// 5.3 +)
$func2 = function($arg1, $arg2) {
  return $arg1 * $arg2;
};
// Matricea pt. parametri
$aray_p = array(7, 8);


// Apeleaza func1()
echo call_user_func_array("func1",
$aray_p);

// Apeleaza $func2
echo call_user_func_array($func2,
$aray_p);
?>
```

ex_fct_call2.php

create_function("argumente", "cod_functie")

- Crează dinamic o funcție ce va avea argumentele "argumente" și codul ce-l execută "cod_functie". Returnează funcția creată (*poate fi asociată cu un nume de variabilă*), sau FALSE. (*"argumente" și "cod_functie" trebuie să fie șiruri*)

Ex.

```
<?php
$var = create_function('$arg', 'return $arg*2;');
echo $var(8);       // 16
?>
```

func_num_args()

Returnează numărul de argumente primite de o funcție. Se folosește in interiorul acelei funcții.

Ex.

```
<?php
function test() {
  // Preia si afiseaza numarul de argumente primite
  $nr_args = func_num_args();
  echo "Numar argumente primite: $nr_args";
}
// Apeleaza functia
test(1, 2, 'a');       // Numar argumente primite: 3
?>
```

func_get_arg(nr)

- Returnează argumentul cu index-ul "nr" din cele primite de o funcție (*primul argument are index 0*). Se folosește in interiorul acelei funcții. Dacă numărul de argumente e mai mic decât "nr", returnează FALSE.

Ex.

```
<?php
function test() {
  // Preia numarul de argumente primite
  $nr_args = func_num_args();
  // Daca nr. argumente >=2, afiseaza argumentul cu index 2
  if ($nr_args>=2) {
    echo 'Argumentul cu indice 2 este: '. func_get_arg(2);
  }
}
// Apeleaza functia
test(1, 'php', ‘CURS');       // Argumentul cu indice 2 este: CURS
?>
```

func_get_args()

- Returnează o matrice cu argumentele primite de o funcție. Se folosește in interiorul acelei funcții.

Ex.

```
<?php
function test() {
  // Preia argumentele primite
  $array_args = func_get_args();
  // Afiseaza structura matricei cu argumentele
  print_r($array_args);
}

