Backend 12

Függvények és hatókör

Rostagni Csaba

2025. október 13.

Tartalom

Felhasználó által definiált függvények

- A függvényeket a `function` kulcsszóval kell bevezetni
- A bemeneti paraméterek típusa megadható opcionálisan
- A kimenet típusa megadható opcionálisan
- Anonymous functions / closures
- Arrow functions (PHP 7.4-től)

Függvény

PHP
```php
function hello_world()
{
    echo "Hello World";
}
```

- A függvényeket a `function` kulcsszóval kell bevezetni
- Ezt követi a függvény neve
- Most nincs paraméter, így üres zárójellel végződik
- A függvény tartalma kapcsoszárójelek között helyezkedik el
- Ha nincs `return`, akkor `NULL` lesz a visszatérés értéke

PHP
```php
hello_world();
```

- A függvényt a nevével lehet meghívni, utána zárójel pár következik

Eredmény
```
Hello World
```

Függvények paraméterekkel

PHP
```php
function hello($nev)
{
    echo "Hello {$nev}";
}
```

- A függvénynek egy (formális) paramétere van

PHP
```php
hello("World");
```

Eredmény
```
Hello World
```

- A függvény argumentuma (tényleges paramétere) lehet literál

PHP
```php
$nev = "Peti";
hello($nev);
```

Eredmény
```
Hello Peti
```

- A függvény argumentuma (tényleges paramétere) lehet változó

Függvények alapértelmezett értékkel

PHP
```php
function hello($nev = "World")
{
    echo "Hello {$nev}";
}
```

- A függvény paraméterei kaphatnak alapérelmezett értéket

PHP
```php
hello("Jocó");
```

Eredmény
```
Hello Jocó
```

PHP
```php
hello();
```

Eredmény
```
Hello World
```

- A függvény paraméter nélkül a nevet `World`-nek veszi

Függvény visszatérési értékkel

PHP
```php
function osszead($a,$b)
{
    return $a + $b;
}
```

- Több paraméter esetén azokat vesszővel kell elválasztani
- A visszatérési értéket adó kifejezés a `return` kulcsszó után szerepel

PHP
```php
echo osszead(3,8);
```

Eredmény
```
11
```

- Több argumentum esetén azokat vesszővel kell elválasztani

Argumentum átadása referenciaként

PHP

```php
function negyzet(int &$szam)
{
    $szam = $szam * $szam;
}

$szam = 5;
negyzet($szam);
echo $szam;
```

- Alapértelmezetten az argumentumok **érték szerint** kerülnek átadásra
  - A függvényben történt módosítások a kinti változóra érték szerint átadáskor nem lesznek érvényesek
- Az & jellel jelzi a **referencia szerinti** átadást
  - A függvényben történt módosítások a kinti változóra referencia szerinti átadáskor érvényesülnek

Függvények visszatérési típussal

PHP
```php
function osszead(float $a,float $b):float
{
    return $a + $b;
}
```

- A paraméter előtt kell a típusát meghatározni (`float $a`)
- A függvény visszatérési értéke a `:` után meghatározott `float` lesz

PHP
```php
echo osszead(7.5,12);
```

Eredmény
```
19.5
```

- A `7.5` valós érték, így a `float` típusnak megfelel
- A `12` egész szám, így itt implicit konverzió hajtódik végre

Linkek:

- Típus delkarálás - PHP dokumentáció

Amikor a type hint pontatlan

PHP

```php
function osszead(int $a,int $b):int
{
    return $a + $b;
}
```

Rossz példa!

```php
echo osszead(7.5,12);
```

Eredmény

19

- A 7.5 nem egész szám, mégis elfogadta
- Az értéket 7-nek vette

Linkek:

- PHP 7 Type Hinting: Inconsistencies and Pitfalls

Szigorúan típusos mód

PHP
```php
declare(strict_types=1);
function osszead(int $a,int $b):int
{
    return $a + $b;
}
```

