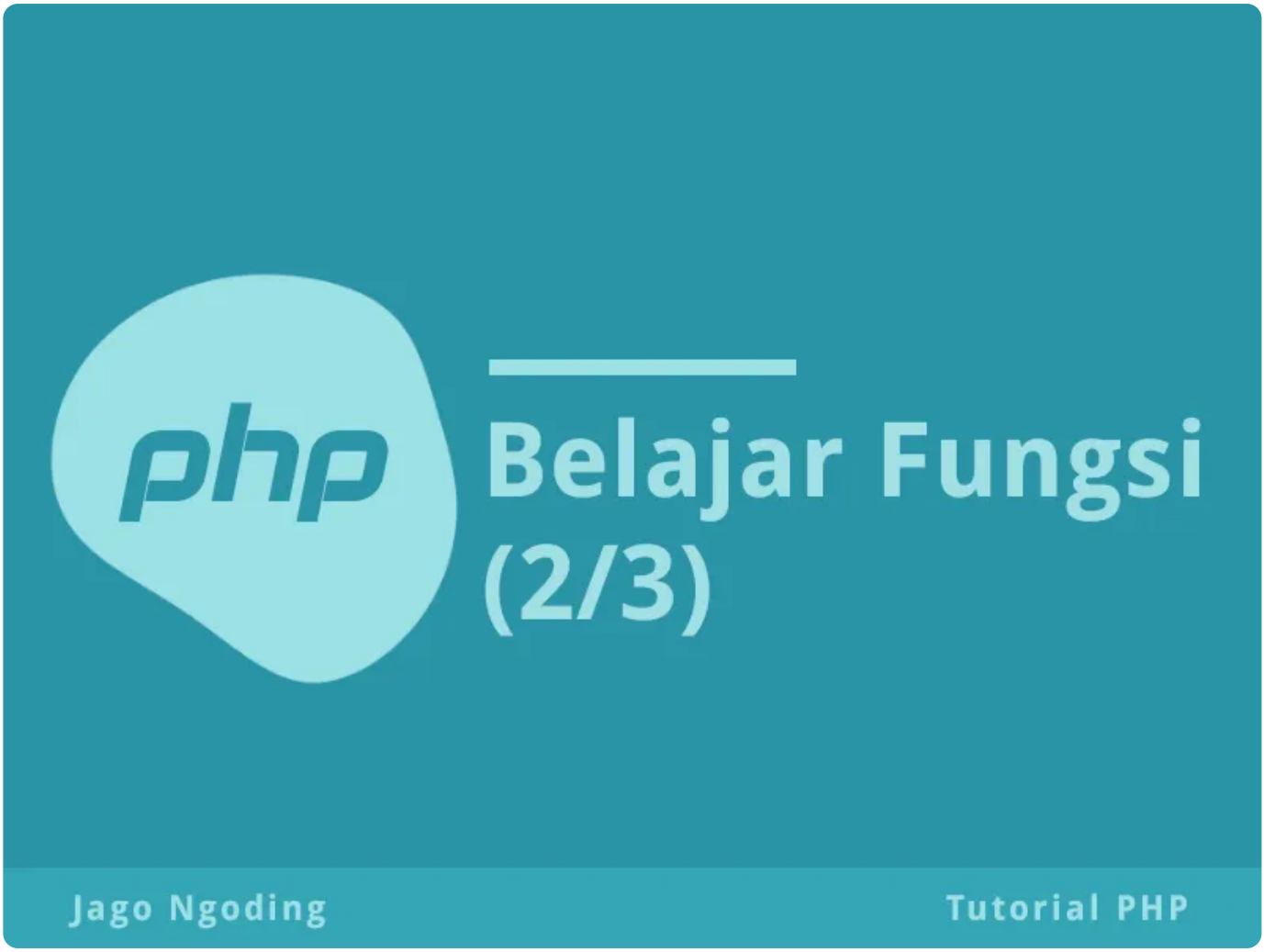
Nurul Huda
5 February 2020

PHP Dasar: Belajar Fungsi (2/3)



PHP PHP Dasar

Pada pertemuan sebelumnya, kita telah membahas bagian pertama dari pembahasan fungsi pada PHP. Kita telah membahas tentang fungsi mulai dari pengertiannya, penerapannya secara sederhana, dan kita juga telah membahas tentang hal-hal yang berkaitan tentang parameter fungsi pada PHP.

