**ლექცია 12. ოპერაციები სტრიქონებზე**

ჩვენ უკვე ვიცნობთ სტრინგის - სტრიქონული ტიპის ცვლადებს და ზოგიერთ ოპერაციას, რო­მელიც შესაძლებელია მათზე განხორციელდეს - გამოტანის ოპერაციები echo და print კონ­სტრუქციების საშუალებით. Echo კონსტრუქცია შეძლება ჰქონდეს ჩაწერის ლაკონური სახე, რომლის სინტაქსისია:

**<?=gamosatani\_striqoni ?>**

მაგალითად შევქმნათ სკრიპტი echo კონსტრუქციის საშუალებით -მისი ლაკონური სახით,ავღნიშნოთ რომ ამ შემთხვევაში აღარაა აუცილებელი PHP-ს სკრიპტისათვის დესკრიპტორების ჩაწერა

ლისტინგი 1. echo კონსტრუქცია ლაკონური სახით

<!DOCTYPE html>

<html>

<body>

<?=$name='ბუბა'?> <br>

<font color=red>ეს არის <?=$name?>-ს ავტონაწილების მაღაზია</font>

</body>

</html>

ლისტინგი 1-ის გაშვების შემდეგ მივიღებთ შემდეგ გამოსახულებას:

ბუბა   
 ეს არის ბუბა-ს ავტონაწილების მაღაზია

როგორც ცნობილია print-ს გამოაქვს ერთი სტრიქონული ცვლადი,

print\_r - არ არის სტრიქონული ოპერაცია, რაც ერთი შეხედვით შეიძლება მივიჩნიოთ. მას გამოაქვს ინფორმაცია ცვლადის შესახებ, რომელიც მომხმარებლისთვის გასაგებია, გამოვიყენოთ იგი სტრინგის და მასივისი გამოსატანად, ამასთან გამოსატანი მასივის სტრუქტურის სადემონსტრაციოთ ვისარგებლოთ <pre> ტეგით:

ლისტინგ 2. print\_r

<!DOCTYPE html>

<html>

<body>

<?php

$name='ბუბა';

print\_r($name); echo"<br><pre>";

$a=array('VAN-TIR','საბურავები',100);

print\_r($a);

echo"</pre>";

?>

</body>

</html>

ლისტინგ 2-ის შესრულების შედეგად მივიღებთ:

ბუბა

Array

(

[0] => VAN-TIR

[1] => საბურავები

[2] => 100

)

**printf() ფუნქციას** გამოაქვს ცვლადების მნიშვნელობები ფორმატიზებული სახით, რომელიც წინასწარ მიეთითება. printf() ფუნქციის სინტაქსისია:

printf(format[,$arg1[,$arg2[,...]]])

$arg1,$arg2 და ა. შ. ცვლადების მნიშვნელობები გამოტანილ იქნება მონიტორზე იმ სახით, როგორი მოთხოვნაცაა ჩაწერილი format-ში.

format-ი არის აუცილებელი არგუმენტი printf() ფუნქციისა და როგორც წესი წარმოადგენს სტრინგულ გამოსახულებას, რომელშიც მითითებულია ცვლადების გამოტანის ფორმატები:

* %% - გამოიტანს პროცენტის ნიშნით
* %b - ორობით რიცხვს
* %c - სიმბოლო რომელიც შეესაბამება ASCII კოდის მნიშვნელობას
* %d - ჩაიწერება ათობითი რიცხვი (უარყოფითი, ნული, ან დადებითი)
* %e - სამეცნიერო ჩანაწერი, რომელივ დაბალ რეგისტრს იყენებს (მაგ 2.1e+4)
* %E - სამეცნიერო ჩანაწერი, რომელივ მაღალ რეგისტრს იყენებს (მაგ 2.1E+4)
* %u - უნიშნო ათობითი რიცხვი (ტოლი ან მეტი ნულზე)
* %f - ათობითი რიცხვი მცოცავი მძიმით (ლოკალ;ური local settings aware)
* %F - ათობითი რიცხვი მცოცავი მძიმით (არალოკალური not local settings aware)
* %g - მოკლე of %e and %f
* %G - მოკლე of %E and %f
* %o - რვაობითი რიცხვი ნიშნის გარეშე
* %s - სტრინგის ფორმატი
* %x - თექსმეტობითი რიცხვი (დაბალ რეგისტრში a,b,c,d,e,f)
* %X - თექსმეტობითი რიცხვი (მათალ რეგისტრში A,B,C,D,E,F)

