Backend 12

PHP Alapok

Rostagni Csaba

2024. szeptember 25.

Tartalom

Tartalom

Mi az a PHP?

- Rekurzív betűszó (**P**HP: **H**ypertext **P**reprocessor)
  „Personal Home Page Tools” névből származik
- Szkript nyelv
  - Értelmező szükséges hozzá
  - Hordozhatóbb, ahol van értelmező futnia kell
- Gyengén típusos
  - A 7-es verzióban már lehet típusokat megadni osztályokban
  - A 8-as verzióban már széles körben használhatóak a típusok

Hello World

hello-01.php

```
1 Hello World
```

Első PHP szkriptünk:

hello-02.php

```
1 <?php
2     echo "Hello World";
3 ?>
```

A kód tovább rövidítve:

hello-03.php

```
1 <?= "Hello World"; ?>
```

echo és var_dump()

PHP
```php
$nev = "Bence";
```

PHP
```php
echo $nev;
```

Eredmény
```
Bence
```

- Az `echo` megjeleníti a mögé írt kifejezéseket
- Mivel nem függvény, hanem nyelvi konstrukció, így a zárójelezés elhagyható

PHP
```php
var_dump($nev);
```

Eredmény
```
string(5) "Bence"
```

- A megadott kifejezéseket jeleníti meg, a típusával együtt

Linkek:

- echo - PHP Dokumentáció
- var_dump - PHP Dokumentáció

Tartalom

Megjegyzések

PHP

```php
<?php
// sor végéig érvényes megjegyzés

# sor végéig érvényes megjegyzés

/*
    akár több
    soros megjegyzés
*/
?>
```

DocBlock

- Speciális megjegyzés
- A dokumentációs kommentet /** és */ jelek közé tesszük.

PHP

```
/**
* Az első sor a rövid összefoglaló
*
*  * Használhatunk _markdown_ -t a leírásban.
*  * Lehet többsoros a leírásunk
*  * Végül a különböző tag-ek következnek.
*
* @author  Mike van Riel <me@mikevanriel.com>
* @since 1.0
* @param int    $example  Ez egy szám paraméter
* @param string $example2 Ez egy szöveges paraméter
*/
```

Tartalom

Alapértelmezett PHP szintaxis

- Nyitó tag **<?php**
- Záró tag: **?>**
- A nyitó és záró tag közötti kódot a PHP értelmezőn keresztül fut
- Ami nincs közötte az közvetlen a kimenetre íródik
- A záró tag benyeli az utána közvetlen következő sortörést

PHP

```php
<?php
    $nev = "Luca";
    echo "Szia $nev ";
?>
jó újra látni.
```

Eredmény

```
Szia Luca jó újra látni.
```

Alapértelmezett PHP szintaxis

- Több PHP kódrészletet is elhelyezhetünk a kódban
- A létrehozott változók az egész fájlra érvényesek, nem csak egy adott blokkon belül

PHP

```
<?php
    $nev = "Luca";
?>
Szia <?php echo $nev; ?>
jó újra látni.
```

Eredmény

```
Szia Luca jó újra látni.
```

Alapértelmezett PHP szintaxis

- A nyitó és záró tagek tetszőlegesen tördelhetőek
- Az utasításokat pontosvesszővel zárjuk le
- A záró tag előtti pontosvessző elhagyható

PHP

```php
<?php $nev = "Luca" ?>

Szia <?php echo $nev ?> jó újra látni.
```

Eredmény

```
Szia Luca jó újra látni.
```

Alapértelmezett PHP szintaxis

- Amennyiben az egész kód egy PHP blokk a záró taget hagyjuk el!
- Ilyenkor az értelmező a fájl végéig php kódként értelmezi.
- Több blokk esetén az utolsó zárótaget hagyjuk el!
- Váratlan vagy felesleges whitespace karakterek okozhatnak gondot (include, verzió kezelés)

PHP

```php
<?php
$nev = "Luca";

echo "Szia ";
echo $nev;
echo " jó újra látni."
```

Eredmény

```
Szia Luca jó újra látni.
```

Alapértelmezett PHP szintaxis

- PHP kódblokkban a whitespace karakterek nem számítanak.
- A blokkon kívüli whitespace karakterek megjelennek a kimeneten.
- Kivéve a záró tag utána közvetlen következő sortörést.

