Hyrje në PHP

PHP: Hypertext Preprocessor (**PHP**) është falas, tepër i populluar, me burime të hapura. Skriptet PHP ekzekutohen në **server**.

Një listë e shkurtër se çfarë është në gjendje të bëjë PHP:

- Gjenerimi i përmbajtjes së faqeve dinamike.
- Krijimi, hapja, leximi, shkrimi, fshirja, dhe mbyllja e filave në server
- Mbledhja e të dhënave nga pyetsorët
-Shtimi, fshirja, dhe modifikimi i informacionit që ndodhet në databazë.
- kontrolli i aksesimit të përdoruesit
- enkriptimi i të dhënave
- dhe shumë më tepër!

Paraprakisht, për të punuar na duhen njohuri bazë të **HTML**.
PHP ka aftësi të punojë në bërthamë të **WordPress-it**, sistemi më i thjeshtë i ndërtimit të një bllogu në web. Për më tepër që PHP ekzekutohet në serverat e **Facebook-ut**, web-i rrjetit më të madh social.

Pse PHP ?

PHP **është** e pavarur nga platforma, që do të thotë funksionon njësoj si në Windows, Linux, Unix, Mac OS X etj.
PHP është i përshtatshëm me çdo server modern, si Apache, IIS etj..
PHP **suporton një grup databazash**.
PHP është falas!
PHP është i thjeshtë për tu mësuar dhe ekzekutohet në mënyrë efiçente në anën e serverit.

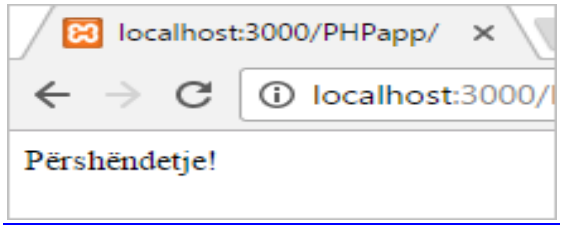
Çfarë duhet për të punuar:

Për të përdorur PHP, hapi i parë është instalimi i një web server në PC, më pas instalimi i PHP.
Mënyra më e thjeshtë është instalimi i një programi që quhet **XAMPP**. Ai përfshin Apache, MySQL, dhe PHP, dhe është falas. XAMPP mund të shkarkohet dhe instalohet nga www.xampp.org.
Hapni XAMPP Control Panel dhe startoni web server-in Apache.

E vetmja gjë që ju duhet për të punuar me PHP është një editor tekst bazë. Krijo një file, shkruaj një shprehje të thjeshtë PHP dhe ruaje me prapashtesën .php:

```
<?php
echo "Përshëndetje!";
?>
```

E ruajmë në folderin rrënjë **index.php** në folderin rrënjë Apache (Në Windows, është zakonisht folder **C:/xampp/htdocs**).
Tani hapni web browser-in dhe shko tek **http://localhost/index.php :**



Sintaksa PHP

Një script PHP mund të vendoset kudo brenda një dokumenti HTML. Fillojnë me tagun **<?php** dhe mbarojnë me **?>**:

```
<?php
// Këtu shkruhet kodi PHP
?>
```

Më poshtë paraqitet një shembull i një file HTML. Skriptet PHP përdorin një funksion të quajtur "**echo**" për të dhënë si output një tekst si "Përshëndetje!" tek një faqe web.

```
<html>
<head>
<title>Faqja ime e parë në PHP</title>
</head>
<body>
<?php
echo "Përshëndetje!";
?>
</body>
</html>
```

Çdo rresht PHP mbaron me pikëpresje **(;)**.

Ne mund të përfshijmë brenda PHP në HTML **<script>** tag.

```
<html>
<head>
<title>Faqja ime e pare në PHP</title>
</head>
<body>
<script language="php">
echo "Përshëndetje!";
</script>
</body>
</html>
```

Por në versionin e fundit rekomandohen vetëm taget <?php ?>

Gjithashtu, mund të përdoren taget e shkurtra, <? ?>, për sa kohë suportohen nga server

Për shembull:

```
<?
echo "Përshëndetje!";
?>
```

Megjithatë, **<?php ?>**, si standart zyrtar, është mënyra që rekomandohen për shkrimin e skripteve PHP .

Echo

PHP ka funksionin e ndërtimit "**echo**" , i cili përdoret për të dhënë output tekst.
Në fakt , nuk është një funksion; është një konstrukt i gjuhës. Pra nuk kërkon thonjëza gjarpërueshe.

Shembull:

```
<?php
echo "Unë po mësoj PHP!";
?>
```

Teksti mund të jetë në thonjëza dyshe ose teke.

Instruksionet PHP

Rreshtat e kodeve në php mbarojnë me pikëpresje.

```
<?php
echo "A";
echo "B";
echo "C";
?>
```

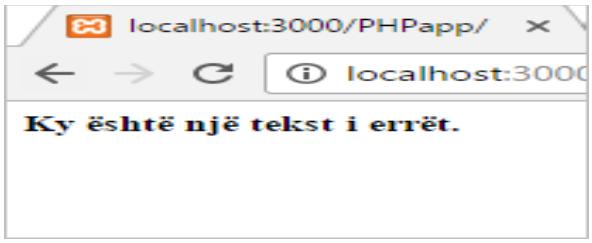
Harrimi i pikëpresjes në fund do të jap error.

Taget HTML mund të shtohen në funksionin php **echo** .

