**ლექცია N 8. კონსტანტები**

კონსტანტები ცვლადების მსგავსია იმით, რომ შეიცავს მნიშვნელობას, მაგრამ ცვლადებისგან ისინი იმით განსხვავდებიან, რომ, თუ კი კონსტანტა ერთხელ გამოცხადდა, აღარ შეიძლება მისი შეცვლა ან გაუქმება.

კონსტანტა არის იდენტიფიკატორი სკალარული (არამასივის, არაობიექტის) ცვალადისა. კონსტანტას მნიშვნელობა არ შეიძლება შეიცვალოს სკრიპტის მუშაობის დროს. კონსტანტის სწორი იდენტიფიკატორი იწყება ან ლათინური ანბანის ასოთი ან ქვედა ტირეთი. კონსტანტის სახელის დასაწყისად **$** სიმბოლო, განსხვავებით ცვლადებისა, არ გამოიყენება.

ცვლადებისგან განსხვავებით, კონსტანტები არიან გამოცხადებისთანავე სუპერგლობალური - ყველა ნაწილში შესაძლებელია მათი გამოყენება მათი კონკრეტული მნიშვნელობით.

კონსტანტის გამოცხადება ხდება **define()** ფუნქციის საშუალებით, რომელსაც აქვს სინტაქსისი:

**define(name, value [, case\_insensitive])**

**define()** ფუნქციას აქვს სამი პარამეტრი - პირველი -name, განსაზღვრავს კონსტანტის სახელს-იდენტიფიკატორს, მეორე - value, კონსტანტის მნიშვნელობას, ხოლო მესამე -case\_insensitive, რომელიც აუცილებელი არაა, განსაზღვრავს კონსტანტის იდენტიფიკატო­რის არამგრძნობე­ლობას რეგისტრის მიმართ და უნდა მიეთითოს **true** ან **false**. თუ კი იქნება მითითებული **true**, მაშინ კონსტანტის იდენტიფიკატორი არაა მგძნობიარე რეგისტრის მიმართ და ამ შემ­თხვევაში **PHP** კონსტანტებს: „**greeting**” და “**GREETING**”-ს, არ განასხვავებს.

თუ კი იქნება მითითებული false, მაშინ კონსტანტის იდენტიფიკატორი მგძნობიარეა რეგისტრის მიმართ და ამ შემთხვევაში **PHP** კონსტანტები: „**salami**” და “**SALAMI**”, სხვადასხვაა. თუ კი მესამაე პარამეტრი **define()** ფუნქციაში მითითებული არაა, მაშინ იგულისხმება, რომ ჩუმათობის პრინციპით, გვაქვს **false**, ანუ კონსტანტის იდენტიფიკატორი მგძნობიარე რეგისტრის მიმართ, მაგალითი:

ლისტინგი 1 კონსტანტის განსაზღვრა

<!DOCTYPE html>

<html>

<body>

<?php

// რეგისტრარამგძნობიარე კონსტანტის განსაზღვრა

define("SALAMI", "სალამი ღვთისაა!", true);

echo SALAMI;

echo "<br>";

// გამოიტანს კონსტანტის მნიშვნელობას

echo salami;

echo "<br>";

// რეგისტრმგძნობიარე კონსტანტის განსაზღვრა

define("GREETING", "მოგესალმებით, აუდიტორიავ!");

echo GREETING;

echo "<br>";

// არ გამოიტანს კონსტანტის მნიშვნელობას

echo greeting;

?>

<?php

?>

</body>

</html>

ლისტინგი 1-ის გაშვებით მივიღებთ

**სალამი ღვთისაა!  
სალამი ღვთისაა!**

**მოგესალმებით, აუდიტორიავ!  
greeting**

დაპროგრამების კარგ ტონად მიღებულია კონსტანტის იდენტიფიკატორი შეიცავდეს ასოებს, მხოლოდ მაღალ რეგისტრში.

თუ კი არგამოცხადებულ კონსტანტას მივმართავთ მივიღებთ შეცდომას, და თუ თუ კი ჩართულია შეცდომების გამოტანის ფუნქცია - error\_reporting(E\_ALL); ყველა შეცდომების ჩვენება ჩართულია - მაშინ ამ შეცდომას ვიხილავთ მონიტორზე.