ლისტინგი 3 printf() ფუნქციის გამოყენება ფორმატიზაციისთვის

<!DOCTYPE html>

<html>

<body>

<?php

$number1 = 9;

$str = "ნიკომ";

$fasi=100;

printf("%s ბუბას ავტობაწილების მაღაზიაში შეიძინა %u ცალი საბურავი თითო %.2f ლარად.",$str,$number1,$fasi);

?>

</body>

</html>

ლისტინგი 3-ის გაშვების შედეგი

ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 9 ცალი საბურავი თითო 100.00 ლარად.

%.2f ჩანაწერი აღნიშნავს იმას, რპმ მზიმის შემდეგ იქნება გამოტანილო ოპი პოზიცია, ესეთი გამოტანა მოსახერხებელია ფასის მნიშვნელობის ფორმატიზაციისთვის. % -ის შემდეფ ფორმატიზაციის სიმბოლომდე თუ მითითებულია რიცხვი - იგი განსაზღვრავს გამოსატანი რიცხვის მთლიან სიგრძეს.

printf() აბრუნებს გამოსატანი სტრინგის სიგრძეს.

**ელემენტის ძებნა სტრიქონში - სტრინგში**

იმის გასაგებად, შედის თუ არა მოცემული ქვესტრიქონი - ქვესტრინგი მოცემულ სტრინგში გამოიყენება strpos() ფუნქცია, რომლის სინტაქსისია

strpos($str, $search[, $offset])

strpos() ფუნქცია აბრუნებს პოზიცის, რომელიც მიუთითებს თუ რა პოზიციიდან იწყება პირვე­ლად $search ქვესტრინგი $str-ში, თუ კი მითითებულია არააცილებელი მესამე პარამეტრი $offset, მაშინ ძებნა იწყება $offset პოზიციით. ფუნქცია აბრუნებს False, თუ კი $search ვერ მოიძებნა $str-ში.

შევნიშნოთ, რომ თუ კი $search არ არის სტრინგი, მაშინ იგი გარდაიქმნება მთელ ტიპად და მოიძაბნება მისი - ამ მთელი რიცხვის შესაბამისი სიმბოლო ASCII-კოდში. PHP- ში ნებისმიერი სიმბოლოს ASCII-კოდი შეიძლება ვიპოვოთ ფუნქციით - ord("simbolo")

ლისტინგი 4 strpos() ფუნქცია და მისი ანალოგები

<!DOCTYPE html>

<html>

<body>

<?php

$str = "ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 9 ცალი საბურავი თითო 100.00 ლარად, ადრე ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 3 საბურავი.";

$number1="ნიკომ ბუბას";

$numb2="ავტონაწილების";

$numb3="მაღაზიაში";

$numb4="შეიძინა";

$pos=strpos($str,$number1);

if ($pos !== false) echo "ფრაზა\"$number1\" პირველად დაიწყო $pos პოზიციაში<br> ";

else echo "ფრაზა \"$number1\" არ მოიძებნა<br>";

$pos=strrpos($str,$number1);

if ($pos !== false) echo "ფრაზა\"$number1\" ბოლოს დაიწყო $pos პოზიციაში <br>";

else echo "ფრაზა \"$number1\" არ მოიძებნა<br>";

$pos1=strpos($str,$numb4);

$pos = strpos($str,"მაღაზიაში",$pos1);

if ($pos !== false) echo "ფრაზა \"$numb3\" ფრაზა \"$numb4\" -ს შემდეგ დაიწყო $pos პოზიციაში <br>";

else echo "ფრაზა \"$number1\" არ მოიძებნა<br>";