Shembull:

```
<?php
echo "<strong> Ky është një tekst i errët. </strong>";
?>
```

Rezultati:



Komentet

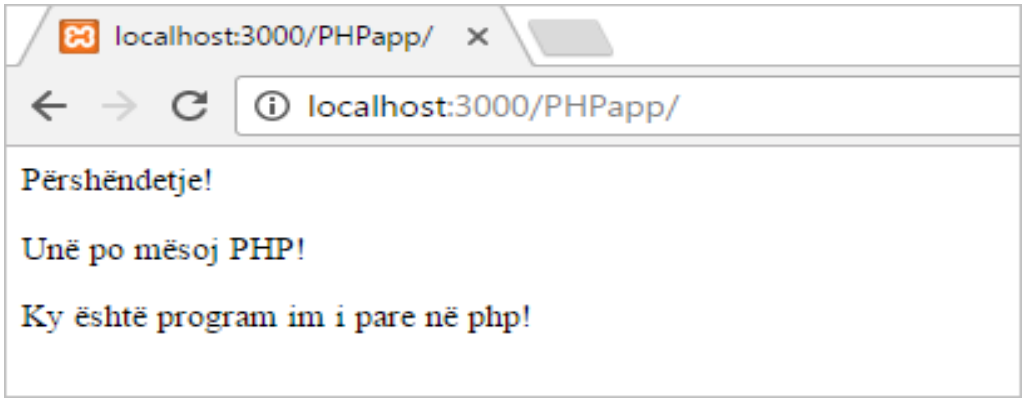
Në kodin PHP, një **koment** është një tekst, i cili nuk ekzekutohet nga programi. Ju mund të përdorni komentet për shpjegimin e kodit që keni shkruar.

Komentet **single-line** fillojnë //:

```
<?php
echo "<p>Përshëndetje!</p>";
// Ky është koment single-line

echo"<p>Unë po mësoj PHP!</p>";
echo "<p>Ky është program  im i pare në php!</p>";
?>
```

Rezultati:



Komentet Multi-line

Komentet **multi-line** përdoret për komentet që janë më shumë se një rresht.

Një koment multi-line fillon me /* dhe mbaron me */.

```
<?php
echo "<p>Përshëndetje!</p>";
/*
Ky është një koment që
ndahet në
tre rreshta
*/
echo"<p>Unë po mësoj PHP!</p>";
echo "<p>Ky është programi im i parë në php!</p>";?>
```

Kjo është një praktikë shumë e mirë, të shtuarit e komenteve në kodin tuaj.
Kjo do të ndihmonte të tjerët për të kuptuar kodin tuaj ose edhe ju kur t'i riktheni një moment më vonë.

Variablat

Variablat përdoren si "mbajtës" në të cilën ne ruajmë informacion.
Një variabël php fillon me një shenjë dollari ($),i cili ndiqet nga emri i variables:

```
$emri_variablës = vlera;
```

Rregullat për variablat PHP:

- Një emër variable duhet të fillojë me një shkronjë ose _
- Një emër variable nuk mund të fillojë me një numër.
- Një emër variable mund të përmbajë shkronja alfanumerike dhe _ (A-z, 0-9, dhe _)
- Emrat e variablave janë case-sensitive ($emri dhe $EMRI janë dy variabla të ndryshme)

Për shembull:

```
<?php
$emri = 'Jon';
$mosha = 25;
echo $emri;
// Output-i 'Jon'
?>
```

Në shembullin e mësipërm, vini re se nuk i themi PHP-së çfarë tipi të dhënash janë variablat. PHP automatikisht konverton variablat në tipin e duhur të të dhënave, në varësi të vlerës së tyre. Ndryshe nga gjuhët e tjera të programimit, PHP nuk ka komanda për deklarimin e variablave. Kjo ndodh në momentin kur jap një vlerë variablave.

Konstantet

Konstantet janë të ngjashme me variablat me përjashtim që ato nuk mund të ripërcaktohen pasi u është dhënë një vlerë e caktuar. Konstantet fillojnë me shkronjë

ose _.
Për të krijuar një konstante, përdoret funksionin **define()** si:

define(emri, vlera, case-insensitive)
Parametrat:
emri: Specifikon emrin e konstantes;
vlera: Specifikon vlerën e konstantes;
case-insensitive: Specifikon nqs do të jetë case-sensitive ose jo. Paraprakisht vlera është is**false;**

Shembulli më poshtë krijon një konstante me emër **case-sensitive** :

```
<?php
define("konstante", "Kjo është vlera konstantes!");
echo konstante;
//Output-i  "Kjo është vlera konstantes!"
?>
```

Shembulli më poshtë krijon një konstante me emër jo **case-sensitive** :

```
<?php
define("konstante", "Kjo është vlera konstantes!", true);
echo KONSTANTE;
//Output-i "Kjo është vlera konstantes!"
?>
```

Përpara emrave të konstanteve nuk është i nevojshëm përdorimi i shenjës së dollarit $

Tipet e të dhënave

Variablat mund të ruajnë lloje të ndryshme të dhënash.
Tipet e të dhënave që suportohen nga
PHP: **String**, **Integer**, **Float**, **Boolean**, **Array**, **Object**, NULL,Resource.
Një **string** është një sekuencë karakteresh ,si "Unë po mësoj PHP!"
Një string mund të jetë brenda thonjëzave teke ose çifte.

```
<?php
$string1 = "Unë po mësoj PHP!"; //Thonjëza çifte
```

```
$string2 = 'Unë po mësoj PHP!'; //Thonjëza teke
?>
```

Për të bashkuar dy stringa përdorim operatorin e bashkimit pikë (.)
Për shembull: **echo $s1 . $s2**

PHP Integer(Numrat e plotë)

Një **integer** është numër i plotë (pa presje) që duhet të përmbush kushtet e mëposhtme:
- Nuk duhet të përmbajë presje ose hapsira
- Nuk duhet të përmbajë pikë dhjetore.
- Mund të marrë vlera positive ose negative:

```
<?php
$int1 = 42; // numër positiv
$int2 = -42; // numër negativ
?>
```

