

Синтаксис на РНР

Лектор: Милен Петров

Лекция 2

Ретроспекция

1. Не можете да видите сървърният код, посредством “View source” в брауъра – а само резултата от изпълнението на сървърният код, който е чист текст/HTML.
2. Кодът на PHP започва с **<?php** и завършва със **?>**.
3. Има съкратен синтаксис – започващ с **<?** и завършващ **?>** (не се препоръчва, т.к. изисква допълнителна конфигурация).
4. Всеки ред в PHP завършва с **;** (например: `echo "Hello World";`)

Коментари в PHP

- използва се `//` за едноредов коментар
- и `/* ... */` за многоредов коментар

```
<html>
<body>
<?php
//This is a comment (едноредов коментар)
/*
Многоредов коментар
*/
?>
</body>
</html>
```

Променливи

- Използват се за съхранение на стойности като числа, символни низове или резултати от функции с цел многократната им употреба в даден скрипт.
- Започват със знакът \$
- Могат да са числа, символни низове или масиви
- Пример:

```
<html>
<body>
<?php
$mytxt="Това е текст";
echo $mytxt ;
?>
</body>
</html>
```

Конкатенация

- Използва се dot оператора (.), например:

```
<html>
```

```
<body>
```

```
<?php
```

```
    $text1="Hello";
```

```
    $text2="My Friend";
```

```
    echo $text1 . " " . $text2 ;
```

```
?>
```

```
</body>
```

```
</html>
```



Резултат "Hello My Friend"

Правила за именуване на променливи

- Името на променливата трябва да **започва** с буква или подчертавка (underscore "_")
- Името на променливата трябва да **съдържа** само букви, цифри и подчертавка (a-Z, 0-9, и _)
- Името на променливата не трябва да съдържа интервали. Ако трябва да се състои от повече от 1 думи – те се разделят с “_” - \$first_name или с главна буква - \$firstName

Оператори в РНР

- Операторите се използват да извършват действия (оперират) над стойности.
 - аритметични (например +, -)
 - за присвояване (например =, +=, *=)
 - за сравнение (например ==, !=, >=)
 - логически (&&, ||, !)

Аритметични оператори

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Аритметични оператори (2)

- В PHP няма оператор за целочислено делене;
- при преобразуване (кастване) – винаги се закръгля надолу, например: `$a = (int) 3.7;`

```
<?php
```

```
var_dump(25/7);    // float(3.5714285714286)
```

```
var_dump((int) (25/7)); // int(3)
```

```
var_dump(round(25/7)); // float(4)
```

```
?>
```

Оператори за присвояване

Assignment Operators

Operator	Example	Is The Same As
=	$x = y$	$x = y$
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x \% = y$	$x = x \% y$

Оператори за сравнение в PHP

Example	Name	Result
<code>\$a == \$b</code>	Equal	TRUE if <i>\$a</i> is equal to <i>\$b</i> after type juggling.
<code>\$a === \$b</code>	Identical	TRUE if <i>\$a</i> is equal to <i>\$b</i> , and they are of the same type.
<code>\$a != \$b</code>	Not equal	TRUE if <i>\$a</i> is not equal to <i>\$b</i> after type juggling.
<code>\$a <> \$b</code>	Not equal	TRUE if <i>\$a</i> is not equal to <i>\$b</i> after type juggling.
<code>\$a !== \$b</code>	Not identical	TRUE if <i>\$a</i> is not equal to <i>\$b</i> , or they are not of the same type.
<code>\$a < \$b</code>	Less than	TRUE if <i>\$a</i> is strictly less than <i>\$b</i> .
<code>\$a > \$b</code>	Greater than	TRUE if <i>\$a</i> is strictly greater than <i>\$b</i> .
<code>\$a <= \$b</code>	Less than or equal to	TRUE if <i>\$a</i> is less than or equal to <i>\$b</i> .
<code>\$a >= \$b</code>	Greater than or equal to <small>@Milen Petrov, milen.petrov@gmail.com</small>	TRUE if <i>\$a</i> is greater than or equal to <i>\$b</i> .