$pos = strpos($str,"მაღაზიაში");

if ($pos !== false) echo "ფრაზა \"$numb3\" პირველად დაიწყო $pos პოზიციაში <br>";

else echo "ფრაზა \"$number1\" არ მოიძებნა<br>";

$darcha=strstr ($str,$numb2);

echo "ფრაზა\"$numb2\"-დან დაიწყო საწყისი ფრაზის შემდეგი ნაწილი - \"$darcha\"";

?>

</body>

</html>

ლისტინგი 4 -ის შესრულების შემდეგ მივიღებთ:

ფრაზა"ნიკომ ბუბას" პირველად დაიწყო 0 პოზიციაში  
ფრაზა"ნიკომ ბუბას" ბოლოს დაიწყო 212 პოზიციაში   
ფრაზა "მაღაზიაში" ფრაზა "შეიძინა" -ს შემდეგ დაიწყო 284 პოზიციაში   
ფრაზა "მაღაზიაში" პირველად დაიწყო 72 პოზიციაში   
ფრაზა"ავტონაწილების"-დან დაიწყო საწყისი ფრაზის შემდეგი ნაწილი - "ავტონაწილების მაღაზიაში შეიძინა 9 ცალი საბურავი თითო 100.00 ლარად, ადრე ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 3 საბურავი."

ლისტინგ 4-ში მე-13-ე სტრიქონში მოცემულია **strrpos**($str,$number1) ფუნქცია, რომელიც strpos($str,$number1) ფუნქციისგან განსხვავებით; $number1 - სტრინგს $str-ში ბოლოს გამოჩენას აფიქსირებს. ხოლო **strstr** ($str,$numb2) ფუნქცია გამოყოფს ქვესტრინგს $str-დან დაწყებული $numb2 დან. strstr ფუნქცის სინონიმია **strchr()**-ი .

**ფუნქცია substr**

იმისთვის, რომ სტრიქონული ცვლადიდან გამოვყოთ მისი ნაწილი, გამოსაყოფი ნაწილის საწყისი და საბოლოო პოზიციების რიცხვითი მნიშვნელობების მითითებით შესაძლებელია ფუნქცია **substr()**-ის საშუალებით. მისი სინტაქსისია:

**substr($str, $start[, $length])**

ფუნქცია substr()-ი აბრუნებს $str სტრიქონიდან ქვესტრიქონს, რომელიც იწყება $start სიმბო­ლოთი და აქვს $length სიმბოლო. თუ $length მითითებული არაა, ეს ნიშნავს იმას, რომ ქვეს­ტრიქონი შეიცავს ყველა სიმბოლოს დაწყებული $start -დან ბოლომდე. $start -ს გარდა დადები­თი მთელი მნიშვნელობისა შეიძლება მიღოს მთელი უარყოფითი მნიშვნელობაც. ამ შემთხვევა­ში საწყისი პოზიცია განისაზღვრება სტრინგის ბოლოდან დაწყებული. თუ ამასთან ქვესტრინ­გის სიგრძის მაჩვენებელი უარყოფითი მთელი რიცხვია, მაშინ ქვესტრინგის ბოლო სიმბოლო განისაზღვრება საწყისი სტრინგის ბოლოდან მარცხნივ დათვილი სიმბოლო.

ლისტინგ 5 ფუნქცია substr()

<!DOCTYPE html>

<html>

<body>

<?php

$str = "<b>ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 9 ცალი საბურავი თითო 100.00 ლარად</b>, <i>ადრე ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 3 საბურავი.</i>";

$nam1=substr($str,0,204);

echo "$nam1 <br>";

$nam2=substr($str,205);

echo "$nam2 <br>";

$nam3=substr($nam1,3,-4);

echo "$nam3 <br>";

$nam4=substr($nam2,4,-4);

echo "$nam4 <br>";

$nam11 = strip\_tags($nam1);

$nam21 = strip\_tags($nam2);

echo "$nam11 <br> $nam21 <br>";

$sigrZe=strlen($str);

echo "საწყისი ფრაზის სიგრძეა - $sigrZe <br>";

echo $str{125};

?>

</body>

</html>

ლისტინგ 5-ის გაშვებისას მივიღებთ:

**ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 9 ცალი საბურავი თითო 100.00 ლარად**   
*ადრე ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 3 საბურავი.*   
ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 9 ცალი საბურავი თითო 100.00 ლარად   
ადრე ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 3 საბურავი.   
ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 9 ცალი საბურავი თითო 100.00 ლარად   
ადრე ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 3 საბურავი.