PHP Float

Një **float**, është një numër i cili përfshin një pikë dhjetore.

```
<?php
$x = 42.168;
?>
```

PHP Boolean

Një **Boolean** mund të marrë dy gjendje të ndryshme: TRUE ose FALSE.

```
<?php
$x = true; $y = false;
?>
```

Përgjithësisht përdoren në testimin e kushteve.
Shumë prej kombinimeve të të dhënave përdoren në kombinim me njera-tjetrën. Në këtë shembull, **string** dhe **integer** përdoren së bashku për gjetjen e shumës së dy numrave.

```
<?php
$str = "10";
$int = 20;
$shuma = $str + $int;
echo ($shuma);
// Output-i 30
?>
```

PHP automatikisht do të konvertojë çdo variabël në tipin e duhur të së dhënës, në varësi të vlerës së saj. Kjo është arsyeja pse variabla **$str** trajtohet si numër në mbledhje.

Deklarimi i variablave

Variablat PHP mund të deklarohen kudo në skript . **Scope** i variablave është ajo pjesë e skriptit në të cilën variablat mund të referohen ose përdoren.
Variablat në PHP mund të deklarohen si **local**, **global**, dhe **static**.
Një variabël e deklaruar jashtë një funksioni quhet **global scope**, dhe mund të aksesohen vetëm jashtë funksionit. Një variabël e deklaruar brenda një funksioni ka një **local scope**, dhe mund të aksesohet vetëm brenda këtij funksioni.

Për shembull:

```
<?php
$emri = 'Mira';
function getEmri() {
echo $emri;
}
getEmri();
// Error: Undefined variable: name
?>
```

Ky script do të prodhojë një gabim sepse variabla **$emri** ka një **global** scope, dhe nuk është e aksesueshme brenda funksionit **getEmri()**.

Variabla Global

Fjala global përdoret për të aksesuar një variabël globale brenda një funksioni.

```
<?php
$emri = 'Mira';
function getEmri() {
global $emri;
echo $emri;
}
getEmri();
//Output-i 'Mira'
?>
```

Variabla e variablave

Me PHP, ne mund të përdorim një variabël për t'i dhënë emrin një variable tjetër. Kështu, **vlera e një variable tjetër trajtohet si emri i variablës**.

Për shembull:

```
<?php
$a = 'Pershendetje';
$Pershendetje= "Si jeni!";
echo $$a;
// Output-i 'Si jeni!'
?>
```

$$a është një variabël që përdor vlerën e një variable tjetër, **$a**, është emri i saj. Vlera e **$a** është e njëjtë me " Pershendetje". Rezultati i variablës është **$Pershendetje**, e cila mban vlerën "Si jeni!".

Operatorët

Operatorët mbajnë veprimet dhe vlerat e variablave.

Operandët Aritmetik

Operatorët Aritmetikë punojnë me vlerat numerike për të kryer veprime aritmetike të zakonshme .

Operatori	Emri	Shembull
+	Mbledhja	$x + $y
-	Zbritja	$x - $y
*	Shumëzimi	$x * $y
/	Pjestimi	$x / $y
%	Moduli	$x % $y

Shembull:

```
<?php
$num1 = 8;
$num2 = 6;

//Mbledhja
echo $num1 + $num2; //14

//Zbritja
echo $num1 - $num2; //2

//Shumëzimi
echo $num1 * $num2; //48

//Pjestimi
echo $num1 / $num2; //1.33333333333
?>
```

Moduli

Operatori i modulit, prezantuar me shenjën **%** , kthen mbetjen e numrit të parë me numrit të dytë:

```
<?php
$x = 14;
$y = 3;
echo $x % $y; // 2
?>
```

Në qoftë se do të perdoren numra me presje ato do të konvertohen në numra të plotë para se të kryhen veprime.

Inkrementimi dhe dekrementimi

Operatorët e inkrementimit janë përdorur për të inkrementuar vlerën e variablës.
Operatorët e dekrementimit përdoren për të dekrementuar vlerën e variablës.$

```
x++; // e njëvlefshme me $x = $x+1;
$x--; // e një vlefshme me  $x = $x-1;
$x++; // post-increment
$x--; // post-decrement
++$x; // pre-increment
--$x; // pre-decrement
```

Post-inkrementimi kthen vlerën e variablës para se të inkrementohet,

Ndërkohë pre-inkrementimi ndryshon vlerën e variablës më parë se të kthejë vlerën.

Shembull:

```
$a = 2; $b = $a++; // $a=3, $b=2
$a = 2; $b = ++$a; // $a=3, $b=3
```

Operatorët e veprimeve Aritmetike

Operatorët e veprimeve aritmetike punojnë me vlerat e numrave për të shkruajtur vlera në variabla.

```
$num1 = 5;
$num2 = $num1;
```

$num1 dhe **$num2** tani mbajnë vlerën 5.

Sintaksa	Njëvleshmëria	Përshkrimi
x+=y	x = x + y	Shuma
x-=y	x = x - y	Zbritja
x*=y	x = x * y	Shumëzimi
x/=y	x = x / y	Pjestimi
x%=y	x = x % y	Moduli

Shembull:

```
<?php
$x = 50;
$x += 100;
echo $x;

// Output-i: 150
?>
```

Operatorët e krahasimit

Operatorët e krahasimit krahasojnë dy vlera (numbrat dhe stringjet).
Operatorët e krahasimit përdoren brenda kushteve, dhe marrin vlera **TRUE** ose **FALSE**.

