Backend 12

Speciális függvények és szövegfüggvények

Rostagni Csaba

2024. november 4.

Tartalom

Anonymous functions / Nételen függvények

- Létrehozhatóak olyan függvények név nélkül is
- A háttérben egy `Closure` objektumot hoz létre
- A névtelen függvények eltárolhatóak változóban, azok függvényként használhatóak

Névtelen függvény példa - növekvő sorrenbe rendezés

PHP

```php
$x = [11,5,67,34]; // 4 elemű tömb

usort($x, function($a,$b){
    return $a - $b;
});
print_r($x);
```

Eredmény

```
Array (
    [0] => 5
    [1] => 11
    [2] => 34
    [3] => 67
)
```

- A `usort()` rendezi az első paraméterként kapott tömböt
- Az első paraméter cím szerint kerül átadásra, így a `$x` értéke változik
- A második paraméter `callable` típusú
- A `$a-$b` értéke
  - `negatív`, ha a `$b` értéke a nagyobb
  - `0`, ha a `$a` és `$b` értéke megegyezik
  - `pozitív`, ha a `$a` értéke a nagyobb

Névtelen függvény példa - csökkenő sorrenbe rendezés

PHP

```php
$x = [11,5,67,34]; // 4 elemű tömb

usort($x, function($a,$b){
    return $b - $a;
});
print_r($x);
```

Eredmény

```
Array (
    [0] => 67
    [1] => 34
    [2] => 11
    [3] => 5
)
```

- A `usort()` rendezi az első paraméterként kapott tömböt
- Az első paraméter cím szerint kerül átadásra, így a `$x` értéke változik
- A második paraméter `callable` típusú
- A `$b-$a` értéke
  - `negatív`, ha a `$a` értéke a nagyobb
  - `0`, ha a `$a` és `$b` értéke megegyezik
  - `pozitív`, ha a `$b` értéke a nagyobb

Szülő hatókörben lévő változó használata

Rossz példa!

```php
$bonusz = 30_000;

$fizetes = function(int $ora, int $oraber) {
    return $ora * $oraber + $bonusz;
};

$osszeg = $fizetes(165,2000);

echo number_format($osszeg,0,",", " ") . " Ft\n";
```

Eredmény

```
Warning: Undefined variable $bonusz in ...
330 000 Ft
```

- A `$bonusz` változó
  - a függvényen kívül lett létrehozva,
  - a globális hatókörben van,
  - a függvényben nem elérhető.
- Az eredmény így hibás lesz!

Szülő hatókörben lévő változó használata

PHP

```php
$bonusz = 30_000;

$fizetes = function(int $ora, int $oraber) use($bonusz){
    return $ora * $oraber + $bonusz;
};

$osszeg = $fizetes(165,2000);

echo number_format($osszeg,0,",", " ") . " Ft\n";
```

Eredmény

```
360 000 Ft
```

- A `use` után zárójelben a globális hatókörben levő változókat át lehet adni a névtelen függvénynek

Tartalom

Arrow functions / Nyíl függvények

- A PHP 7.4.0-ban lett bevezetve
- A névtelen függvényeket rövidíti le
- A nyíl függvényeket `function` helyett `fn` vezeti be
- A háttérben egy `Closure` objektumot hoz létre
- A szülő hatókör változóit automatikusan át veszi érték szerint

Általános meghatározása

```
fn (paraméter lista) => kifejezés
```

Linkek:

- Arrow functions - PHP dokumentáció

Nyíl függvény példa - növekvő sorrenbe rendezés

PHP

```php
$x = [11,5,67,34]; // 4 elemű tömb

usort($x, fn($a,$b) => $a-$b);
print_r($x);
```

Eredmény

```
Array (
    [0] => 5
    [1] => 11
    [2] => 34
    [3] => 67
)
```

- A `usort()` rendezi az első paraméterként kapott tömböt
- Az első paraméter cím szerint kerül átadásra, így a `$x` értéke változik
- A második paraméter `callable` típusú
- A `$a-$b` értéke
  - `negatív`, ha a `$b` értéke a nagyobb
  - `0`, ha a `$a` és `$b` értéke megegyezik
  - `pozitív`, ha a `$a` értéke a nagyobb