PHP

```php
<?php $nev = "Luca"; echo "Szia ";


echo $nev; ?>


jó újra látni.
```

Eredmény

```
Szia Luca


jó újra látni.
```

Rövid PHP szintaxis

- Nyitó tag <?=
- Záró tag: ?>
- A nyitó és záró tag közötti kód kiírásra kerül!
- A záró tag benyeli az utána közvetlen következő sortörést.
- Az 5.4.0 verziótól kezdve **mindig** használható.

Rövidített szintaxis:

PHP
```
<?= "Hello World" ?>
```

A fenti kóddal megegyező kód alapértelmezett szintaxissal:

PHP
```
<?php echo "Hello World" ?>
```

Elavult rövidített szintaxis

A 7.0 előtt az alábbi nyitó és záró tag párosok is használhatóak voltak.

- `<% ... %>`
- `<%= ... %>`
- `<script language="php"> </script>`

Ezek használatát kerüljük!

Hibás szintaxis használat

Rossz példa!

```
<?php $nev = "Luca" ?>
Szia <?php $nev?>
```

Eredmény

```
Szia
```

- Gyakori hiba, hogy lemarad az `echo`

Rossz példa!

```
<?= $x = 5 ?>
```

Eredmény

```
5
```

- A rövidített szintaxis a benne található kódot kiírja a kimenetre. Így jelenhetnek meg váratlan szövegrészek.

Tartalom

Tartalom

Összefűzés (Concatenation)

- Az összefűzés műveletét szövegeken alkalmazzuk, és a két szövegrész egymás után leírva kapjuk meg.
- **PHP ban a "+" helyett a "." lesz a műveleti jel!**

PHP
```php
echo "Szia" . " Peti";
```

Eredmény
```
Szia Peti
```

PHP
```php
$nev = "Viki";
echo "Szia " . $nev;
```

Eredmény
```
Szia Viki
```

PHP
```php
echo "Szia " . $nev . " és Peti";
```

Eredmény
```
Szia Viki és Peti
```

Összefűzés (Concatenation) példa

PHP

```
echo␣1␣.␣1;
```

Eredmény

```
11
```

- Az 1-es szám mindkét esetben implicit átkonvertálódik szöveggé
- A két szöveg között összefűzés történik

Rossz példa!

```
echo␣1.1;
```

Eredmény

```
1.1
```

- Ez egy valós szám, nem összefűzés!

Tartalom

Bináris aritmetikai műveletek

Jel	Művelet	Példa
+	Összeadás	`5 + 3 // 8`
-	Kivonás	`8 - 2 // 6`
*	Szorzás	`8 * 2 // 6`
/	Osztás	`5 / 2 // 2.5`
%	Maradék képzés	`5 % 3 // 2`
**	Hatványozás	`2 ** 8 // 256`

`https://www.php.net/manual/en/language.operators.arithmetic.php`

Tartalom

Értékadó operátorok

Az egy darab egyenlőség jel (=) a klasszikus értékadó operátor.

Jel	Művelet	Példa	Eredmény
=	Értékadás	`$a = 65`	`// 65`
+=	Összeadás és értékadás	`$a += 5`	`// 70`
-=	Kivonás és értékadás	`$a -= 60`	`// 10`
*=	Szorzás és értékadás	`$a *= 5`	`// 50`
/=	Osztás és értékadás	`$a /= 10`	`// 5`
.=	Összefűzés és értékadás	`$a .= " db"`	`// 5 db`

`https://www.php.net/manual/en/language.operators.assignment.php`

Tartalom

Logikai operátorok

- Precedencia szerint csökkenő sorrendbe találhatóak a műveletek!
- A szöveges megadási forma alacsonyabb precedenciájú, mint az értékadás, így őket speciális esetben használjuk!
- A felsorolt operátorok rövidzár (short-circuit) kiértékelésűek.