Operatori	Emërtimi	Shembull	Rezultati
==	Barazimi	$x == $y	Kthen true nëse $x dhe $y kanë vlerë të njëjtë.
===	Identik	$x === $y	Kthen true nëse $x dhe $y kanë vlerë të njëjtë dhe janë të të njëjtit tip.
!=	Jo i barabartë	$x != $y	Kthen true nëse $x dhe $y nuk janë të të njëjtit tip.
<>	Jo të njëjtë	$x <> $y	Kthen true nëse $x dhe $y nuk janë të të njëjtit tip.
!==	Jo identik	$x !== $y	Kthen true nëse $x dhe $y nuk kanë vlerë të njëjtë dhe nuk janë të të njëjtit tip.

Operatorët e krahasimit

Operatorët e krahasimit shtesë:

Operatori	Emërtimi	Shembull	Rezultati
>	Më e madhe	$x > $y	Kthen true nëse $x është më e madhe se $y.
<	Më e vogël	$x < $y	Kthen true nëse $x është më e vogël se $y.
>=	Më e madhe ose e barabartë	$x >= $y	Kthen true nëse $x është më e madhe ose e barabartë me $y.
<=	Më e vogël ose e barabartë	$x <= $y	Kthen true nëse $x është më e vogël ose e barabartë me $y

Matricat

Një matricë është variabël speciale, e cila mund të mbajë më shumë se një vlerë në të njëjtën kohë.
Në qoftë se keni një listë njësish (një listë emrash , për shembull), ruajtja e tyre në një variabël të vetme do të ishte:

```
$gjuha = "JavaScript";
$gjuha = "PHP";
$gjuha = "CSS";
```

Por në qoftë se ju keni një liste prej 100 elementësh? Zgjidhja: Krijimi i një **matrice**!

Matricat Numerike

Matricat numerike ose të indeksuar çdo indeksi i shenjohet një vlerë.
Indeksi fillon gjithmonë nga 0), kështu:

```
$meso = array("HTML", "CSS", "PHP");

$meso[0] = "HTML";
$meso[1] = "CSS";
$meso[2] = "PHP";
```

Kemi përcaktuar një matricë me emrin $meso që ka tre vlera. Elementët e matricës mund të thërriten duke thirrur indeksimin e tyre:

```
echo $meso[1]; // Output-i "CSS"
```

Nuk duhet të harrojmë që elementi i parë në matricë është indeksuar me 0.

Matricat Numerike

Ju mund të keni stringje, dhe tipe të tjera të dhënash në një matricë.
Shembull:

```
<?php
$matrica[0] = "Une";
$matrica[1] = "mesoj";
$matrica[2] = "<strong>PHP</strong>";

echo "$matrica[0]  po  $matrica[1] matricat ne $myArray[2]";

// Output-i "Une po mesoj matricat ne PHP"
?>
```

Matricat Shoqëruese

Matricat shoqëruese janë matricat që përdorin çelësat e emërtuara që ju i jepni atyre: Ekzistojnë dy mënyra për krijimin e matricave shoqëruese:

```
$njerez = array("Ben"=>"20", "Mira"=>"21");
// or
$njerez['Ben'] = "20";
$njerez['Mira'] = "21";
```

Shenja **=>** përdoret për dhënien emrave çelësave, të cilët përdoren për aksesimin e anëtarëve të matricës.

```
$njerëz = array("Ben"=>"20", "Mira"=>"21");

echo $njerez['Mira']; // Output-i 21"
```

Matricat Multi-dimensionale

Një matricë multidimensionale përmban një ose më shumë matrica. Përmasat e një matrice tregohen me anë të numrit të treguesve.
- Për një matricë **dy-dimensionale**, ju duhen dy tregues për të selektuar një elementët.
- Për një matricë **tre-dimensionale**, ju duhen tre tregues për të selektuar një element.
Matricat më shumë se tre dimensionale janë të vështira për t'u menaxhuar.

Le të krijojmë një matricë dy-dimensionale që përmban 3 matrica:

```
$njerez = array(
'HTML'=>array('Ben', 'Mira'),
'PHP'=>array('Joni', 'Redi', 'Alma'),
'JavaScript'=>array('Ina', 'Ida')
);
```

Matrica dy-dimensionale **$njerez** përmban 3 matrica, dhe ka **rreshtat dhe kolonat**.
Për të aksesuar elementët e matricës $njerez pointojmë si tek rreshti edhe tek kolona.

```
echo $njerez['HTML'][0]; //Output-i "Ben"

echo $njerez['PHP'][2]; //Output-i "Alma"
```

PHP Case Sensitivity

Në PHP, të gjitha fjalët kyçe (për shembull if, else, while, echo, etj.), klasat, funksionet, si dhe funksionet e përcaktuara nga përdoruesi NUK janë case-sensitive.

Në shembulin më poshtë të treja deklarimet janë të njëjta:

```
<!DOCTYPE html>
<html>
<body>

<?php
ECHO "Pershendetje!<br>";
echo "Pershendetje!<br>";
EcHo "Pershendetje!<br>";
?>
```

```
</body>
</html>
```

Kushtëzimet PHP

Shpesh ju duhet të përdorni kushte të ndryshme për veprime të ndryshme. Në PHP kemi kushtëzimet e mëposhtme:

- **if statement** – ekzekuton disa kode nëse një kusht është i vërtetë.
- **if...else statement** - ekzekuton disa kode nëse një kusht është i vërtetë dhe disa kode të tjera nëse një kusht nuk është i vërtetë.
- **if...elseif....else statement** – ekzekuton kode të ndryshme për më shumë se dy kushte
- **switch statement** – selekton një nga shumë blloqe për t'u ekzekutuar.

PHP – kushtëzimi If

Kushtëzimi if ekzekuton disa kode nëse kushti është i vërtetë.