Оператори за сравнение

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Логически оператори

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
	or	x=6 y=3 (x==5 y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

Условни (if else elseif) конструкции

- Използват се за извършване на различни действия, на базата на зададено условие или условия;
- Налага се да се използва често
- Разновидности:
 - **if...else** изречение , използващо се за изпълнение на едно действие, ако условието е истина, и друго в противен случай
 - **elseif** изречение (statement) – използва се в комбинация с **if...else** за да изпълни даден код, ако е възможно повече от едно условие да е вярно

The If...Else Statement (1)

- Изпълнява се един оператор ако условието е истина (true) и друго в противен случай се използва **if...else** оператор

Syntax :

if (*condition*)

code to be executed if condition is true;

else

code to be executed if condition is false;

The If...Else Statement (2) - пример

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
    echo "Have a nice weekend!";
else
    echo "Have a nice day!";
?>
</body>
</html>
```

- Ако текущият ден е петък – пожелава приятен уийкенд, в противен случай – приятен ден.

The If...Else Statement (3) - пример

- При повече редове за изпълнение – използването на {} (скоби) е задължително

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
{
    echo "Hello!<br />";
    echo "Have a nice weekend!";
    echo "See you on Monday!";
}
?>
</body>
</html>
```

The Elself Statement

Syntax:

if (*condition*)

code to be executed if condition is true;

elseif (*condition*)

code to be executed if condition is true;

else

code to be executed if condition is false;

The Elself Statement - пример

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
    echo "Have a nice weekend!";
elseif ($d=="Sun")
    echo "Have a nice Sunday!";
else
    echo "Have a nice day!";
?>
</body>
</html>
```

Switch Statement

- Ако искате един от няколко възможни блока код да бъде изпълнен – използвайте switch
- Използва се за избягване на множество if..elseif..else в кода.
- Switch в PHP се използва за да се предприемат различни действия в зависимост от няколко различни условия;

Switch - синтаксис

Syntax

```
switch (expression)
{
case label1:
    code to be executed if expression = label1;
    break;
case label2:
    code to be executed if expression = label2;
    break;
default:
    code to be executed
    if expression is different
    from both label1 and label2;
}
```

Switch – начин на работа

- Оценява се единъж стойността на един израз (често променлива)
- Стойността се сравнява последователно с всяка стойност в case структурата
- Ако има съвпадение, съответният case се изпълнява
- След като се изпълни съответният case, се използва break, който предпазва от изпълнението на следващият case
- Стойността по подразбиране (default) се изпълнява, ако никой от case случаите не съвпада.

Switch – пример (1)

```
<html>
<body>
<?php
switch ($x)
{
case 1:
    echo "Number 1";
    break;
case 2:
    echo "Number 2";
    break;
case 3:
    echo "Number 3";
    break;
default:
    echo "No number between 1 and 3";
}
?>
</body>
</html>
```

Задачи

- **Задача:** Дадени са три числа $\$a$, $\$b$ и $\$c$ -> да се изведе максималното от тях;
// $\$a = 10$; $\$b = 12$; $\$c = 3$;
- **Задача:** по зададено ресто – да се изведат минималният брой и тип на всяка от монетите;

Изход: Ресто – 79 стотинки, се връщат:

1 монета от 50 стотинки

1 монета от 20 стотинки

1 монета от 5 стотинки

2 монети от 2 стотинки

Общо: 79 стотинки

Масиви в PHP

- Когато работите с PHP, рано или късно ще стигнете до необходимостта да съхраняване на **множество подобни променливи** (например променлива a_1 , a_2 , ..., a_{100}).
- Вместо множество променливи – за съхранение на стойностите им като елементи на 1 променлива, може да се използва **масив**;
- Всеки елемент на масива си има идентификатор/номер – т.нар. **индекс** – с цел по-лесното му достъпване.
- Масив е възможността за съхраняване на повече стойности в една променлива;

Видове масиви в PHP

- числови (Numeric array) — ключът им за достъп (индекса) е цяло число
- асоциативни (Associative array) – масив, при който на всеки ключ съответства стойност
- Многомерни (Multidimensional array) – състоящи се от един или повече масиви

Числови масиви (1)

- Автоматично задаване на ключ:

- Пример:

```
$summer_cities = array("Sozopol", "Barcelona", "Santorini",  
    "Chernomoretz", "Thassos");
```

Числови масиви (2)

- Ръчно задаване на ключ:

- Пример:

```
$summer_cities[0] = "Sozopol";
```

```
$summer_cities[1] = "Barcelona";
```

```
$summer_cities[2] = "Santorini";
```

```
$summer_cities[3] = "Chernomoretz";
```

```
$summer_cities[4] = "Thassos";
```

Масиви - използване

```
<?php
$summer_cities[0] = " Sozopol";
$summer_cities[1] = " Barcelona";
$summer_cities[2] = " Santorini";
$summer_cities[3] = " Chernomoretz";
$summer_cities[4] = " Thassos";
echo "I visited " . $summer_cities[0] . " in june 2014, " . $summer_cities[1] . " and
    " . $summer_cities[2] .
    " in july 2014, and in the end " . $summer_cities[3] .
    . " and " . $summer_cities[4] . " in august 2014. It was a great summer!"
;
?>
```

Асоциативни масиви

- За всеки ключ се асоциира стойност; Не за всеки случай номерирането е най-удобният начин за съхраняване;

- **Пример:**

```
$summer_cities = array("Sozopol" => "June", "Barcelona"=>  
    "June", "Santorini" =>"July", "Chernomoretz" =>"July",  
    "Thassos"=>"August");
```

Асоциативни масиви - пример

- **Пример:**

`$summer_cities["Sozopol"] = "June";`

`$summer_cities["Barcelona"] = "June";`

`$summer_cities["Santorini"] = "July";`

`$summer_cities["Chernomoretz"] => "July";`

`$summer_cities["Thassos"] = "August";`

Асоциативни масиви - пример

```
<?php
$summer_cities["Sozopol"] = "June";
$summer_cities["Barcelona"] = "July";
$summer_cities["Santorini"] = "July";
$summer_cities["Chernomoretz"] =>"August";
$summer_cities["Thassos"]="August";

echo "I visited Barcelona in " . $summer_cities['Barcelona'] . "!";
?>
```

Резултат: I visited Barcelona in July!

Многомерни масиви

- Всеки елемент от масива може да бъде масив. Всеки елемент от под-масива също може да бъде масив и т.н.
- `$vacation = array ("June"=> array("Sozopol"),
"July"=>array("Barcelona", "Santorini"),
"August"=>("Chernomoretz", "Thasos"));`

Цикли в PHP

- Често се налага повторението на един и същи набор от команди много пъти. За целта се използва цикъл.
- Съществуват следните видове цикли:
 - **while** – изпълнява даден блок с програмен код толкова пъти, колкото е изпълнено зададено условие;
 - **do...while** - изпълнява даден код веднъж и след това го изпълнява толкова пъти, колкото е изпълнено дадено условие;
 - **for - loops** изпълнява даден блок определен брой пъти;
 - **foreach** – изпълнява толкова на брой пъти тялото на цикъла колкото елементи има в даден масив; използва се за обхождане елементи на масиви;

Цикъл while

- Изпълнява се докато условието е вярно.

- Синтаксис:

while (*condition*)

code to be executed;

Цикъл while - пример

```
<html>
<body>
<?php
$i=1;
while($i<=50)
{
    echo "Номер " . $i . "<br />";
    $i++;
}
?>
</body>
</html>
```

Цикъл do . . . while

- Изпълнява блокът с код поне веднъж; след това повтаря изпълнението толкова пъти колкото пъти е изпълнено условието;
- **Синтаксис:**

