PHP gyorstalpaló, avagy a Hello World-től az űrlapellenőrzésig

```
<!DOCTYPE html>
<html>
<head>
	<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>

<?php
echo "Hello World";
```

SZINTAXIS

A PHP kód többnyire HTML tag-ek között jelenik meg a nyitó **<php** tag-el és a záró **?>** záródik.
Bármilyen karakter használható a forráskódban.
A PHP kód csak akkor lesz értelmezve, ha a fájl kiterjesztése: .php
Lehetségesek olyan fájlok is amikben nincs HTML tag csak php kód.
Minden utasítást **;** zár. Elhagyása szintaktikai hibához vezet.

```
//egysoros megjegyzés
/*
	Több soros
	megjegyzés
*/
```

VÁLTOZÓK

PHP-ben nincs változó deklaráció, akkor jön létre a változó, amikor először elkezded használni.
Hasonlóan a javascript-hez.
Változók előtt mindig egy $ jel áll, ennek lehagyása szintaktikai hibához vezet.
A PHP gyengén típusos nyelv. Automatikus típuskonverzió.
Ha a változó nem kap kezdőértéket, akkor üres.

```php
$txt="Hello World";
$x=16;
echo $txt;
echo $x;
echo "<br />";
```

HATÁSKÖR

A PHP változók hatáskörei a következők lehetnek

- lokális: csak adott függvényen belül érhető el
- globális: bárhonnan elérhető
- statikus: a függvény lokálisan létrehozott változója a függvény végén nem kerül törlésre
- paraméter: a paraméterként átadott változók a függvény lokális változói

Ha egy változó egy függvényben van deklarálva, akkor az lokális lesz és csak azon a függvényen belül érhető el.

```php
$a = 5;
echo $a;
function test() {
    echo $a; // ez egy lokális változó ami üres, mert most van először használva
    $b = 10; // lokális
    echo $b;
    //globális $a változó elérése:
    echo "<br />a változó globálissá minősítése után:<br />";
    global $a; //így $a már a globális $a változót hivatkozza
    echo $a; //globális $a
}
echo $b; //üres globális változó
test();
```

STRING

A string változó, karakter vagy karakter láncok tárolására szolgál, csak úgy mint a többi nyelvnél.

A legfontosabb string kezelő függvények:

- Az összefűzés operátora: .
- A string hossza: strlen()
- Karakter vagy szöveg pozíciójának keresése egy másik karakterláncban: strpos()

Teljes körű String referencia:

http://www.w3schools.com/php/php_ref_string.asp

```php
//Stringek összefűzése:
echo "<br />";
$txt1 = "Mi a mai";
$txt2 = "mocsári gólyahír?";
$txt3 = $txt1.$txt2; //stringek összefűzése egy harmadik változóba
echo $txt1.$txt2." Van e kaja?"; //két változó + egy literál összefűzése
echo "<br />A változók kííratásának másik módja: $txt3";

//strlen(string): a paraméterül megadott string hosszával tér vissza
echo "<br />txt1 hossza: ".strlen($txt1);
//strpos(string1,string2): string1 -ben keresi string2 első előfordulását
echo "<br />A 'hír' szó kezdőpozíciója txt2-ben: ".strpos($txt2,"hír");

```

OPERÁTOROK

A c és javascript nyelvnél megismertekkel megegyezik.

Logikai összekötő operátoroknál használható a pascal-os and or vagy a c szerű: && ||

If … else szerkezet

A c és javascript nyelvnél megismertekkel megegyezik.

Szintaktika:

rövid alak:

```
if (feltétel) tevékenység;
```

hosszú alak:

```
if (feltétel) tevékenység1; else tevékenység2;
```

elseif szerkezet (a PHP nyelv egyik baromsága, hogy összevont két alapszót ezzel létrehozva ezt a valamit)

megj.: ugyanazt csinálja, mintha külön írnánk: else if

```
if (feltétel)
        tevékenység1;
elseif (feltétel)
        tevékenység2;
else
        tevékenység3;
```

Az előzővel ekvivalens verzió, ami minden nyelvben megtalálható:

```
if (feltétel)
        tevékenység1;
else if (feltétel)
        tevékenység2;
else
        tevékenység3;
```

Szemantika:

Ha a feltétel értéke igaz, akkor végrehajtódik az if utáni tevékenység, ellenkező esetben ha van else ág, akkor az abban megadott tevékenység hajtódik végre.