Jel	Művelet	Példa
`!`	nem	`! ( 1 > 2 )`
`&&`	és	`1 > 2 && $a > $b`
`||`	vagy	`1 > 2 || $a > $b`
`AND`	és	`1 > 2 AND $a > $b`
`OR`	vagy	`1 > 2 OR $a > $b`

Linkek:

- PHP dokumentáció: Logiai operátorok
- Sulinet logikai kifejezések

Tartalom

Short-circuit evaluation

- Egy logikai kifejezés eredményéhez nem feltétlen kell teljesen kiértékelni
- Amennyiben egy "**ÉS**" kifejezés **bal oldalán hamis** érték áll, úgy a kifejezés érteke garantáltan **hamis** lesz.
- Amennyiben egy "**VAGY**" **bal oldalán igaz** érték áll, úgy a kifejezés érteke garantáltan **igaz** lesz.

Short-circuit evaluation példa (csak ÉS)

PHP

```php
$a = 1 > 2;      // hamis
$b = 5 != 6;     // igaz
$c = 3 == 10;    // hamis
$d = 1 <= 100;   // igaz
$e = 100 >= 1;   // igaz

var_dump($a && $b && $c && $d && $e);
```

Eredmény

```
bool(false)
```

- A `$a` értéke **hamis**
- Mivel minden változó között **ÉS** kapcsolat van, így már az az egy hamis az egész kifejezés értékét hamissá teszi
- A `$b`, `$c`, `$d` és `$e` értékeket nem vizsgálja

Short-circuit evaluation példa (csak VAGY)

PHP

```php
$a = 1 > 2;      // hamis
$b = 5 != 6;     // igaz
$c = 3 == 10;    // hamis
$d = 1 <= 100;   // igaz
$e = 100 >= 1;   // igaz

var_dump($a || $b || $c || $d || $e);
```

Eredmény

```
bool(false)
```

- A `$b` értéke **igaz**
- Mivel minden változó között **VAGY** kapcsolat van, így már az az egy igaz az egész kifejezés értékét igazzá teszi
- A `$c`, `$d` és `$e` értékeket nem vizsgálja

Short-circuit evaluation példa (ÉS, VAGY)

PHP

```php
$a = 1 > 2;      // hamis
$b = 5 != 6;     // igaz
$c = 3 == 10;    // hamis

var_dump($a && $c || $b && $c);
```

Eredmény

```
bool(false)
```

- `$a && $c` nem gyorsítható, értéke **hamis**
- A vagy jobb oldalát is ki kell értékelni
- A `$b && $c` nem gyorsítható, értéke **hamis**
- Végül a teljes kifejezést ki kell értékelni a végeredményhez

Tartalom

Összehasonlító operátorok (egyenlőség)

Jel	Művelet	Példa
==	Egyenlő (csak az érték)	`5 == 5 // igaz`
===	Azonos (típus és érték)	`5 === "5" // hamis`
!=	Nem egyenlő (csak az érték)	`5 != 5 // hamis`
<>	Nem egyenlő (csak az érték)	`5 <> 5 // hamis`
!==	Nem azonos (vagy a típus vagy az érték)	`5 !== "5" // igaz`

Linkek:

- Összehasonlító operátorok - PHP dokumentáció

Összehasonlító operátorok

Jel	Művelet	Példa
<	Kisebb	`1 < 2 // igaz`
>	Nagyobb	`1 > 2 // hamis`
<=	Kisebb vagy egyenlő	`1 <= 1 // igaz`
>=	Nagyobb vagy egyenlő	`5 >= 3 // igaz`

Linkek:

- Összehasonlító operátorok - PHP dokumentáció

Tartalom

Ternary operator (?:)

- A Ternary operátornak három operandusa van.
- Egy if else rövidített leírása.

```
$eredmeny = $szam % 2 == 0 ? "páros" : "páratlan"
```

`$szam % 2 == 0` — feltétel

`"páros"` — Érték, ha igaz

`"páratlan"` — Érték különben

PHP

```php
$szam = 8;
$eredmeny = $szam % 2 == 0 ? "páros" : "páratlan";
echo $eredmeny;
```

Eredmény

```
páros
```

Ternary operator (?:) rövidítve

Amennyiben a feltétel és az igaz ág megegyezik utóbbi elhagyható.