ლისტინგი 2 ერთი და იგივე კონსტანტის გამოცხადების მცდელობა

<!DOCTYPE html>

<html>

<body>

<?php

error\_reporting(E\_ALL); // ყველა შეცდომების ჩვენება ჩართულია

define("NUMBER", 1);

define("VALUE", "Hello world!");

echo NUMBER; // 1

echo '<br>';

echo VALUE; // Hello world!

echo '<br>';

echo Number; // Notice: Use of undefined

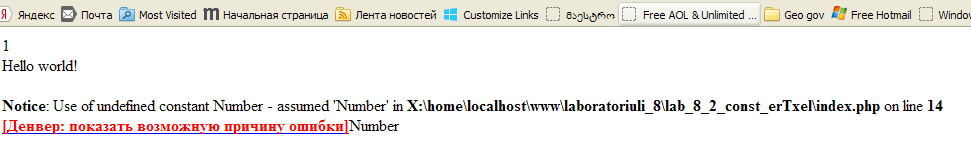
// constant Number - assumed 'Number'

?>

</body>

</html>

ლისტინგი 2-ის გაშვების შემდეგ მივიღებთ



ერთი და იგივე კონსტანტის გამოცხადების მცდელობისას აღინიშნება შეცდომა, და მეორე გამოცხადდება არ შესდგება (define() აბრუნებს false-ს) - კონსტანტას ექნება პირველი გამოცხადების მნიშვნელობა( define() აბრუნებს true-ს). მაგრამ PHP-ში შესაძლებელია, როცა ერთი და იგივე სახელით ორმა კონსტანტან მიიღოს სხვადასხვა მნიშვნელობა. ეს ხდება მხოლოდ იმ შემთხვევაში, როცა define() ფუნქციის საშუალებით გამოცხადებული ერთი და იგივე სახელის მქონე ორი კონსტანტინად, ერთი გამოცხადებუ­ლის მესამე პარამეტრის მნიშვნელობით true, მეორე კი false-თი, მაგალითად

ლისტინგი 3 შეცდომის ჩართვა

<!DOCTYPE html>

<html>

<body>

<?php

error\_reporting(E\_ALL); // ყველა შეცდომების ჩვენება ჩართულია

define("NUMBER", 1);

define("VALUE", "Hello world!");

echo NUMBER; // 1

echo '<br>';

echo VALUE; // Hello world!

echo '<br>';

echo Number; // Notice: Use of undefined

// constant Number - assumed 'Number'

echo '<br>';

**if(define("WASHA", "გაუმარჯოს მშვიდობას!"))**

**echo "კონსტანტა WASHA-მ მიიღო მნიშვნელობა \"გაუმარჯოს მშვიდობას!\"<br>";**

**if(define("WASHA", 1, TRUE))**

**echo "კონსტანტა WASHA-მ მიიღო მნიშვნელობა 1<br>";**

**echo WASHA;**

**echo '<br>';**

**echo Washa;**

?>

</body>

</html>

ლისტინგი 3 -ის გაშვების შემდეგ მივიღებთ

1  
Hello world!  
  
**Notice**: Use of undefined constant Number - assumed 'Number' in **X:\home\localhost\www\laboratoriuli\_8\lab\_8\_3\_Secdoma\index.php** on line **14**  
[**[Денвер: показать возможную причину ошибки]**](http://localhost/laboratoriuli_8/lab_8_3_Secdoma/)Number  
კონსტანტა WASHA-მ მიიღო მნიშვნელობა "გაუმარჯოს მშვიდობას!"  
კონსტანტა WASHA-მ მიიღო მნიშვნელობა 1  
გაუმარჯოს მშვიდობას!  
1

შევნიშნოთ, რომ ის კონსტანტა, გამოცხადებული, როგორც მგრძნობიარე რეგისტრის მიერ გადაფარავს რეგისტრის მიმართ არამგრძნობიარე კონსტანტას, ანუ **echo WASHA;** კონსტრუქ­ციით გამოისახება მნიშვნელობა "გაუმარჯოს მშვიდობას", რომელიც განსაზღვრული იყოროგორც მგრძნობიარე რეგისტრის და არა იმავე სახელის მქონე მეორე კონსტანტას მნიშვნელობა

მომხმარებლის მიერ განსაზღვრულ კონსტანტებთან ერთად არსებობს PHP-ის მიერ განსაზღვრული ძალიან ბევრი საკუთარი კონსტანტა. მათი დათავლიერება-გაცნობა შეიძლება მარტივად phpinfo() ფუნქციის გამოძახებით -

phpinfo();

phpinfo() ფუნქციას სხვა სასარგებლო ინფორმაციაც გამოაქვს, მაგრამ, თუ კი გვსურს მივიღოთ მთელი სია მხოლოდ გამოყენებული კონსტანტებისა, მაშინ უნდა ვისარგებლოთ ფუნქციით get\_defined\_constants().