Példák:

```
1. if (5<10) echo "kisebb<br />"; else echo "nem kisebb<br />";
2. if (10!=10 && 3==5 and $a==$b) echo "igaz<br />"; else echo "hamis<br />";
   //összetett feltétel
3. if (10>10) echo "nagyobb<br />"; //elseif szerkezet
4.     elseif (10<10) echo "kisebb<br />";
5.         else echo "egyenlo<br />";
6. if (10>10) echo "nagyobb<br />"; //előzővel ekvivalens
7.     else if (10<10) echo "kisebb<br />";
8.         else echo "egyenlo<br />";
```

Switch szerkezet

A c és javascript nyelvnél megismertekkel megegyezik.

Szintaktika:

```
swicth (kifejezés) {
	case címke:
		tevékenység;
		break;
	case címke:
		tevékenység;
		break;
	default:
		tevékenység;
}
```

Szemantika:

A kifejezés kiértékelődik és összehasonlításra kerül a címkékkel, ha van egyezés végrehajtódik az adott case ág tevékenysége. A break ugyanúgy működik, mint c-ben (ha lehagyjuk végrehajtódik a többi tevékenység is) a default ág akkor hajtódik végre, ha egyik case ággal sincs egyezés.

Példa:

```php
$x=2;
switch ($x+1){
    case 2:
      echo "x+1= 2<br />";
      break;
    case 3:
      echo "x+1= 3<br />";
      break;
    case 4:
      echo "x+1= 4<br />";
      break;
    default:
      echo "Nem 2,3 vagy 4.<br />";
}
```

Tömbök

A tömbök arra valók, hogy több értéket/adatot tároljunk egyetlen változóban és azt az indexükkel hivatkozzuk.

A PHP nyelvben a tömbök indexei nem csak szám típusúak lehetnek, hanem szövegesek is, az ilyen tömböket asszociatív tömböknek nevezzük.

```php
// Tömbök létrehozása szám típusú tömbindexekkel. Két lehetőség van:
// 1. eset: array kulcsszó használata, ilyenkor az indexek automatikusan hozzárendelődnek. Az első elem indexe 0.
    $cars=array("Saab","Volvo","BMW","Toyota");
    echo $cars[2]."<br />";
// 2. eset: mi mondjuk meg hogy melyik indexre milyen érték kerüljön:
    $cars2[0] = "Saab";
    $cars2[1] = "Volvo";
    $cars2[2] = "BMW";


// Asszociatív tömbök létrehozása (az index szöveges típusú)
// 1. eset:
    $ages = array("Péter"=>32, "Beatrix"=>30, "Tamás"=>34); //személyek nevéhez rendeljük az életkort
    echo 'Beatrix '.$ages['Beatrix']." éves.<br />";
// 2. eset:
    $ages['Péter'] = "32";
    $ages['Beatrix'] = "30";
    $ages['Tamás'] = "34";


// Több dimenziós tömbök: olyan 1 dimenziós tömbök amelyek elemei is tömbö
    $_2x2[0][0] = 0; // 0 1
    $_2x2[1][0] = 1; // 2 3
    $_2x2[0][1] = 2;
    $_2x2[1][1] = 3;
    echo "egy 2*2-es tömb: "."<br />";
    print_r($_2x2); //a print_r metódussal kiírathatjuk egy tömb tartalmát
    echo "<br />[0][1] indexű eleme: ".$_2x2[0][1]."<br />";

    //vegyes tömb (asszociatív és numerikus)
    $families = array( //ennek a tömbnek az első indexe szöveges (Péter, Beatrix, Tamás), második indexe pedig numerikus.
      "Péter"=>array(
        "Péter", //Péter 0-adik indexű gyereke
        "Zoli",  //Péter 1-edik indexű gyereke
        "Aladár" //Péter 2-edik indexű gyereke
      ),
      "Beatrix"=>array(
        "Ágota"  //Beatrix 0-adik indexű gyereke
      ),
      "Tamás"=>array(
        "Helga", //Tamás 0-adik indexű gyereke
        "Ottó"   //Tamás 1-edik indexű gyereke
      )
     );
    print_r($families);
    echo "<br />".$families['Péter'][1];

```

While ciklus:

Szintaktikája és szemantikája is megeggyezik a C ben tanultakkal.

Igazra ismétel.

```
while (feltétel) {
	ciklusmag
}
```

Példa:

```php
echo "while ciklus:<br />";
$i=1;
while($i<=5) {
  echo $i."<br />";
  $i++;
}
```

Do ... while ciklus:

Szintaktikája és szemantikája is megeggyezik a C ben tanultakkal.

