

# Backend 12

## Függvények és hatókör

Rostagni Csaba

2024. október 15.

# Tartalom

- 1 Függvények
- 2 Scope, hatókör

# 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

```
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 kapcsoszárojelek között helyezkedik el
- Ha nincs **return**, akkor **NULL** lesz a visszatérés értéke

PHP

```
hello_world();
```

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

Eredmény

Hello World

# Függvények paraméterekkel

PHP

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

- A függvénynek egy (formális) paramétere van

PHP

```
hello("World");
```

Eredmény

```
Hello World
```

- A függvény argumentuma (tényleges paramétere) lehet literál

PHP

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

Eredmény

```
Hello Peti
```

- A függvény argumentuma (tényleges paramétere) lehet változó

# Függvények alapértelmezett értékkel

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

PHP

- A függvény paraméterei kaphatnak alapértelmezett értéket

```
hello("Jocó");
```

PHP

```
Hello Jocó
```

Eredmény

```
hello();
```

PHP

```
Hello World
```

Eredmény

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

# Függvény visszatérési értékkel

PHP

```
function összead($a,$b)
{
    return $a + $b;
}
```

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

PHP

```
echo összead(3,8);
```

Eredmény

11

- Több argumentum esetén azokat vesszővel kell elválasztani

# Argumentum átadása referenciaként

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

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

PHP

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

```
echo összead(7.5,12);
```

PHP

19.5

Eredmény

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

Linkek:

- Típus deklarálás - PHP dokumentáció

# Amikor a type hint pontatlan

PHP

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

```
echo összead(7.5,12);
```

Rossz példa!

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

```
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

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

Rossz példa!

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

Linkek:

- `strict_types` - [phptutorial.net](https://www.phptutorial.net/strict-types/)

# TypeError

PHP

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

Rossz példa!

```
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

- 1 Függvények
- 2 Scope, hatókör

# 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

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

- Hibaüzenetek, figyelmeztetések bekapcsolása

Rossz példa!

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

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

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

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

- Hibaüzenetek, figyelmeztetések bekapcsolása

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

Rossz példa!

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!

```
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

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

Rossz példa!

Warning: Undefined variable \$x ...

Eredmény

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

NULL

Eredmény

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