Pada tutorial kali ini, kita masih dalam pembahasan fungsi. Kita akan mempelajari tentang *return function*, *fungsi anonim*, *callback* dan juga *arrow function*.

Dan insyaallah pada tutorial selanjutnya akan ada pembahasan fungsi bagian ke-3 yang akan membahas tentang perulangan rekursif.

Fungsi Yang Mengembalikan Nilai

Secara umum fungsi terbagi menjadi dua:

1. Fungsi yang tidak mengembalikan nilai
2. Fungsi yang mengembalikan nilai

Untuk fungsi yang tidak mengembalikan nilai, sudah kita pelajari pada tutorial fungsi bagian 1.

pada variabel.

Banyak fungsi bawaan PHP yang mengembalikan nilai, contohnya adalah fungsi `isset`, `empty`, `is_null`, `count`, dan lain-lain.

Berikut ini adalah contoh beberapa fungsi bawaan PHP yang mengembalikan suatu nilai tertentu:

```php
<?php

$a = ['Merah', 'Kuning', 'Hijau'];

$apakahVariabelABernilaiNull = is_null($a); # hasilnya false
$apakahVariabelABertipeArray = is_array($a); # hasilnya true
$panjangA = count($a); # 3

$passwordTerenkripsi = md5('12345');

var_dump($apakahVariabelABernilaiNull);
echo "<br>";
var_dump($apakahVariabelABertipeArray);
echo "<br>";
var_dump($panjangA);
echo "<br>";
var_dump($passwordTerenkripsi);
echo "<br>";
```

Jika kita perhatikan contoh di atas, kita akan dapati bahwa:

- Fungsi `is_null()` akan mengembalikan dua nilai: `true` atau `false`. Nilai `true` jika parameter yang dilempar adalah `null`, dan bernilai `false` jika parameter yang dilempar tidak bernilai `null`.
- Fungsi `is_array()` sama dengan fungsi `is_null()`, hanya saja ia bertugas untuk memeriksa apakah parameter yang dilempar adalah `array` atau bukan.

- Fungsi `md5` mengembalikan sebuah string hasil enkripsi dari parameter yang dilempar.

Membuat fungsi yang mengembalikan nilai

Misalkan kita akan membuat suatu fungsi untuk menghitung volume balok dengan rumus: `panjang * lebar * tinggi`.

Fungsi tersebut akan menerima 3 buah parameter dengan tipe data `float`.

Dan ketika fungsi tersebut dipanggil, ia akan mengembalikan hasil perhitungan rumus volume balok berdasarkan tiga parameter yang kita lemparkan.

```
<?php

function hitungVolumeBalok (float $panjang, float $lebar, float $tinggi)
  $luasBalok = $panjang * $lebar * $tinggi;

  return $luasBalok;
}
```

Atau bisa kita singkat dengan langsung me-*return* hasil kali dari kesemua parameter:

```
<?php

function hitungVolumeBalok (float $panjang, float $lebar, float $tinggi)
  return $panjang * $lebar * $tinggi;
}
```

Saat memanggil fungsi tersebut, kita bisa menyimpan nilai yang dikembalikan ke dalam sebuah variabel, atau kita juga bisa lakukan untuk hal yang lainnya semisal menjadikan hasil kembaliannya sebagai parameter untuk fungsi yang lain.

```php
# hasil kembalian function langsung di-echo
echo hitungVolumeBalok(5, 20, 10) . "<br>";
# hasil kembalian function disimpan dalam variabel
$volume = hitungVolumeBalok(10, 2, 7);
# hasil kembalian function dijadikan parameter untuk function yang lain
var_dump(hitungVolumeBalok(10, 10, 10));
```

Memperjelas Tipe Data Hasil Kembalian Fungsi

Kita bisa memperjelas tipe data kembalian fungsi dengan menambahkan sintaks `:` lalu diikuti setelahnya oleh tipe data yang kita inginkan.