```
do
{
    code to be executed;
}
while (condition);
```

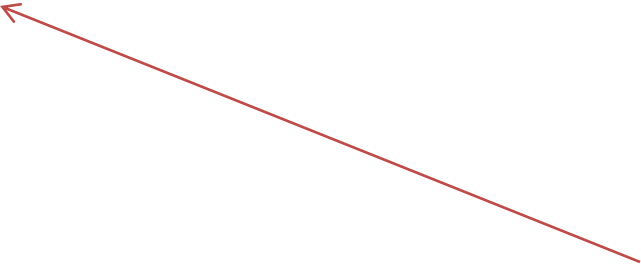
Цикъл do . . . While - пример

```
<html>
<body>
  <?php
    $i=0;
    do
    {
      $i++;
      echo "Номер " . $i . "<br />";
    }
    while ($i<10);
  ?>
</body>
</html>
```

Цикъл for

- Използва се, когато броят на изпълнение на програмен код е познат (например: преброеете до 10).
- **Синтаксис:**

```
for (initialization; condition; increment)  
{  
    code to be executed;  
}
```



Ако има повече от едно условие – се разделят със запетайка (,)

Цикъл for - пример

The following example prints the text "Hello World!" five times:

```
<html>
<body>
<?php
for ($i=0; $i<9; $i++)
{
    echo "Номер " . ($i + 1) . " <br />";
}
?>
</body>
</html>
```


Цикъл foreach

- Използва се за обхождане елементите на масив;
- На всяка итерация елемент от масива array се присвоява елемент на \$value и минава на следващият елемент;

- Синтаксис:

foreach (array as value)

{

code to be executed;

}

Цикъл foreach - пример

```
<html>
<body>
<?php
$arr=array("Barcelona", "Sozopol", "Santorini");
foreach ($arr as $value)
{
    echo "Summer place: " . $value . "!<br />";
}
?>
</body>
</html>
```

Функции в РНР

Функции в PHP

- Функцията е именован блок от код, който може да бъде извикван многократно;
- Функциите могат да се параметризират;
- Функциите могат да връщат стойност;
- В PHP има стотици вградени функции (700+);
- Възможност за създаване на собствени функции.

Създаване на функция (1)

- Всички функции започват със запазената дума `function()`
- Име на функцията – трябва да е възможно да се разбира какво прави функцията само по нейното име. Името на функцията трябва да започва с буква или подчертавка (`_`);
- Добавете отваряща къдрава скоба `"{"` - тялото на функцията следва след отварящата скоба
- Вмъкнете програмният код на функцията
- Добавете затваряща къдрава скоба `"}"` - с което функцията е завършена;

Създаване на функция – пример (2)

```
<html>
<body>
<?php
function say_hello()
{
    echo "Say hello!";
}
say_hello();
?>
</body>
</html>
```

Приложение на функциите – пример (3)

```
<html>
<body>
<?php
function say_hello()
{
    echo "Dear Mr./Mrs. ";
}
echo "Today,<br />";
say_hello();
echo "President <br />";
say_hello();
echo "Professor <br /> ";
say_hello();
echo " Chief Editor <br /> ";
echo " That was it";
echo "Goodbye!";
?>
</body>
</html>
```

Функции с параметър

- Функции без параметри – въпреки, че понякога се налага тяхното използване, по-интересни са функциите с параметри;
- Параметрите са подобни на променливите;
- В скобите се подава име на параметър;

Функция с параметър - пример

```
<html>
<body>
<?php
function say_hello($authority)
{
    echo "Dear Mr./Mrs. " . $authority . "<br />";
}
echo "Today,<br />";
say_hello("President");
say_hello("Professor ");
say_hello("Chief Editor");
echo "<br /> ";
echo " That was it";
echo "Goodbye!";
?>
</body>
</html>
```

Функции с параметри - пример

```
<html>
<body>
<?php
function say_hello($authority, $delimiter)
{
    echo "Dear Mr./Mrs. " . $authority . $delimiter;
}
echo "Today,<br />";
say_hello("President", "<br>");
say_hello("Professor ", "<br>");
say_hello("Chief Editor", "!");
echo "<br /> ";
echo " That was it";
echo "Goodbye!";
?>
</body>
</html>
```

Функции – връщане на стойност

Пример:

```
<html>
<body>
<?php
function sum($a, $b)
{
    $result = $a + $b;
    return $result;
}
echo "5 + 9 = " . sum(5,9) ;
?>
</body>
</html>
```

Вградени функции: date()

- Използва се в PHP да форматира време или дата;
- Синтаксис:
`date(format, timestamp)`

Параметри:

- *format* – задължителен – определя формата на текущото време;
- *timestamp* – незадължителен – задава времето, ако се изпусне – по подразбиране се взема текущото време

Забележка: timestamp (или още Unix Timestamp): времето в секунди от January 1, 1970 at 00:00:00 GMT.

Форматиране на дата

- За форматиране на дата се използва първият аргумент на функцията `date()`, като се използват следните букви:
 - `d` – ден в месеца, с две цифри с водеща нула (01-31);
 - `m` – текущ месец като число с две цифри с водеща нула (01-12)
 - `Y` – текуща година като четири цифри;

Форматиране на дата – пример

```
<?php  
echo date("Y_m_d"); //2014_08_25  
echo "<br />";  
echo date("d.m.Y"); //25.08.2014  
echo "<br />";  
echo date("Y/m/d"); // 2014/08/25  
?>
```

mktime() – създаване на timestamp

- Функцията **mktime()** връща Unix timestamp по зададена дата.
- **Синтаксис:**

mktime(*hour,minute,second,month,day,year,is_dst*)

mktime() - пример

```
<?php
```

```
$tomorrow = mktime(0,0,0,date("m"),date("d")+1,date("Y"));
```

```
echo "Утре ще бъде ". date("d.m.Y", $tomorrow) . "г.;"
```

```
?>
```

Резултат: Утре ще бъде 27.08. 2014г.

Вмъкване на файлове в PHP (1)

- SSI (server-side includes) – се използват за създаване на хедъри, футъри, секции, функции, които ще се използват многократно в различни страници;
- Вмъкването може да стане посредством функциите `include()` или `require()` / `require_once()`;
- Двете функции работят по еднакъв начин с изключение на работата им при възникване на грешка;

Вмъкване на файлове в PHP (2)

- **include()** – функцията се опитва да вмъкне зададеният файл – и ако не успее издава предупреждение, но изпълнението продължава;
- **require()** - функцията се опитва да вмъкне зададеният файл – и ако не успее – генерира фатална грешка и изпълнението на скрипта спира изпълнението си след такава грешка;

include() функция

- Взима текстът от зададеният файл и го копира на мястото във файла, от където е извикана функцията;

- Пример 1:

```
<html>
```

```
<body>
```

```
<h1>My Learning Site</h1>
```

```
<p>Smart information goes here...</p>
```

```
<?php include("site_footer.php"); ?>
```

```
</body>
```

```
</html>
```



site_footer.php

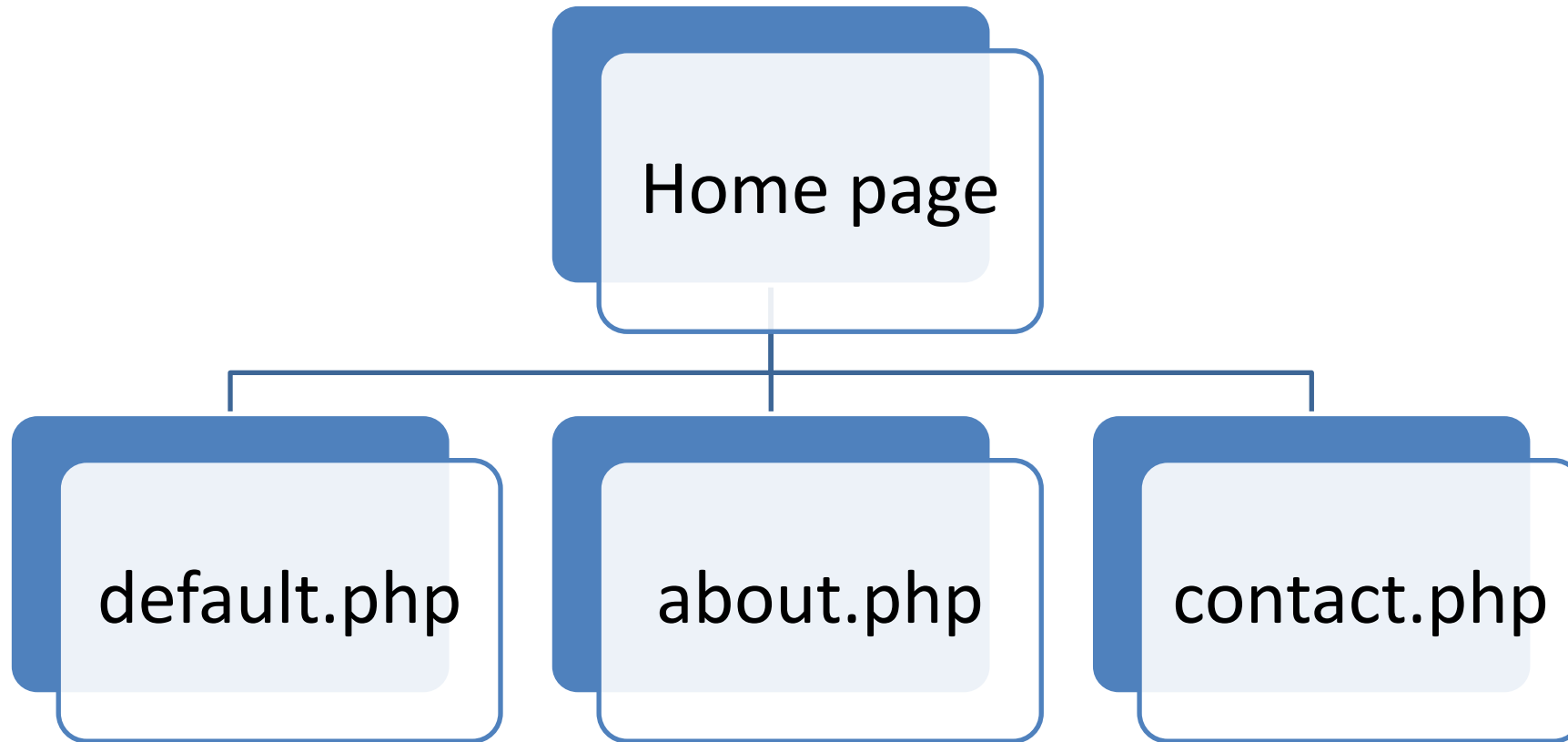
^{Copyright 2014, Milen Petrov}

include() функция – пример 2

Файл: site_menu.php