PHP
```php
$nev = "";
$eredmeny = $nev ?: "ismeretlen";
echo "Szia $eredmeny";
```

- A név üres , így logikailag hamis

Eredmény
```
Szia ismeretlen
```

PHP
```php
$nev = "Robi";
$eredmeny = $nev ?: "ismeretlen";
echo "Szia $eredmeny";
```

- A névnek van konkrét értéke, így logikailag igaz

Eredmény
```
Szia Robi
```

Null Coalescing Operator (??)

`$eredmeny = $a ?? $b`

- Amennyiben a `$a` értéke `null`, akkor `$b` lesz az eredmény
- Különben az eredmény `$a` lesz.

PHP

```php
$a = null;
$b = 5;
$eredmeny = $a ?? $b;

echo $eredmeny; // 5
```

Null Coalescing Operator

- A Null Coalescing operátorok **egymásba ágyazhatóak**.
- Ilyenkor az első nem null érték lesz a végeredmény.

PHP

```php
$a = null;
$b = null;
$c = "x";
$d = "y";

$eredmeny = $a ?? $b ?? $c ?? $d;

echo $eredmeny; // x
```

Tartalom

Tartalom

Változók elnevezése

- Minden változó **$** jellel kezdődik!
- Tartalmazhat aláhúzást.
- Tartalmazhat számot.
- **Számmal nem kezdődhet!**
- Tartalmazhat kis- és nagybetűt is.
- **Érzékeny a kis- és nagybetűkre!**

Linkek:

- `https://www.php.net/manual/en/language.variables.basics.php`
- `https://www.w3schools.com/php/php_variables.asp`

Változó elnevezési példák

Helyes elnevezések:

PHP
```php
$a = 1;
$A = 2;
$_a = 3;
$a4 = 4;
```

Lehetséges, de inkább kerüljük az ékezetes betűk használatát!

PHP
```php
$á = 5;
```

Leggyakrabban a dollár jelet felejtik le.

Rossz példa!
```php
a = 1;
```

Számmal nem kezdődhet a változó neve!

Rossz példa!
```php
$1a = 2;
```

Nem tartalmazhat szóközt!

Rossz példa!
```php
$ b = 3;
```

Szuperglobális változók

A szuperglobális változók bármelyik hatókörből (scope) elérhetőek.

- `$GLOBALS` Az összes szuperglobálist tartalmazza.
- `$_SERVER` Szerverrel és futtatással kapcsolatos információk.
- `$_GET` GET paraméterek.
- `$_POST` POST paraméterek.
- `$_FILES` Fájlfeltöltésnél alkalmazandó.
- `$_COOKIE` Süti adatok.
- `$_SESSION` A munkamenetben tárolt információk.
- `$_REQUEST` A GET, POST és COOKIE adatok asszociatív tömbje.
- `$_ENV` Környezeti változók.

Link: PHP dokumentáció: Szuperglobális változók

Előre definiált változók

- Az összes szuperglobális változó.
- `$php_errormsg` A legutóbbi hibaüzenet.
- `$http_response_header` A válasz fejléc adatai.
- `$argc` A szkript által kapott argumentumok száma.
- `$argv` A szkript által kapott argumentumok tömbje.

Link: PHP dokumentáció: Előredefiniált változók

Tartalom

Típusok

A PHP 10 alaptípust támogat

- Skaláris adattípusok
  - boolean
  - integer
  - float
  - string
- Összetett adattípusok
  - array
  - object
  - callable (meghívható függvények)
  - iterable (pszeudó típus, bejáró)
- Speciális típusok
  - resource (hivatkozás külső forrásra, pl.: fájl)
  - NULL

Linkek:

- Típusok bevezető - PHP dokumentáció

Tartalom

Logikai értékek: a boolean típus

- Logikai értékeket tárolhatunk el benne.
- Csak igaz (`true`) vagy hamis (`false`) értékeket vehet fel.
- A megadása kis- és nagybetűkre **nem** érzékeny.