საწყისი ფრაზის სიგრძეა - 375   
9

ლისტინგ 5-ში გამოყენებულია **strlen($str)** და **$str{ }** ფუნქციები; strlen($str) აბრუნებს $str სტრინგის სიმბოლოების რაოდენობას, ჩვენს შემთხვევაში 375 სიმბოლოს, რამდენადაც ყოველი სიმბოლო ქართული ანბანიდან საჭირებს 3 კოდირების სიმბოლოს. $str{ } ფუნქციის საშუალე­ბით, სადაც გამოყენებულია ფიგურული ფრჩხილები და რომლის შიგნითაც მითითებულია პოზიციის ნომერი, ვღებულობთ ამ პოზიციაზე არსებულ სიმბოლოს. ქართული ანბანისთვის ეს ფუნქვია არ მუშაობს, რადგანაც როგორც ითქვა ყოველი სიმბოლო ქართული ანბანიდან საჭი­რებს 3 კოდირების სიმბოლოს, და ამდენად მხოლოდ ერთი პოზიცია საკმარისი არ არის, მაგრამ თუ კი ქართულ სტრინგში არის საერთაშორისო სიმბოლოები, ვთქვათ ციფრები, მაშინ შესაძ­ლებელია მათი პოზიციის მითითება და მაშინ $str{ } კონსტრუქცია იმუშავებს, ჩვენს შემთხვე­ვაში $str{125} მოგვცა ციფრი 9, რომელიც საწყისი ფრაზა - სტრინგში 125 პოზიციაზეა. strip\_tags($nam1) ფუნქცია $nam1 ცვლადიდან წაშლის ყველა ტეგს; მაგრამ თუ კი საჭიროა რომელიღაც ტეგის დატოვება, მაშინ ის უნდა მიეთითოს დამატებით პარამეტრად, ზოგადად strip\_tags ფუნქციის სინტაქსისია

strip\_tags ($str [,dasashvebi\_tegebi])

**ფუნქცია str\_replace() და ჩანაცვლების სხვა ფუნქციები**

ძალზე ხშირად ხდება საჭირო ცვლილებების შეტანა სტრინგულ ჩანაწერებში, მონაცემთა ბაზების ელემენტებში მა ს შემდეგ რაც მათში რაღაც ძიება შესრულდა. ამ ოპერაციის შესასრულებლად გამოიყენება სპეციალური ფუნქციები. ერთ-ერთი მათგანია str\_replace()

ფუნქცია, რომლის სინტაქსისია:

**str\_replace(**$search, $replace, $str [,&$count]**)**

str\_replace() ფუნქცია აბრუნებს $str სტრიქონის ასლს, რომელშიც ქვესტრიქონი $search შეცვლილია , $replace-ით. არააუცილებელი პარამეტრ $count-ში შეიძლება ჩაიწეროს შეცვლათა რაოდენობა. ცალკეული სტიქონების ნაცვლად $search და $replace შეიძლება შეიცავდეს სტრიქონთა მასივებს. str\_replace() ფუნქცია რეგისტრმრძნობიარეა, მისი ანალოგი - რეგისტრარამგძნობიარე ფუნქციაა **str\_ireplace**($search,$replace, $str [,&$count]). თუ კი საქმე გვაქვს მასივის ელემენტებში შეცვლასთან, მაშინ ეს შეცვლა ეხება მასივის ყოველ ელემენტას და ვღებულობთ ახალ მასივს.