თუ გვინდა გავიგოთ არის თუ არა რომელიმე კონსტანტა, მისი სახელი უნდა ჩავსვათ ფუნქციის defined() მრგვალ ფრჩხილებში, ანუ პარამეტრი უნდა იყოს კონსტანტის სახელი. თუ კი ასეთი კონსტანტა არსებობს, მაშინ ფუნქცია defined() აბრუნებს true-მნიშვნელობას, false -ს თუ კი ასეთი პარამეტრი არ არსებობს. ხოლო თუ გვინდა კონსტანტის მნიშვნელობაც მივიღოთ, მაშინ უნდა გამოვიყენოთ ფუნქცია constants(), მის ფრჩხილებში ჩაისმება კონსტანტას სახელი, რომლის გამოიტანაც გვსურს. ფუნქცია constants()-ის პარამეტრად შეიძლება იყოს ცვლადი, რომლის მნიშვნელობაცაა კონსტანტას სახელი - ამ შემთხვევაში constants() დააბრუნებს კონსტანტის მნიშვნელობას. ამ ფუნქციის გამოყენება შესაძლებელია იყოს ეფექტური, როცა გვსურდ დინამიური კონსტანტას გამოტანა. მაგალითად:

ლისტინგი 4 დინამიური კონსტანტა

<!DOCTYPE html>

<html>

<body>

<?php

// ვახდენთ 1 -დან 10 -მდე შემთხვევითი მთელი რიცხვის გენერირებას

$index = rand(1, 10);

echo "\$index=$index <br>";

// ვახდენთ კონსტანტის სახელის ფორმირებას

$name = "VALUE{$index}";

echo "\$name=$name";

echo '<br>';

// განვსაზღვრავთ კონსტანტას დინამიური სახელით

define($name, 1); // $name ცვლადის მნიშვნელობა - კონსტანტის სახელად მიიღება

// ვღებულობთ კონსტანტას მნიშვნელობას

echo 'constant($name)='.constant($name);

?>

</body>

</html>

ლისტინგ 4-ის ერთერთი რეალიზაციის შედეგად ვღებულობთ;

**$index=5   
$name=VALUE5  
constant($name)=1**

**წინასწარგანსაზღვრული კონსტანტები**

მომხმარებლის მიერ შექმნილი კონსტანტების გარდა არის ისეთი კონატანტები, რომლებიც PHP-ს მიერ წინასწარაა განსაზღვრული. ასეთ კონსტანრებს უკვე შევხვედრივართ, მაგალი­თად - ლოგიკური კონსტანტები - TRUE, true, FALSE, false.

ჩამოვთვალოთ ქვემოთ მოყვანილ ცხრილში წინასწარგანსაზღვრული კონსტანტები.

ცხრილი 1 წინასწარგანსაზღვრული კონსტანტები.

|  |  |
| --- | --- |
| კონსტანტა | აღწერა |
| \_\_LINE\_\_ | ფაილში მიმდინარე სტრიქონი |
| \_\_FILE\_\_ | მიმდინარე ფაილის სრული გზა და სახელი |
| \_\_FUNCTION\_\_ | ფუნქციის სახელი |
| \_\_CLASS\_\_ | კლასის სახელი |
| \_\_METHOD\_\_ | კლასის მეთოდის სახელი |

ლისტინგ 4-ში დავამატოთ სკრიპტის ნაწილში კოდი

**echo "ფაილის სახელი ".\_\_FILE\_\_."<br>";**

**echo "სტრიქონი ".\_\_LINE\_\_;**

და ახალ მიღებულ ლისტინგ 5-ის გაშვების შემდეგ მივიღებთ:

$index=4   
$name=VALUE4  
constant($name)=1  
**ფაილის სახელი X:\home\localhost\www\laboratoriuli\_8\lab\_8\_5\_winasw\_const\index.php  
სტრიქონი 21**

ფაილის სახელი მივიღეთ ლოკალურ სერვერზე მუშაობის შედეგად, დაშორებული სერვერის შემთხვევაში გვექნება:

ავტონაწილების მაღაზიისთვის ფორმების დამუშავების პროგრამაში ჩვენ გამოვიყენებთ კონსტანტებს, მაგრამ მანამდე გავეცნოთ კიდევ ერთხელ ცვლადების მოქმედების არეებს და ცვლადებზე მოქმედებებს.

**ცვლადების მოქმედების არეები**

ლექცია 4-ში ჩვენ უკვე განვიხილეთ ზოგიერთი ცვლადის მოქმედების არე და ვნახეთ, რომ გლობალური, ლოკალური და სტატიკური ტიპის ცვლადებს თავიანთო მოქმედების - ან მხედველობის გარკვეული არეები გააჩნია. მათ გარდა PHP-სი არიან სიდიდეები - ცვლადები, მასივები, კონსტანტები და სხვა, რომელთა თვალსაწიერის არე შეზღუდული არაა. ასეთ სიდიდეებს სუპერგლობალური ეწოდება. კონსტანტებს სუპერგლობალური თვისება გააჩნია.