- A `declare()` a szkript legelején kell, hogy álljon
- az egész szkriptre érvényes

Rossz példa!
```php
echo osszead(7.5,12);
```

- A `7.5` valós érték, így nem fogadja el

Linkek:

- strict_types - phptutorial.net

TypeError

PHP

```php
function osszead(float $a,float $b):float
{
    return $a + $b;
}
```

Rossz példa!

```php
echo osszead("Hello","World");
```

Eredmény

```
Fatal error: Uncaught TypeError: osszead():
Argument #1 ($a) must be of type float, string given ...
Stack trace:
#0 osszead2.php(9): osszead('Hello', 'World') ...
```

- `\TypeError` kivételt dob

Tartalom

Scope/Hatókör

- A változó a hatókörében érhető el
- Hatókörök php-ban:
  - **global**
    - a függvényen, osztályokon kívül elérhető
    - a global kulcsszóval függvényen belül is létrehozható
  - **local**
    - a kódblokkon belül érvényes
    - a blokk végére érve megszűnik
  - **static**
    - a kódblokkon belül érvényes
    - a blokk végére érve **nem** szűnik meg
    - a static kulsszóval kell bevezetni

Infó

A **szuperglobális** változók `$GLOBAL`, `$_GET`, `$_POST`, . . . a függvényeken kívül és belül is elérhetőek.

Global scope példa

PHP

```php
error_reporting(E_ALL);
ini_set("display_errors", 1);
```

- Hibaüzenetek, figyelmeztetések bekapcsolása

Rossz példa!

```php
$x = 1;
function f()
{
    var_dump(isset($x));
}
f();
```

Eredmény

```
bool(false)
```

- A függvényen kívül (global scope) létrehozott változó nem érhető el belülről (local scope)
- Az `isset()` függvény hamis értéket ad, de nincs figyelmeztetés

Global scope példa

Rossz példa!

```php
$x = 1;
function f()
{
    var_dump(get_debug_type($x))
}
f();
```

Eredmény

```
Warning: Undefined variable $x ...
```

- Itt már figyelmeztet: a `$x` változó nincs definiálva

Eredmény

```
string(4) "null"
```

- A `get_debug_type()` szerint a `$x` változó típusa `null`

Global scope példa

Rossz példa!

```php
$x = 1;
function f()
{
    var_dump($x);
}
f();
```

Eredmény

```
Warning: Undefined variable $x ...
```

- Itt már figyelmeztet: a `$x` változó nincs definiálva

Eredmény

```
NULL
```

- A függvényen belül a nem létező `$x` értéke `NULL` lesz

Local scope példa

PHP
```php
error_reporting(E_ALL);
ini_set("display_errors", 1);
```

- Hibaüzenetek, figyelmeztetések bekapcsolása

Rossz példa!
```php
function f() {
    $x = 1;
}
f();
var_dump(isset($x));
```

Eredmény
```
bool(false)
```

- A függvényen belül (local scope) létrehozott változó nem érhető el kívülről (global scope)
- Az `isset()` függvény hamis értéket ad, de nincs figyelmeztetés

Local scope példa

Rossz példa!

```php
function f() {
    $x = 1;
}
f();
var_dump(get_debug_type($x));
```

Eredmény

```
Warning: Undefined variable $x ...
```

- Itt már figyelmeztet: a `$x` változó nincs definiálva

Eredmény

```
string(4) "null"
```

- A `get_debug_type()` szerint a `$x` változó típusa `null`

Local scope példa

Rossz példa!

```php
function f() {
    $x = 1;
}
f();
var_dump($x);
```

Eredmény

```
Warning: Undefined variable $x ...
```

- Itt már figyelmeztet: a `$x` változó nincs definiálva

Eredmény

```
NULL
```

- A függvényen belül a nem létező `$x` értéke `NULL` lesz