Sintaksa

if (*kushti*) {
kodi qëdo të ekzekutohet nëse kushti është i vërtetë;
}

Shembulli më poshtë do të printojë "Ditë të mbarë!" nëse koha aktuale është (HOUR) më pak se 20:

```
<?php
$ora = date("H");

if ($ora < "20") {
    echo "Ditë të mbarë!"
}
?>
```

PHP – Kushtëzimi if...else

Kushti if....else ekzekuton disa kode nëse një kusht është i vërtetë dhe disa kode të tjera nëse kushti është I gabuar.

Sintaksa

if (*kusht*) {
    kodi që do të ekzekutohet nëse kushti është i vërtetë;
} else {
    kodi që do të ekzekutohet nëse kushti është i gabuar;
}

Shembulli më poshtë do të japë "Ditë të mbarë!" nëse ora aktuale është më pak se 18, dhe "Naten e mire!" në të kundërt:

```
<?php
$ora = date("H");

if ($ora < "20") {
    echo "Dite te mbare!";
} else {
    echo "Naten e mire!";
}
?>
```

PHP – Kushtëzimi if...elseif....else

Kushti if....elseif...else ekzekuton kode të ndryshme për më shumë se dy kushte.

Sintaksa

if (*kushti*) {
kodi që do të ekzekutohet nëse kushti është i vërtetë;
} elseif (*kushti*) {
   kodi që do të ekzekutohet nëse kushti është i vërtetë;
} else {
kodi që do të ekzekutohet nëse të gjitha kushtet nuk janë të vërteta;
}

Shembulli më poshtë do të japë "Mirëmëngjes!" nëse ora është më e vogël se 12, dhe "Ditë të mbarë" nëse ora është më e vogël se 16. Përndryshe do të japë output "Mirëmbrëma!":

Shembull:

```
<?php
$ora = date("H");

if ($ora < "12") {
    echo "Miremengjes!";
} elseif ($ora < "18") {
    echo "Dite te mbare!";
} else {
    echo "Naten e mire!";
}
?>
```

PHP - switch

Switch përdoret për të selektuar një nga shumë blloqet e kodeve që do të ekzekutohen.

Sintaksa

```
switch (n) {
    case rasti1:
        kodi që do të ekzekutohet nëse  n=rasti1;
        break;
    case rasti2:
        kodi që do të ekzekutohet nëse  n=rasti2;
        break;
    case rasti3:
      kodi që do të ekzekutohet nëse  n=rasti3;
        break;
    ...
    default:
        kodi që do të ekzekutohet nëse n është e ndryshme nga të gjitha rastet;
}
```

Shpjegimi: Fillimisht kemi një shprehje *n* (në shumicën e rasteve një variabël), e cila vlerësohet njëherë. Vlera e shrehjes krahasohet më pas me secilën vlerë për secilin rast në strukturë. Nëse ka përputhje, blloku i kodit shoqërues do të ekzekutohet. Përdoret **break** për të shmangur ekzekutimin automatik të bllokut pasardhës të kodit . **Default** përdoret nëse nuk ka asnjë përputhje.

Shembull:

```
<?php
$ngjyra = "kuqe";

switch ($ngjyra) {
    case "kuqe":
        echo "Ngjyra juaj e preferuar eshte e kuqja!";
        break;
    case "blu":
        echo "Ngjyra juaj e preferuar eshte blu”
        break;
    case "gjelbert":
        echo "Ngjyra juaj e preferuar eshte e gjelberta”
        break;
    default:
        echo "Ngjyra juaj e preferuar nuk eshte as kuqe, as blu, as gjelbert!";
}
?>
```

PHP Loops

Shpesh kur shkruani kod, dëshironi që të ekzekutoni një bllok kodi disa herë. Për këtë përdoren ciklet(loops).

Në PHP, ekzistojnë ciklet e mëposhtme:

- **while** – ekzekuton një bllok kodi për sa kohë një bllok kodi është i vërtetë.
- **do...while** – ekzekuton njëherë bllokun e kodit, dhe më pas përsërit ekzekutimin e ciklit për sa kohë kushti është i vërtetë.
- **for** – ekzekuton një bllok kodi për një numër të caktuar herësh.
- **foreach** - ciklon një bllok kodi për çdo element në matricë.

Cikli While PHP

Cikli **while** ekzekuton një bllok kodi për sa kohë një kusht është i vërtetë .

Sintaksa

while (*kushti eshte i vertete*) {
    kodi qe do te ekzekutohet;
}

Shembulli më poshtë fillimishti jep vlerën $x në1 ($x = 1). Më pas, do të vazhdojë ekzekutimin për sa kohë $x është më i vogël ose i barabartë 10 ($x <= 10). $x do të rritet me 1 sa herë do të ekzekutohet cikli ($x++):

Shembull

```
<?php
$x = 1;
while($x <= 10) {
    echo "Numri është: $x <br>";
    $x++;
}
?>
```

PHP While Loop

Cikli do...while gjithmonë do të ekzekutojë kodin njëherë, më pas kontrollon kushtin, dhe më pas përsërit ciklin për sa kohë kushti i specifikuar është i vërtetë.