ცხრილ 2-ში მოცემულია სუპერგლობალური ცვლადების ნუსხა

ცხრილი 2 სუპერგლობალური ცვლადების ნუსხა

|  |  |
| --- | --- |
| სუპერგლობალური ცვლადები | აღწერა |
| $GLOBALS | ყველა გლობალური ცვლადების მასივი. საკვანძო სიტყვის global –ის მსაგავსად ამ მასივშიც კონკრეტული გლობალური მონაცემის წვდომისთვის მაგალითად: $GLOBALS[’myvariable’] |
| $\_SERVER | სერერის გარემოს ცვლადების მასივი |
| $\_GET | მონაცემთა მასივი გადაცემული GET მეთოდით |
| $\_POST | მონაცემთა მასივი გადაცემული POST მეთოდით |
| $\_COOKIE | Cookie-ის ცვლადების მასივი |
| $\_FILE | ფაილის ჩატვირთვასთან დაკავშირებული ცვლადების მასივსი |
| $\_ENV | გარემოს ცვალადები |
| $\_REQUEST | მომხმარებლის შეყვანის მასივი, მათ შორის $\_GET, $\_POST, $\_COOKIE, $\_FILE |
| S\_SESSION | სეანსის ცვლადების მასივი |

**PHP -ს ოპერაციები**

***არითმეტიკული ოპერაციები***

ცხრილი 3

|  |  |  |  |
| --- | --- | --- | --- |
| **ოპერაცია** | **დასახელება** | **ნიმუში** | **შედეგი** |
| + | შეკრება | $x + $y | $x -სა და $y-ის ჯამი |
| - | გამოკლება | $x - $y | $x -სა და $y-ის სხვაობა |
| \* | გამრავლება | $x \* $y | $x -სა და $y-ის ნამრავლი |
| / | გაყოფა | $x / $y | $x -სა და $y-ის განაყოფი |
| % | მოდულით | $x % $y | $x -ის $y-ზე გაყოფით მიღებული ნაშთი |
| \*\* | ახარისხება | $x \*\* $y | $x -ის $y ხარისხში აყვანის შედეგი (შემოტანილია PHP 5.6-ში) |

ლისტინგი 6 არითმეტიკული ოპერაციების შესრულება ერთი და იმავე ცვლადებისთვის

<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
$x=13;   
$y=7;  
  
echo ($x + $y);  
echo "<br>";  
echo ($x - $y);  
echo "<br>";  
echo ($x \* $y);  
echo "<br>";  
echo ($x / $y);   
echo "<br>";  
echo ($x % $y);   
?>     
  
</body>  
</html>

ლისტინგ 6-ის გაშვების შემდეგ მივიღებთ შედეგს, რომელსაც კომპაქტურობისთვის ერთ სტრიქონში გამოვიტანთ:

20 6 91 1.8571428571429 6

***მინიჭების ოპერაციები***

მინიჭების ოპერაცია გამოიყენება ცვლადისთვის მნიშვნელობის მისანიჭებლად.

მინიჭების ზირითადი ოპერაციაა ”**=**”, ეს ნიშნავს იმას, რომ მარცხენა ოპერანდი მიიღებს მნიშვნელობას, რომელიც მიიღება მარჯვნივ მოთავსებული გამოსახულებით.

ცხრილი 4 მინიჭება

|  |  |  |
| --- | --- | --- |
| **მინიჭება** | **იგივეა რაც** | **აღწერა** |
| x = y | x = y | მარცხენა ოპერანდი მიიღებს მნიშვნელობას, რომელიც მიიღება მარჯვნივ მოთავსებული გამოსახულებით. |
| x += y | x = x + y | შეკრება |
| x -= y | x = x - y | გამოკლება |
| x \*= y | x = x \* y | გამრავლება |
| x /= y | x = x / y | გაყოფა |
| x %= y | x = x % y | მოდულით |

ლისტინგ 7 მინიჭების ოპერაციის გამოყენება

<!DOCTYPE html>

<html>

<body>

<?php

$x=10;

echo "\$x=$x";

echo "<br>";

$y=20;

$y += 100;

echo "\$y=$y";

echo "<br>";

$z=50;

$z -= 25;

echo "\$z=$z";

echo "<br>";

$i=5;

$i \*= 6;

echo "\$i=$i";

echo "<br>";

$j=10;

$j /= 5;

echo "\$j=$j";

echo "<br>";

$k=15;

$k %= 4;

echo "\$k=$k";

?>

</body>

</html>

ლისტინდ 7-ის შესრულების შედეგად მივიღებთ:

$x=10 $y=120 $z=25 $i=30 $j=2 $k=3

***სტრიქონული (სტრინგის) ოპერაციები***

|  |  |  |  |
| --- | --- | --- | --- |
| **ოპერაცია** | **დასახელება** | **ნიმუში** | **შედეგი** |
| **.** | კონკატენაცია | $txt1 = "Hello" $txt2 = $txt1 . " world!" | ეხლა$txt2 შეიცავს "Hello world!" |
| **.=** | მინიჭების კონკატენაცია | $txt1 = "Hello" $txt1 .= " world!" | ეხლა $txt1შეიცავს "Hello world!" |

ლისტინგი 8 სრტინგის ოპერაციები

!DOCTYPE html>

<html>

<body>

<?php

$a = "Hello";

$b = $a . " world!";

echo $b; // გამოგვყავს Hello world!

echo "<br>";

$x="Hello";

$x .= " world!";

echo $x; // გამოგვყავს Hello world!