განვიხილოთ str\_replace() ფუნქციის გამოყენების მაგალითი, ვთქვათ სტრინგში

$str = "ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 9 ცალი საბურავი თითო 100.00 ლარად, ადრე ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 3 საბურავი." შესაცვლელია ”საბურავი” ”აალების სანთელი”-თ ამისთვის დავწეროთ სკრიპტი:

ლისტინგი 6 str\_replace()

<!DOCTYPE html>

<html>

<body>

<?php

$str = "ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 9 ცალი საბურავი, თითო 100.00 ლარად, ადრე ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 3 საბურავი.";

echo "<b>საწყისი ფრაზაა - </b> $str <br>";

$search="საბურავი";

$replace="<font color=red><b>აალების სანთელი</b></font>";

$str1=str\_replace($search,$replace,$str,$raodenoba);

echo "<b>შეცვლილი ფრაზაა - </b> $str1 <br>";

echo "შეიცვალა $raodenoba -ჯერ <br>";

?>

</body>

</html>

ლისტინგ 6-ის გაშვების შემდეგ მივიღებთ:

**საწყისი ფრაზაა -** ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 9 ცალი საბურავი, თითო 100.00 ლარად, ადრე ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 3 საბურავი.   
**შეცვლილი ფრაზაა -** ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 9 ცალი **აალების სანთელი**, თითო 100.00 ლარად, ადრე ნიკომ ბუბას ავტონაწილების მაღაზიაში შეიძინა 3 **აალების სანთელი**.   
შეიცვალა 2 -ჯერ

ა) თუ კი საქმე გვაქვს მასივის ელემენტებში შეცვლასთან, მაშინ ეს შეცვლა ეხება მასივის ყოველ ელემენტს და ვღებულობთ ახალ მასივს. ვღვათ გვაქვს მასივი

salami(”სალამი ბატონო!”,”სალამი ქალბატონო!”,”სალამი მეგობარო!”) და მასში სიტყვა ’სალამი” უნდა შეიცვალოს სიტყვა ”გამარჯობით” , შევქმნათ სკრიპტი

ლისტინგი 7 შეცვლა მასივებში

<!DOCTYPE html>

<html> <body>

<?php

$salami = array("სალამი ბატონო!","სალამი ქალბატონო!","სალამი მეგობარო!");

echo "<b>საწყისი მასივი</b><br>";

print\_r($salami);

echo "<br> უნდა შეიცვალოს სიტყვა \"სალამი\" სიტყვა \"გამარჯობა\"-თი <br>";

$gamarjoba=str\_replace("სალამი","გამარჯობა",$salami,$raodenoba);

echo "<b>შეცვლილი მასივი</b><br>";

print\_r($gamarjoba);

echo "<br> შეიცვალა $raodenoba -ჯერ <br>";

?>

</body> </html>

ლისტინგი 7-ის შესრულებისას მივიღებთ:

**საწყისი მასივი**  
Array ( [0] => სალამი ბატონო! [1] => სალამი ქალბატონო! [2] => სალამი მეგობარო! )   
უნდა შეიცვალოს სიტყვა "სალამი" სიტყვა "გამარჯობა"-თი   
**შეცვლილი მასივი**  
Array ( [0] => გამარჯობა ბატონო! [1] => გამარჯობა ქალბატონო! [2] => გამარჯობა მეგობარო! )   
შეიცვალა 3 -ჯერ

ბ) ვთქვათ ეხლა საძებნი ელემენტი მასივია და ის რითაც უნდა შეიცვალოს - ჩაინაცვლოს არის ასევე მასივი. ამ შემთხვევაში საძებნი მასივის ყოველი ელემენტი შეიცვლება ჩასანაცვლებელი მასივის ელემენტებით საწყის მასივში და სწორედ ეს შეცვლილი ელემენტები შექმნიან მიღებულ მასივს; თუ კი ჩასანაცვლებელი მასივის ელემენტების რაოდენობა ნაკლებია საძებნი მასივის ელემენტებსე, ისინი ცარიელი სტრიქონით ”ივსება”. მაგალითად გვაქვს საწყისი მასივი

$sawyisi("სალამი ბატონო!","სალამი ქალბატონო!","როგორ ხარ მეგობარო!","სალამი ყველას!","Hello!","Hi!");

საძებნი მასივია $saZebni("სალამი","მეგობარო","Hello!")