Sintaksa

do {
    kodi që do të ekzekutohet;
} while (*kushti është i vërtetë*);

Shembulli më poshtë fillimisht i jep variablës $x vlerën 1 ($x = 1). Më pas, cikli **do while** do të shkruajë outputin, dhe më pas do të inkrementojë variablën $x me 1. Pasi kontrollohet kushti (nëse $x është më i vogël, ose i barabartë me 10), dhe më pas cikli do të vazhdojë ekzekutimin për sa kohë $x është më i vogël ose i barabartë me 10:

Shembull

```
<?php
$x = 1;
do {
    echo "Numri është : $x <br>";
    $x++;
```

```
} while ($x <= 10);
?>
```

Kushti në ciklin **do while** do të ekzekutohet **pasi** blloku i kodit të jetë ekzekutuar njëherë. Kjo do të thotë që cikli **do while** do të ekzekutohet të paktën njëherë edhe pse kushti mund të mos jetë i vërtetë. Shembulli më poshtë i jepvariablës $x vlerën 10, ekzekuton ciklin më pas kontrollohet kushti:

Shembulli:

```
<?php
$x = 10;

do {
   echo "Numri është: $x <br>";
   $x++;
} while ($x<=7);
?>
```

Cikli for

Cikli **for** përdoret kur dihet paraprakisht se sa herë do të ekzekutohet cikli.

Sintaksa

for (*vlera fillestare; kushti; rrit vlerën fillestare*) {
   kodi që do të ekzekutohet;
}

Parametrat:

- *vlera fillestare*: Inicializimi i ciklit
- *kushti* : Kontrollohet për çdo cikël. Nëse është i vërtetë, cikli vazhdon. Nëse nuk është i vërtetë cikli mbaron.
- *Vlera rritjes*: Rrit vlerën e numëruesit të ciklit.

Shembulli më poshtë paraqet numrat nga 0 në 10:

Shembull:

```
<?php
for ($x = 0; $x <= 10; $x++) {
```

```
    echo "Numri është: $x <br>";
}
?>
```

Cikli PHP foreach

Cikli **foreach** punon vetëm në matrica, dhe përdoret për të kaluar në secilin element të matricës.

Sintaksa

foreach ($*matrica* $*vlera*) {
kodi që do të ekzekutohet;
}

Shembull:

```
<?php
$ngjyrat = array("kuqe", "gjelbërt", "blu", "verdhë");

foreach ($ngjyrat as $values) {
    echo "$value <br>";
}
?>
```

Funksonet PHP

Fuqia reale e PHP vjen nga funksionet e tij; ka më shumë se 1000 funksione të para-ndërtuara.

Funksionet e paracaktuara PHP

Pavarsisht funksioneve të paracktuara PHP, ne mund të krijojmë funksione sipas nevojës.

Një funksion është një bllok i cili përdoret në mënyrë të përsëritur në program.

Kur ngarkohet një faqe, funksionet nuk ekzekutohen menjëherë.

Një funksion do të ekzekutohet kur funksioni thërritet.

Krjimi i një funksioni

Një funksion i përcaktuar nga përdoruesi fillojnë deklarimin me fjalën kyçe "function"

Sintaksa

function *EmriFunksioni*() {
    kodi që do të ekzekutohet;
}

Shënim: Një emër funksioni mund të fillojë me shkonjë ose një underscore (-) jo numbër.

Këshillë: Emërtoji funksionet me emra që identifikojnë rolin e tyre.

 Funksionet e emrave **NUK jane case-sensitive.**

Në shembullin më poshtë, krijojmë funksionin "mesazh()". Kllapat gjarpërushe ({) tregojnë fillimin e bllokut të një funksioni, dhe (}) tregojnë fundin e një funksioni. Output-i "Pershendetje". Për të thërritur funksionin, shkruajmë thjesht emrin e funksionit:

Shembull

```
<?php
function mesazh() {
    echo "Pershendetje!";
}
mesazh(); // therritja e funksionit
?>
```

Argumentat e funksioneve PHP

Informacioni mund të kalohet funksioneve përmes argumentave. Një argument është si një variabël.

Argumentat specifikohen pas emrit të funksionit brenda kllapave. Mund të shtohen sa argument të jetë e nevojshme të ndara me presje.

Shembulli i mëposhtëm ka një funksion me një argument ($mbiemer). Ku mbiemri() është funksioni thërritur:

Shembull:

```
<?php
function mbiemri($mbiemer) {
    echo "$mbiemer <br>";
```

```
}
mbiemri("Jani");
mbiemri("Hoxha");

?>
```

Funksionet PHP – Kthimi i vlerave

Për kthimin e vlerës së një funksioni përdoret fjala kyçe return:

Shembull:

```
<?php
function shuma($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "1 + 2 = " . shuma(1, 2) . "<br>";
echo "5 + 10 = " . shuma(5, 10) . "<br>";
echo "20 + 40 = " . shuma(20, 40);
?>
```

Një matricë ruan vlera të shumta në një vriabël të vetme.:

Shembull:

```
<?php
$meso = array("HTML", "CSS", "JavaScript",”PHP”);
echo "Une po mesoj " . $HTML[0] . ", " . $CSS[1] . "," . $JavaScript[2] . "  dhe " . $PHP
[3] . ".";
?>
```

Çfarë është një matricë?

Një matricë është një variabël speciale, e cila mund të mbajë më shumë se një vlerë në të njëjtën kohë.

Nese kni një list të njësive (një listë të gjuhëve që duhen mësuar), ruajtja e tyre në një variabël të vetme do të ishte kështu:

```
$meso1= "HTML";
$meso2 = "CSS";
$meso3 = "JavaScript";
$meso4= "PHP"
```

Por në rastin kur nuk keni katër por më shumë?
Zgjidhja është krijimi i matricave!

Një matricë mund të mbajë shumë vlera në një emër të vetëm dhe ju mund t'i aksesoni duke ju referuar një numër indeksi.