Nyíl függvény példa - csökkenő sorrenbe rendezés

PHP

```php
$x = [11,5,67,34]; // 4 elemű tömb

usort($x, fn($a,$b) => $b - $a);
print_r($x);
```

Eredmény

```
Array (
    [0] => 67
    [1] => 34
    [2] => 11
    [3] => 5
)
```

- A `usort()` rendezi az első paraméterként kapott tömböt
- Az első paraméter cím szerint kerül átadásra, így a `$x` értéke változik
- A második paraméter `callable` típusú
- A `$b-$a` értéke
  - `negatív`, ha a `$a` értéke a nagyobb
  - `0`, ha a `$a` és `$b` értéke megegyezik
  - `pozitív`, ha a `$b` értéke a nagyobb

Tartalom

Az ASCII karakterkódolás

- Minden karakternek egy számot feleltetünk meg.
- Az adattárolás 1 byte-on, azaz 8 bit-en történik.
- Így 256 különböző karaktert tud eltárolni.
  - 0-31 és a 127 a nem nyomtatható karakterek.
  - 32-126 nyomtatható karakterek:
    Az angol abc kis- és nagybetűi, számok, írásjelek
  - 128-255 Kibővített karakterek.
    A magyar ékezetes betűk a Latin-2 karakterkészletben.

ASCII karakterkódolás példák (ISO/IEC 8859-2)

karakter	decimális	bináris
ESC	27	0001 1011
!	33	0100 0001
0	48	0011 0000
1	49	0011 0001
A	65	0100 0001
B	66	0100 0010
a	97	0110 0001
Ą	161	1010 0001
Á	193	1100 0001
Ă	195	1100 0011
á	225	1110 0001

Az UTF-8 karakterkódolás

- Unicode karakterek kódolási formája.
- 8-bit Unicode Transformation Format
- Az adattárolás változó számú byte-on történik.
- Helytakarékos.
- ASCII-vel felülről kompatibilis
- Az ASCII első 128 karaktere 1 byte-on van eltárolva. (0-val kezdve)
- Prefix kód

https://www.johndcook.com/blog/2019/09/09/how-utf-8-works/

UTF-8 karakterkódolás példák

karakter	bináris	byte
ESC	00011011	1
0	00110000	1
A	01000001	1
a	01100001	1
Á	11000011:10000001	2
á	11000011:10100001	2
ฉ *	11100000:10111000:10001001	3
	11110000:10011111:10001101:10000110	4

* Thai karakter

- Az első egyes után következő egyesek adják meg, hogy a következő hány byte tartozik még az adott karakterhez.
- Az "10" kezdetű byte-ok nem lehetnek új karakterek kezdete.

árvíztűrőtükörfúrógép

Ugyanaz a szöveg különböző kódolásokkal

UTF-8

árvíztűrőtükörfúrógép

ASCII (8859-2)

ĂĄrvĂztŰrŐtĂźkĂśrfĂşrĂłgĂŠp

bináris

11000011 10100001 01110010 01110110 11000011 10101101 01111010
01110100 11000101 10110001 01110010 11000101 10010001 01110100
11000011 10111100 01101011 11000011 10110110 01110010 01100110
11000011 10111010 01110010 11000011 10110011 01100111 11000011
10101001 01110000

https://onlineutf8tools.com/convert-utf8-to-binary

Tartalom

Szövegfüggvények

- A PHP nem támogatja natívan az utf-8 karakterkódolást.
- Bizonyos függvények működhetnek így is
- A legtöbb szövegfüggvényből létezik multibyte-os változat, ami helyesen kezeli az utf-8 kódolású szöveget karaktereket.
- a `mbstring` csomag tartalmazza a multibyte-os függvényeket
- Ezeknek a függvényeknek a neve `mb_` karakterekkel kezdődik

Linkek:

- Multibyte String Functions - PHP dokumentáció

mb_internal_encoding

```php
mb_internal_encoding("UTF-8");
```

- Meghatározza a multibyte-os függvényeknél, hogy milyen karakterkódolást (encoding) alkalmazzon a működése során

Linkek:

- mb_internal_encoding() - PHP dokumentáció
- PHP's internal character encoding - texelate.co.uk

Tartalom

Hossz meghatározása

```
mb_strlen(string $string, ?string $encoding = null): int
```

- A `$string` az a szöveg, aminek a hosszára kíváncsiak vagyunk.
- Az `$encoding` a karakterkódolás. Nem kötelező kitölteni.
- A **visszatérési értéke** a szöveg hossza.

A strlen() és az mb_strlen() összehasonlítása

PHP
```php
$szoveg = "árvíztűrőtükörfúrógép";
```

Rossz példa!
```php
echo strlen($szoveg);
```

Eredmény
```
30
```

PHP
```php
echo mb_strlen($szoveg);
```

Eredmény
```
21
```

- A `strlen()` a multibyte karaktereket, mint a magyar ékezetes karakterek többnek számolja,
- ellenben az `mb_strlen()` helyesen egy karakternek számítja.
- A 9 ékezetes karakter 1 helyett 2 bájtot foglal, így jön ki a különbség a két hossz között

Tartalom

mb_strtoupper()

```
mb_strtoupper(string $string, ?string $encoding = null):
↪  string
```

Az `mb_strtoupper()` a paraméterül kapott szöveget nagybetűssé alakítja át.

- A `$string` az eredeti szöveg
- A `$encoding` karakterkódolás (opcionális)
- A **visszatérési értéke** a nagybetűssé alakított szöveg
- Az eredeti szöveget nem módosítja
- Létezik nem multibyte-os verzió `strtoupper()` néven

Linkek:

- mb_strtoupper() - PHP dokumentáció

mb_strtolower()

```
mb_strtolower(string $string, ?string $encoding = null):
↪  string
```

Az `mb_strtolower()` a paraméterül kapott szöveget kisbetűssé alakítja át.

- A `$string` az eredeti szöveg
- A `$encoding` karakterkódolás (opcionális)
- A **visszatérési értéke** a kisbetűssé alakított szöveg
- Az eredeti szöveget nem módosítja
- Létezik nem multibyte-os verzió `strtolower()` néven

Linkek:

- mb_strtolower() - PHP dokumentáció

mb_convert_case()

```
mb_convert_case(string $string, int $mode, ?string
↪  $encoding = null): string
```

Az `mb_convert_case()` a paraméterül kapott szöveget átalakítja

- A `$string` az eredeti szöveg
- A `$mode` olyan konstans, ami az átalakítás módját határozza meg
- A `$encoding` karakterkódolás (opcionális)
- A **visszatérési értéke** az átalakított szöveg
- Az eredeti szöveget nem módosítja

Linkek:

- mb_convert_case() - PHP dokumentáció

MB_CASE_UPPER vs MB_CASE_UPPER_SIMPLE

PHP

```php
$str = "árvÍz tűRŐ tükör FÚRÓ gép";
echo mb_convert_case($str, MB_CASE_UPPER) . "\n";
echo mb_convert_case($str, MB_CASE_UPPER_SIMPLE) . "\n";
```

Eredmény

```
ÁRVÍZ TŰRŐ TÜKÖR FÚRÓ GÉP
ÁRVÍZ TŰRŐ TÜKÖR FÚRÓ GÉP
```

PHP

```php
$str = "Gößmann";
echo mb_convert_case($str, MB_CASE_UPPER) . "\n";
echo mb_convert_case($str, MB_CASE_UPPER_SIMPLE) . "\n";
```

Eredmény

```
GÖSSMANN
GÖßMANN
```

MB_CASE_LOWER vs MB_CASE_LOWER_SIMPLE

PHP

```php
$str = "árvÍz tűRŐ tükör FÚRÓ gép";
echo mb_convert_case($str, MB_CASE_LOWER) . "\n";
echo mb_convert_case($str, MB_CASE_LOWER_SIMPLE) . "\n";
```