```
<a href="http://www.w3schools.com/default.php">Home</a> |  
<a href="http://www.w3schools.com/about.php">About Us</a>  
|  
<a href="http://www.w3schools.com/contact.php">Contact  
Us</a>
```

Site map (site_menu.php + site_footer.php)



Упражнение

- Направете собствен сайт с използването на include функцията и отделяне на шаблони *site_header.php*, *site_menu.php*, *site_footer.php*;
- и страници съответно: *default.php*, *about.php*, *contact.php*, *location.php*

Благодаря за вниманието!

Допълнителни слайдове

Вградени функции

Математически функции (1)

- [abs](#) — **Absolute value**
- [acos](#) — Arc cosine
- [acosh](#) — Inverse hyperbolic cosine
- [asin](#) — Arc sine
- [asinh](#) — Inverse hyperbolic sine
- [atan2](#) — Arc tangent of two variables
- [atan](#) — Arc tangent
- [atanh](#) — Inverse hyperbolic tangent
- [base_convert](#) — **Convert a number between arbitrary bases**
- [bindec](#) — **Binary to decimal**
- [ceil](#) — **Round fractions up**
- [cos](#) — Cosine
- [srand](#) — Seed the random number generator
- [tan](#) — Tangent
- [tanh](#) — Hyperbolic tangent

Source: <http://php.net/manual/en/ref.math.php>

Математически функции (2)

- [cosh](#) — Hyperbolic cosine
- [decbin](#) — **Decimal to binary**
- [dechex](#) — **Decimal to hexadecimal**
- [decoct](#) — **Decimal to octal**
- [deg2rad](#) — Converts the number in degrees to the radian equivalent
- [exp](#) — **Calculates the exponent of e**
- [expm1](#) — Returns $\exp(\text{number}) - 1$, computed in a way that is accurate even when the value of number is close to zero
- [floor](#) — **Round fractions down**
- [fmod](#) — **Returns the floating point remainder (modulo) of the division of the arguments**
- [getrandmax](#) — Show largest possible random value
- [hexdec](#) — **Hexadecimal to decimal**
- [hypot](#) — Calculate the length of the hypotenuse of a right-angle triangle
- [is_finite](#) — Finds whether a value is a legal finite number
- [is_infinite](#) — Finds whether a value is infinite
- [is_nan](#) — Finds whether a value is not a number
- [lcg_value](#) — Combined linear congruential generator

Математически функции (3)

- [log10](#) — **Base-10 logarithm**
- [log1p](#) — Returns $\log(1 + \text{number})$, computed in a way that is accurate even when the value of number is close to zero
- [log](#) — **Natural logarithm**
- [max](#) — **Find highest value**
- [min](#) — **Find lowest value**
- [mt_getrandmax](#) — Show largest possible random value
- [mt_rand](#) — **Generate a better random value**
- [mt_srand](#) — **Seed the better random number generator**
- [octdec](#) — **Octal to decimal**
- [pi](#) — Get value of pi
- [pow](#) — **Exponential expression**
- [rad2deg](#) — Converts the radian number to the equivalent number in degrees
- [rand](#) — **Generate a random integer**
- [round](#) — **Rounds a float**
- [sin](#) — Sine
- [sinh](#) — Hyperbolic sine
- [sqrt](#) — **Square root**

Функции със символни низове 1/3

Function	Description
addslashes()	Returns a string with backslashes in front of the specified characters
bin2hex()	Converts a string of ASCII characters to hexadecimal values
chop()	Removes whitespace or other characters from the right end of a string
chr()	Returns a character from a specified ASCII value
chunk_split()	Splits a string into a series of smaller parts
convert_cyr_string()	Converts a string from one Cyrillic character-set to another
convert_uudecode()	Decodes a uuencoded string
convert_uuencode()	Encodes a string using the uuencode algorithm
count_chars()	Returns information about characters used in a string
crc32()	Calculates a 32-bit CRC for a string
crypt()	One-way string encryption (hashing)
echo()	Outputs one or more strings
explode()	Breaks a string into an array
fprintf()	Writes a formatted string to a specified output stream
get_html_translation_table()	Returns the translation table used by htmlspecialchars() and htmlentities()
hebrevc()	Converts Hebrew text to visual text

hebrevc()	Converts Hebrew text to visual text and new lines (\n) into
hex2bin()	Converts a string of hexadecimal values to ASCII characters
html_entity_decode()	Converts HTML entities to characters
htmlentities()	Converts characters to HTML entities
htmlspecialchars_decode()	Converts some predefined HTML entities to characters
htmlspecialchars()	Converts some predefined characters to HTML entities
implode()	Returns a string from the elements of an array
join()	Alias of implode()
lcfirst()	Converts the first character of a string to lowercase
levenshtein()	Returns the Levenshtein distance between two strings