Krijimi i matricave PHP

Në PHP,funksioni array() përdoret për krijimin e një matrice:

array();

Në PHP ekzistojnë tre tipe matricash:

- **Matricat e indeksuara** – Matricat me indeks numerik
- **Matricat shoqëruse** – Matricat me çelësa emrash
- **Matricat multidimensionale** – Matricat që përmbajnë një ose më shumë matrica

Matricat e indeksuara

Janë dy mënyra për krijimin e matricave të indeksuara:

Indeksi shenjohet automatikisht (index fillon gjithmonë me 0), si:

$meso = array("HTML", "CSS", "JavaScript", "PHP"););

Ose indeksi mund të shenjohet manualisht:

```
$meso[0] = "HTML";
$meso[1] = "CSS";
$meso[3] = "JavaScript";
$meso[4] = "PHP";
```

Shembulli i mëposhtëm krijon një matricë të indeksuar të quajtur mëso :

```
<?php
$meso = array("HTML", "CSS", "JavaScript",”PHP”);
echo "Une po mesoj " . $HTML[0] . ", " . $CSS[1] . "," . $JavaScript[2] . "  dhe " . $PHP
[3] . ".";
?>
```

Gjetja e gjatësisë së një matrice – Funksioni count()

Funksioni count() përdoret për kthimin e gjatësisë (numrit të elementëve të një matrice) në një matrice:

Shembull

```
<?php
$meso = array("HTML", "CSS", "JavaScript",”PHP”);
echo count($meso);
?>
```

Cikli përmes një matrice të indeksuar

Për të kaluar dhe printuar të gjitha vlerat e një matrice përdoret for loop, si:

Shembull:

```
<?php
$meso = array("HTML", "CSS", "JavaScript",”PHP”);
$permasa = count($meso);

for($x = 0; $x < $permasa; $x++) {
    echo $meso[$x];
    echo "<br>";
}
?>
```

Matricat PHP Shoqëruese

Dy mënyra për krijimin e matricave shoqëruese:

```
$nota = array("Matematike"=>"9", "Programim"=>"10", "Letersi"=>"8");
```

ose:

```
$nota['Matematike'] = "9";
$nota['Programim'] = "10";
$nota['Letersi'] = "8";
```

Çelësat e emëruar mund të përdoren më pas në skript:

Shembull

```
<?php
$nota = array("Matematike"=>"9", "Programim"=>"10", "Letersi"=>"8");

echo "Në matematike ka marre noten " . $nota['9'] . " kete semester.";
?>
```

Cikli në një matricë shoqëruese

```
<?php
$nota = array("Matematike"=>"9", "Programim"=>"10", "Letersi"=>"8");

foreach($nota as $x => $x_vlera) {
    echo "Celesi=" . $x . ", Value=" . $x_vlera;
    echo "<br>";
}
?>
```

Variablat Globale PHP - Superglobalet

Disa nga variablat e paracaktuaranë PHP janë "superglobalet", që do të thotë që janë gjithmonë të aksesueshme nga çdo funksion, klasë ose file(skedar).

Variablat superglobale PHP janë:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

- Variabla PHP $GLOBALS

$GLOBALS është një variabël globale e cila përdoret për aksesimin e variablave globale nga kudo në program (edhe nga brenda funskioneve apo metodave).

PHP ruan të gjitha variablat globale në një matricë të quajtur $GLOBALS[*index*]. Fjala *index* ruan të gjithë emrat e variablave:

Shembull përdorimi të $GLOBALS:

Shembull:

```
<?php
$x = 100;
$y =200;
 function mbledhja() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}
mbledhja();
echo $z;
?>
```

Në shembullin e mësipërm, mëqënëse z është variabël brenda matricës $GLOBALS, është gjithmonë e aksesueshme nga jashtë funksionit!

PHP $_SERVER

$_SERVER është një variabël super globale e cila mban informacion rreth *headers*, patheve, dhe vendodhjeve të skriptit.

Shembulli më poshtë tregon se si të përdorim disa elemente në $_SERVER:

Shembull:

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
```

```
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

PHP $_REQUEST

PHP $_REQUEST përdoret për mbledhjen e të dhënave pas dërgimit të dhënave të një kontakt forme.

Shembulli më poshtë tregon një kontakt formë me një fushë input dhe një buton submit (të dërgimit të të dhënave). Kur përdoruesi klikon butonin "Dergo", të dhënat dërgohen tek skedari i specifikuar në atributin action të tagut <form> . Në këtë shembull, ne pointojmë sërish tek ky skeadar. Nëse përdoret një skedar tjetër PHP atëherë duhet të vendoset emri apo path-i I duhur tek atributi. Më pas, ne mund të përdorim variablat super globale si $_REQUEST për mbledhjen e të dhënave nga fushat input:

Shembull

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
Emri: <input type="text" name="emri">
  <input type="Dergo">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // mbledhja e të dhënave nga fushat input
    $emri = $_REQUEST['emri'];
    if (empty($emri)) {
        echo "Emri  është bosh";
    } else {
        echo $emri;
    }
}
?>
</body>
</html>
```

PHP $_POST

PHP $_POST përdoret gjerësisht për mbledhjen e të dhënave pas dërgimit të një forme HTML, gjithashtu method="post". $_POST përdoret gjerësisht për kalimin e variablave.

Shembulli më poshtë tregon një formë me një fushë input dhe dhe një buton të dërgimit të të dhënave. Kur një përdorues dërgon të dhënat duke klikuar "Dërgo", të dhënat e formës do të dërgohen tek skedari i specifikuar në atributin action të tagut <form>. Në këtë shembull, ne ointojmë tek vetë skedari për proçesimin e të dhënave. Nëse do të përdorim një skedar tjetër PHP për proçesimin e të dhënave, atëherë emri i skedarit do të përdoret tek atributi action. Më pas, ne mund të përdorim variablën super globale $_POST për mbledhjen e të dhënave nga fusha input:

Shembull:

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
Emri: <input type="text" name="emri">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // mbledhja e të dhënave nga fusha input
    $emri = $_POST['emri'];
    if (empty($emri)) {
        echo "Emri është bosh";
    } else {
        echo $emri;
    }
}
?>

</body>
</html>
```

PHP $_GET

PHP $_GET mund të përdoret gjithashtu për mbledhjen e të dhënave pas dërgimit të formës me metodën GET : method="get".