?>

</body>

</html>

ლისტინგი 8-ს გაშვების შედეგად მივიღებთ:

**Hello world!  
Hello world!**

ინკრიმენტ/დეკრემენტის ოპერაციები

ინკრიმენტი ნიშნავს გაზრდას, ჩვენს შემთხვევაში ერთი ერთეულით გაზრდას, რადგანაც ასეთნაირად ცვლადის მნიშვნელობის შეცვლა ხშირად ხდება. იგივე ითქმის დეკრემენტ (შემცირების) ოპერაციაზე

ცხრილი ინკრიმენტი/დეკრემენტი

|  |  |  |
| --- | --- | --- |
| ოპერაცია | დასახელება | აღწერა |
| ++$x | პრე-ინკრიმენტი/ პრეფიქსული ინკრიმენტი | გაიზრდება $x ცვლადი ერთი ერთეულით და ეს მნიშვნელობა ბრუნდება $x -ში |
| $x++ | პოსტ ინკრიმენტი/ სუფიქსური ინკრიმენტი | მნიშვნელობა ბრუნდება $x -ში და ამის შემდეგ გაიზრდება $x ცვლადი ერთი ერთეულით |
| --$x | პრე-დეკრემენტი / პრეფიქსული დეკრემენტი | შემცირდება $x ცვლადი ერთი ერთეულით და ეს მნიშვნელობა ბრუნდება $x -ში |
| $x++ | პოსტ დეკრემენტი / სუფიქსური დეკრემენტი | მნიშვნელობა ბრუნდება $x -ში და ამის შემდეგ შემცირდება $x ცვლადი ერთი ერთეულით |

ქვემოთ მოყვანილი მაგალითი ინკრიმენტ/დეკრემენტის გამოყენების ნიუანსების დემონსტრაციას ახდენს:

ლისტინგი 9. ინკრიმენტ/დეკრემენტი

<!DOCTYPE html>

<html>

<body>

<?php

$x=10;

echo "\$x=$x <br>";

echo '++$x -ის ბეჭვდისას ვღებულობთ '.++$x;

echo "<br>"; echo "<br>";

$y=10;

echo "\$y=$y <br>";

echo '$y++ -ის ბეჭვდისას ვღებულობთ '.$y++;

echo "<br>"; echo "<br>";

$z=5;

echo "\$z=$z <br>";

echo '--$z -ის ბეჭვდისას ვღებულობთ '.--$z;

echo "<br>"; echo "<br>";

$i=5;

echo "\$i=$i <br>";

echo '$i-- -ის ბეჭვდისას ვღებულობთ '.$i--;

?>

</body>

</html>

ლისტინგი 9 -ის გაშვებისას ვღებულობთ

$x=10   
++$x -ის ბეჭვდისას ვღებულობთ 11  
  
$y=10   
$y++ -ის ბეჭვდისას ვღებულობთ 10  
  
$z=5   
--$z -ის ბეჭვდისას ვღებულობთ 4  
  
$i=5   
$i-- -ის ბეჭვდისას ვცღებულობთ 5

**შედარების ოპერაციები**

შედარების ოპერაციები გამოიყენებს ორი ცვლადის (რიცხვითი ან სტრიქონულის) შედარებისთვის

ცხრილი შედარების ოპერაციები

|  |  |  |  |
| --- | --- | --- | --- |
| ოპერაცია | დასახელება | ნიმუში | შედეგი |
| == | ტოლობა | $x == $y | ჭეშმარიტია თუ $x $y-ის ტოლია |
| === | იდენტური | $x == =$y | ჭეშმარიტია თუ $x $y-ის ტოლია და ერთნაირი ტიპის არიან |
| != | არაა ტოლი | $x != $y | ჭეშმარიტია თუ $x $y-ის ტოლი არაა |
| <> | არაა ტოლი | X<> $y | ჭეშმარიტია თუ $x $y-ის ტოლი არაა |
| !== | არაა იდენტური | $x !== $y | ჭეშმარიტია თუ $x $y-ის ტოლია არაა ან ერთნაირი ტიპის არ არიან |
| > | მეტია ვიდრე | X> $y | ჭეშმარიტია თუ $x $y-ზე მეტია |
| < | ნაკლებია ვიდრე | X< $y | ჭეშმარიტია თუ $x $y-ზე ნაკლებია |
| >= | მეტია ან ტოლია ვიდრე | X>= $y | ჭეშმარიტია თუ $x $y-ზე მეტია ან ტოლი |
| <= | ნაკლებია ან ტოლია ვიდრე | X<= $y | ჭეშმარიტია თუ $x $y-ზე ნაკლებია ან ტოლი |