localeconv()	Returns locale numeric and monetary formatting information
ltrim()	Removes whitespace or other characters from the left side of a string
md5()	Calculates the MD5 hash of a string
md5_file()	Calculates the MD5 hash of a file
metaphone()	Calculates the metaphone key of a string
money_format()	Returns a string formatted as a currency string
nl_langinfo()	Returns specific local information
nl2br()	Inserts HTML line breaks in front of each newline in a string

Функции със символни низове 2/3

number_format()	Formats a number with grouped thousands
ord()	Returns the ASCII value of the first character of a string
parse_str()	Parses a query string into variables
print()	Outputs one or more strings
printf()	Outputs a formatted string
quoted_printable_decode()	Converts a quoted-printable string to an 8-bit string
quoted_printable_encode()	Converts an 8-bit string to a quoted printable string
quotemeta()	Quotes meta characters
 rtrim()	Removes whitespace or other characters from the right side of a string
setlocale()	Sets locale information
sha1()	Calculates the SHA-1 hash of a string
sha1_file()	Calculates the SHA-1 hash of a file
similar_text()	Calculates the similarity between two strings
soundex()	Calculates the soundex key of a string
sprintf()	Writes a formatted string to a variable
sscanf()	Parses input from a string according to a format
str_getcsv()	Parses a CSV string into an array
str_ireplace()	Replaces some characters in a string (case-insensitive)
str_pad()	Pads a string to a new length
str_repeat()	Repeats a string a specified number of times
str_replace()	Replaces some characters in a string (case-sensitive)
str_rot13()	Performs the ROT13 encoding on a string

str_shuffle()	Randomly shuffles all characters in a string
str_split()	Splits a string into an array
str_word_count()	Count the number of words in a string
strcasecmp()	Compares two strings (case-insensitive)
strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
strcmp()	Compares two strings (case-sensitive)
strcoll()	Compares two strings (locale based string comparison)
strcspn()	Returns the number of characters found in a string before any part of some specified characters are found
strip_tags()	Strips HTML and PHP tags from a string
stripslashes()	Unquotes a string quoted with addslashes()
stripslashes()	Unquotes a string quoted with addslashes()
strpos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
stristr()	Finds the first occurrence of a string inside another string (case-insensitive)
strlen()	Returns the length of a string
strnatcasecmp()	Compares two strings using a "natural order" algorithm (case-insensitive)
strnatcmp()	Compares two strings using a "natural order" algorithm (case-sensitive)
strncasecmp()	String comparison of the first n characters (case-insensitive)
strncmp()	String comparison of the first n characters (case-sensitive)
strpbrk()	Searches a string for any of a set of characters
strpos()	Returns the position of the first occurrence of a string inside another string (case-sensitive)
strrchr()	Finds the last occurrence of a string inside another string
strrev()	Reverses a string
strripos()	Finds the position of the last occurrence of a string inside another string (case-insensitive)
strrpos()	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strspn()	Returns the number of characters found in a string that contains only characters from a specified charlist
strstr()	Finds the first occurrence of a string inside another string (case-sensitive)

ФУНКЦИИ СЪС СИМВОЛНИ НИЗОВЕ 3/3

<u>strtok()</u>	Splits a string into smaller strings
<u>strtolower()</u>	Converts a string to lowercase letters
<u>strtoupper()</u>	Converts a string to uppercase letters
<u>strtr()</u>	Translates certain characters in a string
<u>substr()</u>	Returns a part of a string
<u>substr_compare()</u>	Compares two strings from a specified start position (binary safe and optionally case-sensitive)
<u>substr_count()</u>	Counts the number of times a substring occurs in a string
<u>substr_replace()</u>	Replaces a part of a string with another string
<u>trim()</u>	Removes whitespace or other characters from both sides of a string
<u>ucfirst()</u>	Converts the first character of a string to uppercase
<u>ucwords()</u>	Converts the first character of each word in a string to uppercase
<u>fprintf()</u>	Writes a formatted string to a specified output stream
<u>vprintf()</u>	Outputs a formatted string
<u>vsprintf()</u>	Writes a formatted string to a variable
<u>wordwrap()</u>	Wraps a string to a given number of characters

Благодаря за вниманието!