$_GET mund të mbledhë edhe të dhëna nga një URL.

Supozojmë se kemi një faqe HTML që përmban hyperlinqet me parametrat:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=mesoprogramim.com">Test $GET</a>

</body>
</html>
```

Kur përdoruesi klikon tek linku "Test $GET", parametrat "subject" dhe "web" dërgohen tek "test_get.php", më pas ju mund të aksesoni vlerat e tyre tek "test_get.php" me $_GET.

Shembulli më poshtë tregon kodin në "test_get.php":

Shembull:

```
<html>
<body>

<?php
echo "Meso " . $_GET['tema'] . " at " . $_GET['web'];
?>

</body>
</html>
```

Variablat super globale PHP $_GET dhe $_POST përdoren për mbledhjen e të dhënave nga format.

Format

Shembulli më poshtë tregon një formë të thjeshtë HTML me dy fusha input HTML dhe buton të dërgimit të të dhënave (submit):

Shembull

```
<html>
<body>
<form action="regjistrimi.php" method="post">
Emri <input type="text" name="emri"><br>
E-mail: <input type="text" name="email"><br>

Regjistrohu<input type="submit">
</form>
</body>
</html>
```

Kur përdoruesi plotëson të dhënat dhe klikon butonin dërgo, të dhënat dërgohen për proçesim tek një skedar PHP të quajtur regjistrimi.php. Të dhënat dërgohen me meodën HTTP POST.

Për shfaqjen e të dhënave ju thjesht bëni echo të gjithë variablave. Skedari "regjistrimi.php" do të jetë kështu:

```
<html>
<body>

Përshëndetje <?php echo $_POST["emri"]; ?><br>
ju u regjistruat në adresën : <?php echo $_POST["email"]; ?>

</body>
</html>
```

I njëjti rezultat mund të merret edhe duke përdorur metodën HTTP GET:

Shembull:

```
<html>
<body>

<form action="regjistrimi.php" method="get">
Emri: <input type="text" name="emri"><br>
E-mail: <input type="text" name="email"><br>
Dërgo <input type="submit">
</form>

</body>
</html>
```

Dhe "regjistrimi.php" duket kështu:

```
<html>
<body>

Përshëndetje <?php echo $_POST["emri"]; ?><br>
ju u regjistruat në adresën : <?php echo $_POST["email"]; ?>

</body>
</html>
```

Kodi më sipër është shumë i thjeshtë . Megjithatë, mungon gjëja më e rëndësishme. Ju duhet të validoni të dhënat për të mbrojtur skriptin nga kodet e rrezikshme.

Mendo SIGURINE kur proçeson të dhënat me format PHP!

Këtu tregohet ndonjë validim forme, thjesht tregon se si mund të dërgoni dhe merrni të dhëna.

GET vs. POST

Të dyja GET dhe POST krijojnë një matricë (shembull: array(çelësi1=> vlera1, çelësi2=> vlera2, çelësi3=> vlera3, ...)). Kjo matricë mban çiftet çelës/vlerë, ku çelësat janë emrat e formave dhe vlerat janë të dhënat input nga përdoruesi.

Të dyja GET dhe POST trajtohen si $_GET dhe $_POST. Këto janë superglobalet, që do të thotë janë gjithmonë të aksesueshëm, dhe mund të thërriten nga çdo funksion, klasë apo skedar.

$_GET është një matricë variablash që i kalon skriptit aktual parametrat URL.

$_POST është një matricë variablash që i kalon skriptit aktual nëpërmjet metodës HTTP POST.

Kur mund të përdoret GET?

Informacioni I dërguar nëpërmjet metodës GET është I dukshëm tek çdokush (të gjithë emrat dhe vlerat e variablave shfaqen në URL). GET ka gjithashtu limite në sasinë e informacionit që do të dërgohet.Limitimi është rreth 2000 karaktere. Megjithatë, mëqënëse variablat shfaqen në URL, është e mundur bërja bookmark e faqes. Kjo mund të jetë e dobishme në disa raste.

Shënim: GET NUK duhet të përdoret asnjëherë për dërgimin e password-eve ose të dhënave të tjera delikate!

Kur mund të përdoret POST?

Informacioni i dërguar nga metoda POST nuk është i dukshëm tek të tjerët (të gjitha emrat/vlerat janë të ndërfutura në trupin e kërkesës HTTP) dhe **nuk kanë limite** në sasinë e informacionit që dërgohen.

Megjithatë, mëqënëse variablat nuk janë të dukshme në URL, nuk është e mundur bërja bookmark e faqes.

Tagu formë :

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

Të dhënat e formës dërgohen me metodën post method="post".

Çfarë është variabla $_SERVER["PHP_SELF"]?

$_SERVER["PHP_SELF"] është një variabël super globale e cila kthen emrin e skedarit aktual që po ekzekutohet.

Pra variabla super globale $_SERVER["PHP_SELF"] ia dërgon të dhënat vetë faqes, në vend të kapërcimit në një faqe tjetër.

Çfarë është funksioni htmlspecialchars() ?

Funksioni htmlspecialchars() konverton karakteret speciale në entitete HTML. Kjo do të thotë që karakteret si

< dhe > do të zëvendësohen me < dhe >. Kjo do të parandalojë sulmuesit nga ndërhyrjet në kodin HTML

apo Javascript.

Kujdes!!

Variabla $_SERVER["PHP_SELF"] mund të aksesohet nga hackers-at!