Eredmény

```
árvíz tűrő tükör fúró gép
árvíz tűrő tükör fúró gép
```

PHP

```php
$str = "Gößmann";
echo mb_convert_case($str, MB_CASE_LOWER) . "\n";
echo mb_convert_case($str, MB_CASE_LOWER_SIMPLE) . "\n";
```

Eredmény

```
gößmann
gößmann
```

MB_CASE_TITLE vs MB_CASE_TITLE_SIMPLE

PHP

```php
$str = "árvÍz tűRŐ tükör FÚRÓ gép";
echo mb_convert_case($str, MB_CASE_TITLE) . "\n";
echo mb_convert_case($str, MB_CASE_TITLE_SIMPLE) . "\n";
```

Eredmény

```
Árvíz Tűrő Tükör Fúró Gép
Árvíz Tűrő Tükör Fúró Gép
```

PHP

```php
$str = "schönbrunner straße";
echo mb_convert_case($str, MB_CASE_TITLE) . "\n";
echo mb_convert_case($str, MB_CASE_TITLE_SIMPLE) . "\n";
```

Eredmény

```
Schönbrunner Straße
Schönbrunner Straße
```

MB_CASE_FOLD vs MB_CASE_FOLD_SIMPLE

PHP

```php
$str = "GÖMANN";   // csupa nagybetű *
echo mb_convert_case($str, MB_CASE_FOLD) . "\n";
echo mb_convert_case($str, MB_CASE_FOLD_SIMPLE) . "\n";
```

Eredmény

```
gössmann
gößmann
```

PHP

```php
$str = "gößmann";   // csupa kisbetű
echo mb_convert_case($str, MB_CASE_FOLD) . "\n";
echo mb_convert_case($str, MB_CASE_FOLD_SIMPLE) . "\n";
```

Eredmény

```
gössmann
gößmann
```

* A nagybetűs ß karaktert átalakította SS-re a LaTeX

Tartalom

Keresés szövegben

```
mb_strpos( string $haystack, string $needle, int $offset =
↪  0, ?string $encoding = null): int|false
```

- A `$haystack`-ben keres
- A `$needle` a keresett szöveg
- Az `$offset` segítségével lehet megadni, hogy hányadik indextől kezdje a keresést. Opcionális, alapértlemezett értéke: 0.
- Az `$encoding` segítségével a karakterkódolást lehet megadni.
- A **visszatérési értéke** találat esetén az illeszkedés első indexét adja (`int`), míg ha nem találta meg, akkor az értéke `false` lesz

Linkek:

- mb_strpos() - PHP dokumentáció

A strpos() és az mb_strpos() összehasonlítása

PHP
```php
$szoveg = "Hódmezővásárhely";
```

Rossz példa!
```php
echo strpos($szoveg,"ár");
```

Eredmény
```
13
```

PHP
```php
echo mb_strpos($szoveg,"ár");
```

Eredmény
```
10
```

- A `strpos()` a multibyte karaktereket, mint a magyar ékezetes karakterek többnek számolja
- Az `mb_strpos()` helyesen, egy karakternek számítja
- Az ó, ő és az első á karakterek 1 helyett 2 bájtot foglalnak, így jön ki a különbség a két függvény eredménye között

Tartalom

str_contains() [PHP 8]

```
str_contains(string $haystack, string $needle): bool
```

- A `$haystack`-ben keres
- A `$needle` a keresett szöveg
- A **visszatérési értéke** (`bool`) meghatározza, hogy a "szénakazalban" megtalálható -e a "tű"
- Kis- és nagybetűkre érzékeny
- Nincs `mb_` prefix változata

Linkek:

- str_contains() - PHP dokumentáció
- str_contains() - PHP RFC

str_starts_with() [PHP 8]

```
str_starts_with(string $haystack, string $needle): bool
```