ქვემოთ მოყვანილი მაგალითი გვიჩვენებს ზოგიერთი შედარების ოპერაციას გამოყენების შედეგებს

ლისტინგ 10 შედარების ოპერაციები

<!DOCTYPE html>

<html>

<body>

<?php

$x=100;

$y="100";

echo '$x=100 და $y=\'100\'<br>';

var\_dump($x == $y); // აბრუნებს ჭეშმარიტს იმიტომ, რომ ცვლადები არიან ტოლი

echo ' //$x == $y აბრუნებს ჭეშმარიტს იმიტომ, რომ ცვლადები არიან ტოლი <br>';

var\_dump($x === $y); // აბრუნებს მცდარს იმიტომ, რომ ცვლადების ტიპები არ არიან ტოლი

echo ' //$x === $y აბრუნებს მცდარს იმიტომ, რომ ცვლადების ტიპები არ არიან ტოლი<br>';

var\_dump($x != $y); // აბრუნებს მცდარს იმიტომ, რომ ცვლადები არიან ტოლი

echo ' //$x != $y აბრუნებს მცდარს იმიტომ, რომ ცვლადები არიან ტოლი<br>';

var\_dump($x !== $y); // აბრუნებს ჭეშმარიტს იმიტომ, რომ ცვლადების ტიპი არაა ტოლი

echo ' //$x !== $y აბრუნებს ჭეშმარიტს იმიტომ, რომ ცვლადების ტიპი არაა ტოლი<br><br>';

$a=50;

$b=90;

echo ' //$a=50 და $b=90 <br>';

var\_dump($a > $b);

echo ' //$a > $b ბრუნებს მცდარს, რადგანაც პირველი ცვლადი არაა მეორეზე მეტი <br>';

var\_dump($a < $b);

echo ' //$a < $b აბრუნებს ჭეშმარიტს, რადგანაც პირველი ცვლადი ნაკლებია მეორეზე <br>';

?>

</body>

</html>

ლისტინგ 10-ის გაშვებით მივიღებთ

$x=100 და $y='100'  
bool(true) //$x == $y აბრუნებს ჭეშმარიტს იმიტომ, რომ ცვლადები არიან ტოლი   
bool(false) //$x === $y აბრუნებს მცდარს იმიტომ, რომ ცვლადების ტიპები არ არიან ტოლი  
bool(false) //$x != $y აბრუნებს მცდარს იმიტომ, რომ ცვლადები არიან ტოლი  
bool(true) //$x !== $y აბრუნებს ჭეშმარიტს იმიტომ, რომ ცვლადების ტიპი არაა ტოლი  
  
//$a=50 და $b=90   
bool(false) //$a > $b ბრუნებს მცდარს, რადგანაც პირველი ცვლადი არაა მეორეზე მეტი   
bool(true) //$a < $b აბრუნებს ჭეშმარიტს, რადგანაც პირველი ცვლადი ნაკლებია მეორეზე

**ლოგიკური ოპერაციები**

|  |  |  |  |
| --- | --- | --- | --- |
| ოპერაცია | დასახელება | ნიმუში | შედეგი |
| and | და | $x and $y | ჭეშმარიტია, თუ $x და $y ჭეშმარიტია |
| or | ან | $x or $y | ჭეშმარიტია, თუ $x ან $y ჭეშმარიტია |
| xor | ხორ | $x xor $y | ჭეშმარიტია, თუ $x ან $y ერთერთი ჭეშმარიტია, მაგრამ არა ორივე ერთდროულად |
| && | და | $x && $y | ჭეშმარიტია, თუ $x და $y ჭეშმარიტია |
| || | ან | $x || $y | ჭეშმარიტია, თუ $x ან $y ჭეშმარიტია |
| ! | არა | !x | ჭეშმარიტია, თუ $x არაა ჭეშმარიტი |

ქვემოთ მოყვანილი მაგალითი გვიჩვენებს ზოგიერთი ლოგიკური ოპერაციების გამოყენების შედეგებს

ლისტინგ 11 ლოგიკური ოპერაციები

<!DOCTYPE html>

<html>

<body>

<?php

$x = true;

$y = false;

$z = true;

echo "<br>";

echo 'true and false - ';

var\_dump($x and $y);

echo "<br>";

echo 'true and true - ';

var\_dump($x and $z);

echo "<br>";

echo '!false - ';

var\_dump(!$y);

?>

</body>

</html>

ლისტინდ 11-ის გაშვების შემდეგ მივიღებთ

true and false - bool(false)   
true and true - bool(true)   
!false - bool(true)