შესაცვლელი მასივის $Sesacvleli("გამარჯობა","ძმაო")

შევქმნათ სკრიპტი

ლისტინგი 8 მასივების მასივებით ჩანაცვლება

<!DOCTYPE html>

<html> <body>

<?php

$sawyisi = array("სალამი ბატონო!","სალამი ქალბატონო!","როგორ ხარ მეგობარო!","სალამი ყველას!","Hello!","Hi!" );

echo "<b>საწყისი მასივი</b><br>";

print\_r($sawyisi);

$saZebni=array("სალამი","მეგობარო","Hello!");

echo "<br><b>საძებნი მასივი</b><br>";

print\_r($saZebni);

$Sesacvleli=array("გამარჯობა","ძმაო");

echo "<br><b>ჩამნაცვლელი მასივი</b><br>";

print\_r($Sesacvleli);

$akhali\_masivi=str\_replace($saZebni,$Sesacvleli,$sawyisi);

echo "<br><b>შეცვლილი მასივი</b><br>";

print\_r($akhali\_masivi);

?>

</body> </html>

ლისტინგი 8-ის შესრულებისას მივიღებთ:

**საწყისი მასივი**  
Array ( [0] => სალამი ბატონო! [1] => სალამი ქალბატონო! [2] => როგორ ხარ მეგობარო! [3] => სალამი ყველას! [4] => Hello! [5] => Hi! )   
**საძებნი მასივი**  
Array ( [0] => სალამი [1] => მეგობარო [2] => Hello! )   
**ჩამნაცვლელი მასივი**  
Array ( [0] => გამარჯობა [1] => ძმაო )   
**შეცვლილი მასივი**  
Array ( [0] => გამარჯობა ბატონო! [1] => გამარჯობა ქალბატონო! [2] => როგორ ხარ ძმაო! [3] => გამარჯობა ყველას! [4] => [5] => Hi! )

გ) როცა ჩამნაცვლებელი არის სტრინგი, ხოლო საძებნი (ის რომელიც უნდა შეიცვალოს) არის მასივი - მაშინ სტრინგი შეცვლის საძებნი მასივის ელემენტებს, მაგალითად

ლისტინგი 9 სტრინგით იცვლება მასივის ელემენტები

<!DOCTYPE html>

<html> <body>

<?php

$sawyisi = array("სალამი ბატონო!","სალამი ქალბატონო!","როგორ ხარ მეგობარო!","Hi ყველას!","Hello!");

echo "<b>საწყისი მასივი</b><br>";

print\_r($sawyisi);

$saZebni=array("სალამი","მეგობარო","Hello");

echo "<br><b>საძებნი მასივი</b><br>";

print\_r($saZebni);

$Sesacvleli="გამარჯობა ძვირფასო";

echo "<br><b>ჩამნაცვლელი სტრინგი</b><br>";

echo $Sesacvleli;

$akhali\_masivi=str\_replace($saZebni,$Sesacvleli,$sawyisi);

echo "<br><b>შეცვლილი მასივი</b><br>";

print\_r($akhali\_masivi);

?>

</body> </html>

ლისტინგი 9-ის შესრულების შემდეგ მივიღებთ:

**საწყისი მასივი**  
Array ( [0] => სალამი ბატონო! [1] => სალამი ქალბატონო! [2] => როგორ ხარ მეგობარო! [3] => Hi ყველას! [4] => Hello! )   
**საძებნი მასივი**  
Array ( [0] => სალამი [1] => მეგობარო [2] => Hello )   
**ჩამნაცვლელი სტრინგი**  
გამარჯობა ძვირფასო  
**შეცვლილი მასივი**  
Array ( [0] => გამარჯობა ძვირფასო ბატონო! [1] => გამარჯობა ძვირფასო ქალბატონო! [2] => როგორ ხარ გამარჯობა ძვირფასო! [3] => Hi ყველას! [4] => გამარჯობა ძვირფასო! )