Perhatikan fungsi `hitungVolumeBalok() : float` berikut:

```php
<?php

function hitungVolumeBalok (float $p, float $l, float $t) : float {
  return $p * $l * $t;
}
```

Jika anda berusaha untuk mereturn selain `float`, maka PHP akan menyatakan statemen tersebut sebagai error.

Fungsi Anonim atau Closure

Dari yang telah kita pelajari, kita tahu bahwasanya fungsi pada PHP harus memiliki nama. Peraturan penamaannya kurang lebih sama dengan penamaan variabel, hanya saja kita tidak bisa memberi nama fungsi kita dengan nama yang telah digunakan oleh PHP sebagai fungsi bawaan.

Untuk mendefinisikan fungsi, kita bisa dengan rumus seperti ini:

```
function namaFungsi ($param1, $param2, $param3 = null) {
  # kode program
}
```

Akan tetapi di dalam PHP, terdapat *anonymous function* alias fungsi tanpa nama. Kita bisa mendefinisikan suatu fungsi tanpa nama dan menaruhnya di dalam variabel, atau bahkan kita bisa menjadikannya sebagai parameter pada fungsi yang lain.

Perhatikan contoh berikut:

```
<?php

$namaLengkap = function ($namaDepan, $namaBelakang) {
  return "{$namaDepan} {$namaBelakang}";
};

echo $namaLengkap("Nurul", "Huda");
```

Perhatikan kode program di atas. Kita dapat simpulkan beberapa hal:

- Kita mendefinisikan satu fungsi tanpa nama
- Fungsi tersebut menerima dua parameter `$namaDepan` dan `$namaBelakang`
- Fungsi tersebut kita *assign* ke variabel `$namaLengkap`
- Kita **wajib** memberi titik koma (`;`) pada saat mendefinisikan fungsi anonim dan menyimpannya dalam variable
- Kita memanggil fungsi tersebut dengan cara memanggil nama variabel yang telah kita assign dengan diikuti tanda `()` .

Fungsi Anonim Sebagai Parameter Fungsi Lain

Logikanya, sesuatu yang bisa disimpan pada variabel, ia juga bisa kita gunakan sebagai parameter pada suatu fungsi.

Sekarang kita akan mencoba untuk membuat suatu fungsi, yang mana fungsi tersebut menerima fungsi anonim sebagai salah satu parameternya.

Silakan praktikkan contoh kode program berikut dengan perlahan. Saya sudah memberikan komentar pada setiap baris kode yang penting agar lebih mudah dipahami.

```php
<?php

/**
 * Fungsi ini untuk melakukan foreach pada setiap item pada array.
 * Lalu parameter ke-2 adalah fungsi anonim untuk menangani item array
 * tersebut mau diapakan
 */
function tampilkanArray (array $array, callable $fungsiEchoer) {
  foreach ($array as $key => $item) {
    # di sini kita tidak tahu fungsi anonim ini tugasnya seperti apa
    # karena ia tergantung fungsi yang dilemparkan sebagai parameter
    $fungsiEchoer($key, $item, count($array));
  }
}

# kita bikin satu variabel array berisi kumpulan nama mahasiswa
$listMahasiswa = ['Nurul Huda', 'Wahid Abdullah', 'Lendis Fabri'];

# kita panggil fungsi tampilkanArray()
# kita passing variabel $listMahasiswa untuk parameter 1, dan
# fungsi anonim yang akan menangani setiap item dari array
# sebagai parameter 2
tampilkanArray($listMahasiswa, function ($key, $nama) {
  echo "{$key} - Bung {$nama} <br>";
});
```

```
    echo "{$nama}";

    if ($key < $panjangArray - 1) {
      echo " - ";
    }
  });

  echo "<br>";
```

Fungsi Anonim Sebagai Callback

Inti *callback* adalah sebuah fungsi anonim yang dijadikan parameter pada fungsi yang lain (seperti yang sudah kita praktikkan di atas). Akan tetapi pada penggunaannya, umumnya *callback* adalah sesuatu yang dilakukan setelah suatu tugas tertentu telah selesai.