PHP

```php
$b = true;
$b = TRUE;
$b = True;
$b = TruE;

$b = false;
```

Tartalom

Egész számok: az integer típus

- Egész számokat tárolhatunk el benne.
- Mérete: `PHP_INT_SIZE` (rendszer függő)
- Legnagyobb eltárolható érték:
  - `PHP_INT_MAX`
  - 32-bites rendszeren kb. 2 milliárd
  - 64-bites rendszeren: 9223372036854775807
- Legkisebb eltárolható érték:
  - `PHP_INT_MIN`
  - 32-bites rendszeren kb. -2 milliárd
  - 64-bites rendszeren: -9223372036854775808

Egész szám megadása decimális (10-es) számrendszerben.

PHP

```php
$a = 42;
$b = -8;
$c = 1_499_000; // PHP 7.4.0 verziótól kezdve
```

Egész számok: az integer típus

Egész szám megadása oktális (8-as) számrendszerben.

PHP
```php
$a = 052;   // értéke: 42
$b = -010;  // értéke: -8
```

Egész szám megadása hexadecimális (16-os) számrendszerben.

PHP
```php
$a = 0x2A;   // értéke: 42
$b = -0x8;  // értéke: -8
```

Egész szám megadása bináris (2-es) számrendszerben. (PHP 5.4.0.-tól)

PHP
```php
$a = 0b101010;   // értéke: 42
$b = -0b1000;  // értéke: -8
```

Valós számok: a float típus

- A valós számokat lebegőpontosan tárolja.
- Tárolás módja:
  IEEE Standard for Floating-Point Arithmetic (IEEE 754)
- Mérete rendszer függő
- **Tizedes pontot kell alkalmazni!**

Valós szám megadása

PHP

```php
$a = 4.2;         // 4.2
$b = 1.5e3;       // 1500
$b = 1.5e-3;      // 0.0015
$c = 5E-10;       // 0.0000000005
$c = 5E10;        // 50 000 000 000
$d = 1_499.123; // PHP 7.4.0 verziótól kezdve
```

Tartalom

Típuskényszerítés

- PHP-ban hasonlóan a C típusú nyelvekhez a kikényszerítendő típust zárójelekben kell az érték elé helyezni.

PHP

```php
<?php
$i = 50;
$b = (bool) $i;      // true  (boolean)
$c = (bool) "";      // false (boolean)

$f = 10.5;
$x = (int) $f;       // 10 (int)
?>
```

Típuskényszerítés

- (int), (integer) - cast to integer
- (bool), (boolean) - cast to boolean
- (float), (double), (real) - cast to float
- (string) - cast to string
- (array) - cast to array
- (object) - cast to object
- (unset) - cast to NULL !!PHP 7.2.0!!

Valós számok pontossága

- A decimális (10-es) számrendszerben jól eltárolható értékek: `0.1`, `0.7`
- A bináris (2-es) számrendszerre átváltva veszít a pontosságából

PHP
```php
$x = 1 + 7 ;
echo $x;                    // 8
echo (int) $x;              // 8
```

PHP
```php
$x =  ( 0.1 + 0.7 ) * 10;
echo $x;                    // 8 ahogy vártuk
echo (int) $x;              // 7
```

- https://en.wikipedia.org/wiki/IEEE_754
- https://floating-point-gui.de

Hamis értékek

- A logikai `false`
- A következő egész számok `0` és `-0`
- A következő valós számok `0.0` és `-0.0`
- Az üres szöveg `""`
- Az egy darab 0-t tartalmazó szöveg: `"0"`
- Az üres tömb: `[]`
- A NULL típus: `null`
- A még be nem állított változók
- SimpleXML objects created from empty tags

Minden mást IGAZ-nak kell tekinteni.

Hamis értékek

PHP

```php
<?php
var_dump((bool) "");        // bool(false)
var_dump((bool) 1);         // bool(true)
var_dump((bool) -2);        // bool(true)
var_dump((bool) "foo");     // bool(true)
var_dump((bool) 2.3e5);     // bool(true)
var_dump((bool) array(12)); // bool(true)
var_dump((bool) array());   // bool(false)
var_dump((bool) "false");   // bool(true)
?>
```

Tartalom

Szöveg: a string típus

- A string típus egy byte-os karakterek sorozatát takarja.
- Így 256 különböző karaktert tud eltárolni.
- Nincs natív unicode támogatás
- A 32 bites rendszereken és PHP 7.0.0 előtt max méret: 2 GB
- Megadásának módjai
  - aposztróf
  - idézőjel
  - heredoc
  - nowdoc