**substr\_replace() ფუნქცია**

str\_replace() და substr() ფუნქციების გაერთიანებად შეიძლება მივიჩნიოთ substr\_replace() ფუნქცია, რომელსაც შემდეგი სინტაქსისი გააჩნია

substr\_replace($str, $replacement, $start[, $length]

substr\_replace() ფუნქცია აბრუნებს $str სტრინგის ასლს, რომელშიც ქვესტრინგი, რომელიც იწყება $start სიმბოლოთი და აქვს $length სიგრძე, შეცვლილია $replacement-ით. თუ სიგრძის არგუმენტი $length მითითებული არაა, მაშინ შეცვლა ხდება სტრიქონის ბოლომდე. თუ $start

არგუმენტის მნიშვნელობა დადებითია, ათვლა ხდება სტრიქონის დასაწყისიდან, თუ უარყოფითი - სტრიქონის ბოლოდან; თუ სიგრძის არგუმენტის - $length-ის მნიშვნელობა უარყოფითია, მაშინ ის მიუთითებს $start სტრიქონის ბოლოდან სიმბოლოების რაოდენობას, რომელამდეც უნდა შეიცვალოს ეს ფრაგმენტი.

ლისტინგი 10 substr\_replace()

<!DOCTYPE html>

<html> <body>

<?php

$var = 'ABCDEFGH:/MNRPQR/';

echo "საწყისი: $var<br>";

// ორჯერ იცვლება $var სტრიქონი თავიდან ბოლომდე 'bob'-ით

echo substr\_replace($var, 'bob', 0)."<br>";

echo substr\_replace($var, 'bob', 0, strlen($var))."<br>";

// მოვათავსებთ 'bob'-ს $var-ის თავში

echo substr\_replace($var, 'bob', 0, 0)."<br>";

// ორჯერ 'MNRPQR' $var-ში იცვლება'bob'-იტ

echo substr\_replace($var, 'bob', 10, -1)."<br>";

echo substr\_replace($var, 'bob', -7, -1)."<br>";

// ამოვშლით 'MNRPQR'-ს $var -დან

echo substr\_replace($var, '', 10, -1)."<br>";

?>

</body> </html>

ლისტინგი 10-ის გაშვებით ვღებულობთ:

საწყისი: ABCDEFGH:/MNRPQR/  
bob  
bob  
bobABCDEFGH:/MNRPQR/  
ABCDEFGH:/bob/  
ABCDEFGH:/bob/  
ABCDEFGH://

სტრიქონების დაყოფა და გაერთიანება

**explode()** ფუნქცია ახდენს სტრიქონული ცვლადის მნიშვნელობის დაყოფას რაიმე გამომყოფის მიხედვით, და გააჩნია სინტაქსისი:

explode($delimiter, $str [, $limit])

explode() ფუნქცია აბრუნებს მასივს რომლის ყოველი ელემენტი -სტრიქონი არის, $str - საწყისი მასივის ფრაგმენტი, რომელიც მოთავსებულია გამომყოფების - $delimiter -ს შორის . არააუცილებელი პარამეტრი $limit განსაზღვრავს მასივის ელემენტების რაოდენობას მასივში. დაუნაწილებელი ფრაგმენტი $str-საქყისი სტრიქონისა მოთავსდება ახლადშექმნილი მასივის ბოლო ელემენტში. მაგალითი

ლისტინგი 11 explode()

<!DOCTYPE html>

<html>

<body>

<?php

$str = "პეტრე, სიხარულიძე, pesi@sou.ge";

echo "საწყისი სტრინგი - $str </pre>";

$exp\_str = explode(",", $str);

echo "<pre>";

print\_r ($exp\_str);

echo "</pre>";

?>

</body>

</html>

ლისტინგი 11 -ის გაშვების შემდეგ გვექნება:

საწყისი სტრინგი - პეტრე, სიხარულიძე, pesi@sou.ge

Array

(

[0] => პეტრე

[1] => სიხარულიძე

[2] => pesi@sou.ge

)

ფუნქცია **implode()** არის explode() ფუნქციის საპირისპირო - იგი მასივის ელემენტებს აერთიანებს სტრინგში! მისი სინტაქსისია:

**implode($delimiter, $arr)**

ფუნქცია implode() აბრუნებს სტრიქონს, რომელიც შეიცავს მასივის ელემენტებს, რომლებიც მოცემულია $arr მასივში, ამ ელემენტებს შორის სტრინგში მოთავსებულია მნიშვნელობა, რომელიც განისაზღვრება $delimiter არგუმენტის მნიშვნელობით. მაგალითად

ლისტინგი 12 implode()

<!DOCTYPE html>

<html>

<body>

<?php

$arr = array("პეტრე", "სიხარულიძე", "pesi@sou.ge");

echo "საწყისი მასივი </pre>";

echo "<pre>";

print\_r ($arr);

echo "</pre>";

echo "მასივის ელემენტების გაერთიანებით მიღებული სტრიქონი: <br>".implode(", ", $arr);

?>

</body>

</html>

ლისტინგი 12-ის გაშვებით მივიღებთ:

საწყისი მასივი

Array

(

[0] => პეტრე

[1] => სიხარულიძე

[2] => pesi@sou.ge

)

მასივის ელემენტების გაერთიანებით მიღებული სტრიქონი:   
პეტრე, სიხარულიძე, pesi@sou.ge

**HTML-კოდის გამოტანა**

იმისათვის, რომ გამოვიტანოთ HTML-კოდი, საჭიროა გამოვიყენოთ ფუნქცია htmlspecialchars() რომელსაც აქვს შემდეგი სინტაქსისი:

**htmlspecialchars ($str [, brWyalebis stili [, kodireba]])**

htmlspecialchars() ფუნქციას გადაჰყავს სპეციალური სიმბოლოები, ისეთი როგორიცაა ”<”, ”>”,”&”, """ , "'" HTML-ენის ისეთ არსად, როგორიცაა - შესაბამისად "&lt;", "&gt;", "&amp;", "&quot;", "&#039;"

დამატებითი არგუმენტი brWyalebis stili მიუთითებს, თუ როგორ უნდა მოხდეს ორმაგი და ერთმაგი ბრჭყალების ინტერპრეტაცია - და მას შეძლება ჰქონდეს სამი მნიშვნელობა: ENT\_COMPAT, ENT\_QUOTES, ENT\_NOQUOTES. კონსტანტა ENT\_COMPAT მიუთითებს იმას, რომ ორმაგი ბრჭყალები უნდა იქნენ გადაყვანილი სპეცსიმბოლოებად, ხოლო ერთმაგი უცვლელად რჩებიან; ENT\_QUOTES მიუთითებს იმას, რომ ორივე ტიპის ბრჭყალები უნდა იქნან კონვერტირებული; ENT\_NOQUOTES-ის შემთხვევაში ორივე ბრჭყალები უცვლელი რჩებიან.

დამატებითი არგუმენტი kodireba მიუთითებს კოდირების ტიპს, ის შეიძლება იყოს - UTF-8, ISO-8859-1 ან სხვა. მაგალითად

ლისტინგი 13 htmlspecialchars()

<!DOCTYPE html>

<html><body>

<?php

$new = htmlspecialchars("<a

href='mailto:pesi@sou.ge'>

დავწეროთ წერილი</a>", ENT\_QUOTES);

echo $new;

?>

</body></html>

ლისტინგი 13-ის გაშვების შემდეგ ვღებულობთ

<a href='mailto:pesi@sou.ge'> დავწეროთ წერილი</a>

"<a href='mailto:pesi@sou.ge'> დავწეროთ წერილი</a>" სტრიქონი გარდაიქმნება ამ ჩანაწერად -

&lt;a href=&#039;mailto:au@mail.ru&#039;&gt;დავწეროთ წერილი&lt;/a&gt;

htmlspecialchars() ფუნქცია ახდენს ხშირად გამოყენებადი სიმბოლოების გარდაქმნას, თუ კი არის აუცილებლობა ყველა სიმბოლოების კოდირებისა გამოიყენება ფუნქცია - htmlentities()