// Apeleaza functia
test(1, ‘Curs', ‘PHP');       // Array ( [0] => 1 [1] => Curs [2] => PHP )
?>
```

Şiruri

- Crearea şi afişarea şirurilor
- Crearea datelor de ieşire formatate:
  - **printf() –** afișează un șir formatat
  - **sprintf()** – returnează un șir formatat
- Specificatorii de tip PHP folosiţi la formatarea şirurilor
- Funcția **`number_format()`** returnează o valoare de tip șir conținând un rezultat formatat.
- Funcția number_format()
  http://php.net/manual/ro/function.number-format.php
- Funcția **`sscanf()`** prelucrează un şir citit în funcţie de un format.
- Funcția sscanf()
  http://php.net/manual/ro/function.sscanf.php

Funcţii ale şirurilor

- **strlen()**
- **strtoupper()** returnează valoarea convertită la majuscule
- **strtolower()** returnează valoarea convertită la minuscule
- **ucfirst() r**eturnează valoarea cu primul caracter din şir majusculă
- **str_replace(cauta, înlocuire, subiect)**
- **substr_replace(subiect, înlocuire, start, lungime)**
- **Compararea şirurilor** **fct_compar_sir.pdf**
- **Eliminarea spațiilor dintr-un şir** **fct_elimin_sir.pdf**

Expresii regulate Regex

Expresiile regulate (regex) sunt un șir de caractere șablon care descriu mulțimea cuvintelor posibile care pot fi formate cu acele caractere, respectând anumite reguli. Aceste expresii regulate folosesc paranteze (*rotunde, pătrate, acolade*) prin care generează regulile de formare a cuvintelor.

Utilitatea cea mai frecventă a unei expresii regulate constă în a recunoaște dacă un șir conține sau nu cuvinte sau sub-șir care pot fi formate prin expresia regulată respectivă.

Ex.

1. expresia **m[ai]r** poate forma cuvintele: *mar* si *mir*

2. expresia regulată **[aeiou]{1,4}** corespunde șirurilor care pot conține intre 1 și 4 vocale

3. expresia regulată **([sml]at){1 ,2}** corespunde unui număr de una sau două repetări ale oricăruia dintre șirurile "sat", "mat" sau "lat".

Structura regex in PHP

O expresie regulată este de forma și obligatoriu in ordinea următoare:

- delimitator
- reguli
- modificatori

Delimitatorul - este cel care separă regulile de modificatori, specificând că este o expresie regulată. Delimitatorul recomandat este caracterul /(slash)

Regulile - sunt scrise cu ajutorul **operatorilor regex (caractere speciale regex** sau altfel spus **meta caracterele)** și caracterelor simple.

Modificatorii - sunt cei care modifică modul cum este înțeles regex-ul de PHP (Ex. modificatorul **i** spune ca regex-ul să se facă in mod **case insensitive**)

o listă cu mai multe caractere speciale și rolul lor in expresiile regulate:

^ - indică inceputul liniei

$ - indică sfârșitul liniei

. - (punct) orice caracter

[] - specifică oricare caracter dintre cele din paranezele pătrate

[^] - orice caracter, in afara celor din paranezele pătrate

**** - scoate din contextul formării expresiei caracterul care urmează

+ - caracterul (expresia) anterior acestui semn se poate repeta o data și de câte ori este posibil (*de la 1 la infinit*)

***** - caracterul (expresia) anterior acestui semn se poate repeta de câte ori este posibil sau niciodată (*de la 0 la infinit*)

? - caracterul (expresia) anterior acestui semn se poate repeta cel mult o dată

<> - un cuvânt întreg

(|) - lista de opțiuni (operatorul boolean SAU)

{m, n} - repetarea expresiei de la "m" la "n" ori

d - specifică un caracter numeric – este similar cu **[0-9]**

D - specifică un caracter non-numeric – este similar cu **[^0-9]**

w - specifică un caracter alfanumeric – este similar cu **[a-zA-Z0-9]**

W - specifică un caracter non-alfanumeric – este similar cu **[^a-zA-Z0-9]**

[:alnum:] - reprezintă un caracter alfanumeric. Similar cu **w** și **[A-Za-z0-9]**

[:alpha:] - reprezintă caracterele alfabetice. Similar cu **[A-Za-z]**

[:blank:] - reprezintă space sau blank. Similar cu **[ ]**

[:digit:] - reprezintă un număr. Similar cu **[0-9]**

[:upper:] - reprezintă majusculele. Similar cu **[A-Z]**

[:lower:] - reprezintă minusculele. Similar cu **[a-z]**

Funcția preg_match

- Funcția preg_match este folosită pentru a testa dacă o expresie regulată se potriveste cu un string. Ea returnează 1 in caz ca **patternul**(expresia regulată) se potrivește cu **subiect**-ul (stringul).

- Functia preg_match poate fi folosită in general pentru verificări:

dacă o adresa e-mail este validă

dacă o parolă este sigură

dacă un username conține caractere valide

dacă un număr de telefon este valid

Verificare dacă stringul conține numere

```
<?php
$string = 'Acest string contine 1 numar.';
$regex  = '/[0-9]/';
if(preg_match($regex, $string)){
    echo 'Acest string contine numere.';
} else{
    echo 'Acest string nu contine numere.';
}
?>
```

Rezultat

Acest string contine numere.

```
<?php
$string = 'Acesta este un string.';
if(preg_match('/este/', $string)){
    echo 'Cuvantul <b>este</b> a fost gasit in stringul dat.';
} else{
    echo 'Cuvantul <b>este</b> nu a fost gasit in stringul dat.';
}
?>
```

Rezultat

Cuvantul **este** a fost gasit in stringul dat.

```
<?php
    if(isset($_GET['email'])):
        $adresaEmail = $_GET['email'];
    $pattern = '/^[a-zA-Z0-9_-]{3,30}@[a-zA-Z0-9-]{3,30}.[a-zA-Z]{1,4}$/';
        if(preg_match($pattern, $adresaEmail)){
            echo 'Adresa de e-mail este corecta.';
        } else{
            echo 'Adresa de e-mail nu este corecta.';
        }
        echo '<hr/><a href="?">Doresc sa testez din nou.</a>';
    else:
        echo '<form>';
            echo '<input type="text" name="email" placeholder="Introduceti
adresa de e-mail"/>';
            echo '<input type="submit" value="Verifica daca este valida"/>';
        echo '</form>';
    endif;
?>
```

ex_verif_email.php