Escaped characters

Minta	Röv.	Megnevezés	Jelentés
`"\n"`	LF	linefeed	soremelés
`"\r"`	CR	carriage return	kocsi vissza
`"\t"`	HT	horizontal tab	(vízszintes) tabulátor
`"\e"`	ESC	escape	
`"\\"`	\	backslash	visszaperjel
`"\$"`	$	dollar sign	dollár
`"\""`	"	double-quote	idézőjel

A string literál megadása idézőjellel (")

A legegyszerűbb megadási forma:

PHP
```php
$a = "Burger King";
echo $a;
```

Eredmény
```
Burger King
```

PHP
```php
echo "Subway";
```

Eredmény
```
Subway
```

Aposztróf esetén nincs szükség escape karakterre:

PHP
```php
$b = "McDonald's";
echo $b;
```

Eredmény
```
McDonald's
```

A string literál megadása idézőjellel (")

Az escape karakter nem csak az aposztróf jelhez használható, hanem sortöréshez is.

PHP
```php
echo "KFC\nTaco Bell";
```

Eredmény
```
KFC
Taco Bell
```

A backslash karaktert itt már kell escapelni:

Rossz példa!
```php
echo "C:\tanar";
```

Eredmény
```
C:    anar
```

A változókat behelyettesíti:

PHP
```php
$w = "Wendy's";
echo "A $w finom.";
```

Eredmény
```
A Wendy's finom.
```

A string literál megadása aposztróffal (')

A legegyszerűbb megadási forma:

PHP
```php
$a = 'Burger King';
echo $a;
```

Eredmény
```
Burger King
```

PHP
```php
echo 'Subway';
```

Eredmény
```
Subway
```

Az escape karakter a "\" jel:

PHP
```php
$b = 'McDonald\'s';
echo $b;
```

Eredmény
```
McDonald's
```

A string literál megadása aposztróffal (')

Az escape karakter csak az aposztróf jelhez használható, **sortörésre már nem**.

Rossz példa!

```
echo 'KFC\nTaco Bell';
```

Eredmény

```
KFC\nTaco Bell
```

A backslash karaktert itt nem kell escapelni:

PHP

```
echo 'C:\tanar';
```

Eredmény

```
C:\tanar
```

Nem tud behelyettesíteni változót:

Rossz példa!

```
echo '$a';
```

Eredmény

```
$a
```

Változók behelyettesítése egyszerű szintaxis

- Idézőjelek és heredoc használatakor a szövegbe behelyettesíti a változókat.
- Amint talál egy **$** jelet úgy a lehető leghosszabb érvényes változónevet próbálja meg behelyettesíteni.

PHP

```php
$nev = "Peti";
echo "Hello $nev!";
```

Eredmény

```
Hello Peti!
```

Rossz példa!

```php
$osztaly = "10.";
echo "$osztalyA osztály";
```

Eredmény

```
Ismeretlen változó:
$osztalyA
```

PHP

```php
$a = 2;
$b = "B";
echo "$a$b osztály";
```

Eredmény

```
2B osztály
```

Változók behelyettesítése komplex szintaxis

- Idézőjelek és heredoc használatakor a szövegbe behelyettesíti a változókat.
- Tetszőleges változó, tömbelem, objektum tulajdonság behelyettesíthető, ha **kapcsos zárójelek közé tesszük**.

PHP
```php
$nev = "Peti";
echo "Hello {$nev}!";
```

Eredmény
```
Hello Peti!
```

PHP
```php
$osztaly = "10.";
echo "{$osztaly}A osztály";
```

Eredmény
```
10.A osztály
```

PHP
```php
$t = ["A","B","C"];
echo "1:{$t[0]}\n2:{$t[1]}";
```

Eredmény
```
1:A
2:B
```

Hozzáférés a szöveg karaktereihez

- A string típusra gondolhatunk úgy, mint egy 0-tól indexelt tömbre.
- A szögletes zárójelekkel tetszőleges karaktert elérhetünk.
- A (PHP 7.1.0) negatív értékeket is elfogadja, ekkor hátulról indul az indexelés .