**მასივის ოპერაციები**

მასივის ოპერაციები გამოიყენება მასივების შესადარებლად

|  |  |  |  |
| --- | --- | --- | --- |
| ოპერაცია | დასახელება | ნიმუში | შედეგი |
| + | დამატება | $x + $y | გააერთიანებს $x და $y-ს(ისე, რომ გა­საღებები არ გადაიწერება)ჭეშმარიტია |
| == | ტოლია | $x == $y | ჭეშმარიტია, თუ $x ან $y -ს აქვს ერთი და იგივე წყვილები გასაღები/ მნიშვნე­ლობა |
| === | იდენტურია | $x === $y | ჭეშმარიტია, თუ $x ან $y -ს აქვს ერთი და იგივე წყვილები გასაღები/მნიშვნე­ლობა და იგივე ტიპები აქვთ |
| != | არატოლი | $x!= $y | ჭეშმარიტია, თუ $x და $y არაა ტოლე­ბი |
| <> | არატოლი | $x<>$y | ჭეშმარიტია, თუ $x და $y ტოლები არ არიან |
| !== | არაიდენტური | $x!==$y | ჭეშმარიტია, თუ $x არაა $y-ის იდენტური |

ქვემოთ მოყვანილი მაგალითი გვიჩვენებს ზოგიერთი შედარების ოპერაციას გამოყენების შედეგებს

ლისტინგი 12 მასივების შედარება

<!DOCTYPE html>

<html>

<body>

<?php

$x = array("a" => "red", "b" => "green");

$y = array("c" => "blue", "d" => "yellow");

$z = $x + $y; // $x -ის და $y მასივების გაერთიანება

var\_dump($z);

echo "<br>";

var\_dump($x == $y);

echo "<br>";

var\_dump($x === $y);

echo "<br>";

var\_dump($x != $y);

echo "<br>";

var\_dump($x <> $y);

echo "<br>";

var\_dump($x !== $y);

?>

</body>

</html>

ლისტინგ 12 გაშვებით მივიღებთ

array(4) { ["a"]=> string(3) "red" ["b"]=> string(5) "green" ["c"]=> string(4) "blue" ["d"]=> string(6) "yellow" }   
bool(false)   
bool(false)   
bool(true)   
bool(true)   
bool(true)

თუ კი მასივებში არის ერთი და იგივე გასაღები, მაშინ რჩება პირველი მასივიდან მნიშვნელობა, ხოლო მეორე უგულველყოფილ იქნება, მაგალითად, ლისტინგ 12-ში თუ შევცვლით მეექვსე სტრიქონს შემდეგით:

$y = array("b" => "blue", "d" => "yellow");

მაშინ გაერთიანების შედეგი იქნება

array(3) { ["a"]=> string(3) "red" ["b"]=> string(5) "green" ["d"]=> string(6) "yellow" }

ეხლა დავიფრინდეთ ავრონაწილების ნაღაზიისთვის ფორმის დამუშავებას და გამოვთვალოთ შეკვეთის ჯამური თანხა და გადასახადის რაოდენობა.

დავუმატოთ ლისტინგ 9-ში ლექცია 7-ში სპრიპტის ნაწილში კოდი:

$totalamount = $tireqty \* TIREPRICE

+ $oilqty \* OILPRICE

+ $sparkqty \* SPARKPRICE;

echo "ჯამში: $".number\_format($totalamount,2)."<br />";

$taxrate = 0.10; // ადგილობრივი მოსაკრებელი გაყიდვებისგან შეადგენს 10%

$totalamount = $totalamount \* (1 + $taxrate);

echo "მთლიანად, გაყიდვებისგან ბაჟის ჩართვით: $".number\_format($totalamount,2)."<br />";

მივიღებთ ლექცია 7-ის ლისტინგ 6-ს, ამდენა ფორმის შევსებისა და მისი გაშვებისას, რომელსაც დასჭირდება ფორმის ცვლადების დამუშავება, ვღებულობთ

# ავტონაწილები ბუბასგან

## შეკვეთის შედეგი

შეკვეთა დამუშავდა18:10, 2nd November 2014

შეკვეთილია:

4 საბურავი  
8 ბოთლი ზეთი  
6 აალების სანთელი  
  
შეკვეთილი საქონელი: 18  
ჯამში: $504.00  
მთლიანად, გაყიდვებისგან გადასახადის ჩათვლით: $554.40

ფორმის ცვლადების დამუშავება მოცემულია

ლისტინგ 13-ში, შეკვეთის დამუშავება

<html>

<head>

<title>ავტონაწილები ბუბასგან — შეკვეთის შედეგები</title>

</head>

<?php

// ცვლადების მოკლე სახელების შექმნა

$tireqty = $\_POST['tireqty'];

$oilqty = $\_POST['oilqty'];

$sparkqty = $\_POST['sparkqty'];