- A `$haystack`-ben keres
- A `$needle` a keresett szöveg
- A **visszatérési értéke** (`bool`) meghatározza, hogy a "szénakazal" **kezdetén** megtaláható -e a "tű"
- Kis- és nagybetűkre érzékeny
- Nincs `mb_` prefix változata

Linkek:

- str_starts_with() - PHP dokumentáció
- str_starts_with() - PHP RFC

str_ends_with() [PHP 8]

```
str_ends_with(string $haystack, string $needle): bool
```

- A `$haystack`-ben keres
- A `$needle` a keresett szöveg
- A **visszatérési értéke** (`bool`) meghatározza, hogy a "szénakazal" **végén** megtaláható -e a "tű"
- Kis- és nagybetűkre érzékeny
- Nincs `mb_` prefix változata

Linkek:

- str_ends_with() - PHP dokumentáció
- str_ends_with() - PHP RFC

Tartalom

str_pad()

```
str_pad(string $string, int $length, string $pad_string =
↪   " ", int $pad_type = STR_PAD_RIGHT): string
```

- A `$string` amit szeretnénk kiegészíteni
- A `$length` az a hossz, amit végeredményként szeretnénk megkapni
- A `$pad_string` a kitöltőkarakter lesz
- A `$pad_type` a kitöltés típusa. nem kötelező megadni, ilyenkor `STR_PAD_RIGHT` meghatározza melyik oldalon legyen a kitöltés
  - `STR_PAD_RIGHT` a szöveg után (alapértelmezett)
  - `STR_PAD_LEFT`, a szöveg elé
  - `STR_PAD_BOTH`, a szöveg elé és után is "megfelezve"
- A **visszatérési értéke** a szöveg a kitöltött karakterekkel együtt
- Nincs `mb_` prefix verziója

Linkek:

- mb_strpad() - PHP dokumentáció

mb_str_pad() saját függvény

PHP

```php
if (!function_exists('mb_str_pad'))
{
    function mb_str_pad($input, $pad_length, $pad_string=' ',
    ↪  $pad_type=STR_PAD_RIGHT)
    {
        $diff = strlen($input) - mb_strlen($input);
        return str_pad($input, $pad_length+$diff, $pad_string,
        ↪  $pad_type);
    }
}
```

- A `str_pad` multibyte verziója megírható
- A `function_exists` ellenőrzi, hogy nem -e létezik a függvény (jövő álló)
- Itt a `$input` lehet multibyte, a `$pad_string` továbbra sem

Tartalom

str_replace()

```
str_replace(array|string $search, array|string $replace,
↪  string|array $subject, int &$count = null):
↪  string|array
```

- A `$search` a keresett érték
- Találat esetén a `$replace` értékét helyezi el
- A `$subject` az eredeti szöveg
- A `$count` referencia szerint kerül átadásra. Bár nem kötelező, ha megadjuk akkor ebbe a változóba adja meg hány csere történt.
- A **visszatérési értéke** az eredeti szöveg, amiben a keresett értéket lecserélte a megfelelő értékre
- Kis- és nagybetűkre érzékeny, ellentétben a `str_ireplace()` függvénnyel
- Nincs `mb_` prefix változata

Linkek:

- str_replace() - PHP dokumentáció

str_replace() - egyszerű példa

PHP

```php
$szoveg = "baba";
$mit = "b";
$mire = "m";

$eredmeny = str_replace($mit,$mire,$szoveg);

echo $eredmeny;
```

Eredmény

```
mama
```

str_replace() - tömbökkel

PHP

```php
$szoveg = "árvíztűrőtükörfúrógép";
$ekezetes = [ "á", "é", "í", "ó", "ö", "ő", "ú", "ü", "ű" ];
$ekezetlen = [ "a", "e", "i", "o", "o", "o", "u", "u", "u" ];

echo str_replace($ekezetes,$ekezetlen,$szoveg);
```