PHP

```php
$nev = "Laci";
echo "{$nev[0]}\n";
echo "{$nev[1]}\n";
echo "{$nev[-1]}\n";
```

Eredmény

```
L
a
i
```

A multibyte-os karakterek esetén az eredmény értelmezhetetlen lehet:

Rossz példa!

```php
$nev = "László";
echo "{$nev[0]}\n";
echo "{$nev[1]}\n";
echo "{$nev[-1]}\n";
```

Eredmény

```
L
?
?
```

Tartalom

Típus vizsgálat

Az alábbi függvények logikai értéket adnak vissza.

fv	vizsgálat
`is_array()`	Tömb
`is_bool()`	Logikai
`is_callable()`	Meghívható-e, mint függvény
`is_float()`	Lebegőpontos szám
`is_int()`	Egész szám
`is_null()`	Null érték
`is_numeric()`	Szám, vagy szövegként tárolt szám
`is_object()`	Objektum
`is_scalar()`	integer, float, string vagy boolean
`is_string()`	Szöveg

Érték kiolvasása típus szerint

Az alábbi függvények a változóból kinyerik a megadott típusú értéket.

fv	típus
`boolval()`	Logikai
`doubleval()`	Valós szám
`floatval()`	Valós szám
`intval()`	Egész szám
`strval()`	Szöveg

floatval()

- Amennyiben egy szöveg számmal kezdődik, úgy annak az értékét adja vissza.
- Egyéb esetben 0-t ad vissza.
- Nulla esetén érdemes ellenőrizni, hogy nem e nulla volt az eredeti szöveg.

PHP

```php
$s = "10.5 kg";
$mennyiseg = floatval($s);
echo $mennyiseg;
```

Eredmény

```
10.5
```

Tartalom

Tartalom

Elágazás: if

Egyszerű elágazás:

```php
<?php
if ( $x > 10 )
{
    echo "$x nagyobb, mint 10.";
}
```

Alternatív szintaxis:

```php
<?php
if ( $x > 10 ):
    echo "$x nagyobb, mint 10.";
endif;
```

Elágazás: if ... else

PHP

```php
<?php
$szam = 10;
if( $szam % 2 == 0 )
{
    echo "A szám ({$szam}) páros.";
}
else
{
    echo "A szám ({$szam}) páratlan.";
}
```

Eredmény

```
A szám (10) páros.
```

Elágazás: if ... else alternatív szintaxis

PHP

```php
<?php
$szam = 10;
if( $szam % 2 == 0 ):
    echo "A szám ({$szam}) páros.";
else:
    echo "A szám ({$szam}) páratlan.";
endif;
```

Eredmény

```
A szám (10) páros.
```

Elágazás: if ... elseif

PHP

```php
<?php
$a = 1; $b = 2;
if( $a > $b )
{
    echo "A nagyobb, mint B";
}
elseif ( $a < $b )
{
    echo "B nagyobb, mint A";
}
else
{
    echo "A és B egyenlőek."
}
```

Elágazás: if ... elseif alternatív szintaxis (hibás)

Rossz példa!

```php
<?php
$a = 1;  $b = 2;
if( $a > $b ):
    echo "A nagyobb, mint B";
else if ( $a < $b ):
    echo "B nagyobb, mint A";
else:
    echo "A és B egyenlőek."
endif;
```

https://www.php.net/manual/en/control-structures.elseif.php

Elágazás: if ... elseif alternatív szintaxis

PHP

```php
<?php
$a = 1;  $b = 2;
if( $a > $b ):
    echo "A nagyobb, mint B";
elseif ( $a < $b ):
    echo "B nagyobb, mint A";
else:
    echo "A és B egyenlőek."
endif;
```

Eredmény

```
B nagyobb, mint A
```

Többirányú elágazás: switch

PHP

```php
<?php
switch ($valtozo) {
    case "alma":
        echo "A változó értéke: alma";
        break;
    case 1:
    case 2:
        echo "A változó értéke egy szám";
        break;
    default:
        echo "Minden más esetében ide jut.";
        break;
}
```