Misalkan kita memiliki fungsi untuk menyimpan data ke dalam database bernama `saveToDatabase()`.

```
<?php

function saveToDatabase ($data, callable $callback) {
  # save to database
  # ...

  # setelah selesai, jalankan callback
  $callback();
}

# panggil fungsi saveToDatabase
saveToDatabase($dataInvoice, function () {
  kirimNotifikasiViaEmail();
});
```

penggunaan calback.

Arrow Function

Arrow function adalah fitur baru yang ada pada PHP 7.4. Ia merupakan versi singkat dari sintaks fungsi anonim yang mereturn sebuah nilai tertentu.

Arrow function sendiri sudah populer di bahasa pemrograman lain semisal javascript dan juga di java (kalau tidak salah sejak versi 8).

Akan tetapi di PHP, arrow function hanya sederhana sekali, ia hanya support satu baris ekspresi saja. Sehingga untuk tugas yang kompleks, kita tetap harus menggunakan sintaks fungsi anonim konvensional.

Perhatikan fungsi anonim berikut:

```php
<?php

$faktor = 10;
$himpunanAsli = [1, 2, 3, 4];
$himpunanKelipatan10 = array_map(function ($n) use ($faktor) {
  return $n * $faktor;
}, $himpunanAsli);

var_dump($himpunanAsli);
echo "<br>";
var_dump($himpunanKelipatan10);
```

Kita bisa mengubahnya ke dalam versi arrow function yang hanya satu baris saja:

```php
<?php

$faktor = 10;
$himpunanAsli = [1, 2, 3, 4];
$himpunanKelipatan10 = array_map(fn($n) => $n * $faktor, $himpunanAsli);
```

```
var_dump($himpunanKelipatan10);
```

Arrow function juga menggunakan scope variabel parent-nya, sehingga kita tidak perlu menggunakan perintah `use($variable)` untuk menggunakan variable dari scope yang berada di luar fungsi.

Untuk detil perintah `array_map` silakan baca pembahasan tentang Bekerja dengan Array pada PHP.

Pembahasan Selanjutnya

Pembahasan kita tentang fungsi sudah semakin *advanced*. Sehingga sekarang kita sudah memiliki cukup banyak senjata untuk bisa menggunakan bahasa pemrograman PHP.

Pada pembahasan selanjutnya, kita masih akan membahas tentang fungsi bagian ke-3, yaitu tentang fungsi rekursif —Insyaallah—.

Kita telah menyinggung tentang fungsi rekursif pada pembahasan perulangan pada PHP.

Ia adalah metode perulangan dengan mengeksekusi suatu fungsi, yang mana fungsi tersebut akan memanggil dirinya sendiri.

Bagaimana caranya?

Sampai jumpa lagi!

Bagikan:

Nurul Huda

Web Developer. FOSS addict. Pengguna Arch Linux (dan Ubuntu). Penyuka kopi saset. Dan pernah kuliah Teknik Informatika sampai lulus.

Dukung Jago Ngoding ☕

← Sebelumnya
PHP Dasar: Belajar Fungsi (1/3)

Selanjutnya →
PHP Dasar: Fungsi Rekursif (3/3)

Artikel Terkait

PHP Dasar: Belajar Fungsi (1/3)

PHP Dasar: Perulangan

PHP Dasar: Logika Percabangan

PHP Dasar: Macam Macam Operator

PHP Dasar: Tipe Data Dan Variabel

0 Comments - *powered by utteranc.es*

Write | Preview

Sign in to comment

Styling with Markdown is supported

Sign in with GitHub