?>

<body>

<h1>ავტონაწილები ბუბასგან</h1>

<h2>შეკვეთის შედეგი</h2>

<?php

echo "<p>შეკვეთა დამუშავდა".date('H:i, jS F Y')."</p>";

echo '<br><p>შეკვეთილია: </p>';

echo $tireqty.' საბურავი<br />';

echo $oilqty.' ბოთლი ზეთი<br />';

echo $sparkqty.' აალების სანთელი<br />';

echo '<br>';

$totalqty = 0;

$totalqty = $tireqty + $oilqty + $sparkqty;

echo "შეკვეთილი საქონელი: ".$totalqty."<br />";

$totalamount = 0.00;

define('TIREPRICE', 100);

define('OILPRICE', 10);

define('SPARKPRICE', 4);

$totalamount = $tireqty \* TIREPRICE

+ $oilqty \* OILPRICE

+ $sparkqty \* SPARKPRICE;

echo "ჯამში: $".number\_format($totalamount,2)."<br />";

$taxrate = 0.10; // ადგილობრივი მოსაკრებელი გაყიდვებისგან შეადგენს 10%

$totalamount = $totalamount \* (1 + $taxrate);

echo "მთლიანად, გაყიდვებისგან გადასახადის ჩათვლით: $".number\_format($totalamount,2)."<br />";

?>

</body>

</html>

ფორმის მონაცემებისას ჩვენ გამოვიყენეთ არითმეტიკული ოპერაციები: შეკრება (+), გამრავლება (\*), სტრიქონების კონკატენაცია (.) და მონაცემები ფგამოვიყვანეთ ფორმატში რომელიც მითითებულია ფუნქციაში number\_format() –საბოლოო ჯამი წარმოდგენილ იქნა სტრიქონის სახით გამოყოფილი ორი ათწილადი ნაწილით, ეს ფუნქცია PHP- მათემატიკური ფუნქციების ბიბლიოთეკაში შედის.

იმისთვის, რომ გასაგები გახდეს, საბოლოო ჯამოის გამოთვლისა, რატომ მოხდა ჯერ გამრავლების შესრულება და მერე ეკრების, ჩვენ უნდა გავეცნოთ ოპერაციათა პრიორიტეტულობას და ასოციურობას

პრიორიტეტი და ასოციურობა

ოპერაციებს გააჩნიათ შესრულების პრიორიტეტულობა - შესრულების რიგითობა. გარდა ამისა ოპერაციას აქვს ასოციურობის მახასიათებელი, რაც მიუთითებს იმას, თუ რა თანამიმდევრობით უნდა შესრულდეს ერთი და იგივე პრიორიტეტების მქონე ოპერაციები . ზოგადად თანამიმდევრობა სრულდება მარჯვნიდან-მარცხნივ, ან პირიქით, მარცხნიდან - მარჯვნივ. მოვიყვანოთ ცხრილი პრიორიტეტულობისა და ასოციარობის მითითებით, ამასთან ოპერაციები დავალაგოთ ზემოდან ქვემოთ, მათი პრიორიტეტობის ზრდით:

ცხრილი ოპერაციათა პრიოპიტეტულობა

|  |  |  |
| --- | --- | --- |
| პრიორი­­­-  ტეტის № | ასოციურობა | ოპერაციები |
| 1 | მარცხნიდან-მარჯვნივ | **,** |
| 2 | მარცხნიდან-მარჯვნივ | **or** |
| 3 | მარცხნიდან-მარჯვნივ | **xor** |
| 4 | მარცხნიდან-მარჯვნივ | **and** |
| 5 | მარჯვნიდან-მარცხნივ | **print** |
| 6 | მარცხნიდან-მარჯვნივ | **= += -= \*= /= .= %= &= |= ^= ~= <<= >>=** |
| 7 | მარცხნიდან-მარჯვნივ | **? :** |
| 8 | მარცხნიდან-მარჯვნივ | **| |** |
| 9 | მარცხნიდან-მარჯვნივ | **&&** |
| 10 | მარცხნიდან-მარჯვნივ | **|** |
| 11 | მარცხნიდან-მარჯვნივ | **^** |
| 12 | მარცხნიდან-მარჯვნივ | **&** |
| 13 | არგამოიყენება | **== != === !==** |
| 14 | არგამოიყენება | **< <= > >=** |
| 15 | მარცხნიდან-მარჯვნივ | **<< >>** |
| 16 | მარცხნიდან-მარჯვნივ | **+ -** |
| 17 | მარცხნიდან-მარჯვნივ | **\*/ %** |
| 18 | მარჯვნიდან-მარცხნივ | **! ~ ++ -- (int) (float) (string) (array) (object) @** |
| 19 | მარჯვნიდან-მარცხნივ | **[]** |
| 20 | არგამოიყენება | **new** |
| 21 | არგამოიყენება | **()** |