Eredmény

```
arvizturotukofurogep
```

str_replace() - asszociatív tömb

PHP

```php
$t = [
    "á" => "a",
    "é" => "e",
    "í" => "i",
    "ó" => "o",
    "ö" => "o",
    "ő" => "o",
    "ú" => "u",
    "ü" => "u",
    "ű" => "",
];
$szoveg = "árvíztűrőtükörfúrógép";
echo str_replace(array_keys($t),array_values($t),$szoveg);
```

Eredmény

```
arvizturotukofurogep
```

Tartalom

Szövegrészlet kivágása

```
mb_substr(string $string, int $start, ?int $length = null,
↪  ?string $encoding = null): string
```

- A `$string` az eredeti szöveg
- A `$start` a kezdő pozíció
- A `$length` a kivágandó szöveg hossza (karakterek száma)
- A `$encoding` karakterkódolás (opcionális)
- A **visszatérési értéke** a kivágott szöveg a `$start` indextől kezdve `$length` darab karakter.
  - Amennyiben nem megfelelőek az értékek, úgy üres szöveget ad vissza
- Az eredeti szöveget nem módosítja

Linkek:

- mb_substr() - PHP dokumentáció

Tartalom

trim()

```
trim(string $string, string $characters = "
↪  \n\r\t\v\x00"): string
```

- A `$string` az eredeti szöveg
- A `$characters` eltávolítandó karakterek sorozata
- A **visszatérési értéke** a szöveg az elejéről és végéről eltávolított karakterekkel
- Az eredeti szöveget nem módosítja

Linkek:

- trim() - PHP dokumentáció

Tartalom

explode()

```
explode(string $separator, string $string, int $limit =
↪  PHP_INT_MAX): array
```

- A `$separator` az elválasztó karaktersorozat
- A `$string` az eredeti szöveg
- A `$limit` maximum ennyi elemre bontja szét
  - Pozitív érték esetén az utolsó darab a szöveg teljes végét tartalmazza
  - Negatív érték esetén az utolsó `n` elemet kihagyja a végeredményből
  - Nulla esetén az értéket 1-nek veszi
- A **visszatérési értéke** egy olyan tömb, ami a szövegrészekből áll össze
- Az eredeti szöveget nem módosítja

Linkek:

- explode() - PHP dokumentáció

explode() - példa

PHP

```php
$szoveg = "alma/banán/barack/dinnye";
$gyumolcsok = explode("/",$szoveg);
var_dump($gyumolcsok);
```

Eredmény

```
array(4) {
    [0] => string(4) "alma"
    [1] => string(6) "banán"
    [2] => string(6) "barack"
    [3] => string(6) "dinnye"
}
```

explode() - példa

PHP

```php
$szoveg = "alma/banán/barack/dinnye";
$gyumolcsok = explode("/",$szoveg,3);
var_dump($gyumolcsok);
```

Eredmény

```
array(4) {
    [0] => string(4) "alma"
    [1] => string(6) "banán"
    [2] => string(13) "barack/dinnye"
}
```

explode() - példa

PHP

```php
$szoveg = "alma/banán/barack/dinnye";
$gyumolcsok = explode("/",$szoveg,-1);
var_dump($gyumolcsok);
```

Eredmény

```
array(4) {
    [0] => string(4) "alma"
    [1] => string(6) "banán"
    [2] => string(6) "barack"
}
```

implode()

```
implode(string $separator, array $array): string
```

- A `$separator` az elválasztó karaktersorozat
- A `$array` az eredeti szöveg
- A **visszatérési értéke** egy string, amiben a tömb elemei szerepelnek egymás után a `$separator` karakterrel elválasztva
- A `join()` egy alias erre a függvényre

Figyelem!

A korábbi verziók esetében fordított sorrendben voltak a paraméterek! PHP 8-tól kezdve a korábbi legacy sorrend nem működik!

Linkek:

- implode() - PHP dokumentáció

implode() - példa

PHP

```php
$gyumolcsok = ["alma","banán","barack","dinnye"];
$szoveg = implode("/",$gyumolcsok);
echo $szoveg;
```

Eredmény

```
alma/banán/barack/dinnye
```