Igazra ismétel.

```
do {
	ciklusmag
} while(feltétel)
```

Példa:

```php
echo "<br />do while ciklus:<br />";
$i=1;
do
  {
  echo $i."<br />";
  $i++;
  }
while ($i<=5);
```

For ciklus:

Szintaktikája és szemantikája is megeggyezik a C ben tanultakkal.

```
for (inicializáló kifejezés; feltétel; növekmény) {
	ciklusmag
}
```

Példa:

```php
echo "for ciklus:<br />";
for ($i=1; $i<=5; $i++) {
    echo $i."<br />";
}
```

Foreach ciklus:

Arra való, hogy tömböket járjunk be vele. Működése hasonló a mootools each -hez.

```
foreach ($tomb as $ertek) {
	ciklusmag
}
```

$ertek változó minden egyes ciklislépésben felveszi a tömb következő adatelemét (tehát itt a ciklusváltozó nem az indexeken, hanem magán az adatokon megy végig)

Példa:

```php
echo "<br />foreach ciklus:<br />";
$tomb=array("egy","kettő","három");
foreach ($tomb as $ertek) {
    echo $ertek."<br />";
}

```

Függvények:

Több mint 700 beépített függvény található a PHP-ben, amik segítik a munkánkat, de természetesen saját függvényeket is készíthetünk.

php függvények referenciája: http://www.w3schools.com/php/default.asp

Hasonlóan a javascripthez itt is, a függvényben lévő utasítások nem oldalbetöltődéskor, hanem a függvény meghívásakor hajtódnak végre.

Függvényt bárhol meg lehet hívni, ahol utasítás állhat.

Szintaktika:

```
function függvényNév(paraméterLista) {
}
```

A paraméterlistában átadhatók a függvénynek változók, tömbök stb. Ha üres a paraméterlista a zárójeleket akkor is kitesszük.

Valamint, ahogy a többi nyelvnél is megszokhattuk a függvény visszatérhet értékkel is a hívó felé, erre szolgál a return utasítás.

Példa:

```php
function osszeadas($x, $y) {
    $total = $x + $y;
    return $total;
}
echo "Függvényhívás eredménye: ".osszeadas(3,2)."<br />";

```

$_GET és $_POST előre definiált változók

Mindkettő egy előre definiált változó, tulajdonképpen egy-egy asszociatív tömb.
Ezekbe a tömbökbe a kliens oldalon adatokat lehet tenni, majd szerver oldalon a PHP ki tudja belőle nyerni.
$_GET változó arra való, hogy adatokat gyűjtsön be egy űrlapból, vagy URL-ből amely get-el lett elküldve.
$_POST változó arra való, hogy adatokat gyűjtsön be egy űrlapból, amely post-al lett elküldve.
A post vagy get beállítása az űrlap tag-nél lehetséges: <form method="post"> vagy <form method="get">
-A kettő között a különbség az hogy a get-el küldött adat látható a felhasználó számára, megjelenik az URL-ben, és a vele küldhető adatmennyiség korlátozott.
-A post-al küldött adat ezzel szemben nem jelenik meg az URL-ben, a felhasználó nem láthatja és a vele küldhető adatmennyiség is jóval nagyobb alap értéken 8Mb, de ez állítható a php.ini-ben (post_max_size)
Mikor érdemes get-et használni:

- ha a küldendő adatokat a felhasználó megtekintheti az url-ben, ez azért jöhet jól, mert így a link könyvjelzőzhető
- például navigációkor, hogy az url-t valakinek átmásolva ugyanarra az aloldalra jusson
- amikor védett adatot akarunk küldeni a szervernek mint például jelszó akkor nem érdemes használni helyette ott a post

Mikor érdemes post-ot használni:

- ha a küldendő adatokat el akarjuk rejteni
- nagyobb mennyiségű adatoknál

A $_REQUEST változó tartalmazza mind a post és mind a get adatait.
Példa get-re:

```
1. ?>
2. <a href="php_basic.php?tulajdonsag=ertek">URL-
   be rakunk információt, amit aztán ki is olvasunk</a>
3. <?php
4. if (isset($_GET["tulajdonsag"])) { //ez az ág csak akkor fog lefutni, ha a
   $_POST["name"] változónak van értéke.
5.     echo "<br />Az url-
   ben a tulajdonsag erteke: ".$_GET["tulajdonsag"]."<br />";
6. }
```

Űrlapok feldolgozása:

Az egyik legfontosabb és talán legsűrűbben használt funkcionalitása a PHP-nek hogy űrlapokat dolgozzunk fel vele, leellenőrizzük és az adatokat adatbázisba felvigyük.

Példa:

```
?>
<form action="php_basic.php" method="post">
Név: <input type="text" name="name">
Életkor: <input type="text" name="age">
<input type="submit">
</form>
<?php
if (isset($_POST["name"])) { //ez az ág csak akkor fog lefutni, ha a $_POST["name"] változónak van értéke.
    echo "Szia ".$_POST["name"]."!";
    $age=$_POST["age"];
    if ($age>18)
        echo "<br />Jó öreg vagy hehe :D";
    else
        echo "<br />életkorod: ".$age;
}

```