Syntax PHP - Lanjutan

PHP Loop

Loop sementara mengeksekusi blok kode selama kondisi yang ditentukan benar.

Sintaksis:

```
while (kondisi benar) {
    kode yang akan dieksekusi;
}
```

Contoh

```
<?php
$x = 0;

while($x <= 100) {
    echo "Angka Sekarang: $x <br>";
    $x+=10;
}
?>
```

Penjelasan:

$ x = 0; - Inisialisasi penghitung loop ($ x), dan tetapkan nilai awal ke 0

$ x <= 100 - Lanjutkan loop selama $x kurang dari atau sama dengan 100

$ x + = 10; - Tingkatkan nilai penghitung loop sebesar 10 untuk setiap iterasi

While loop pada indexed Array

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
<?php
    $hewan = ['buaya','anjing', 'babi', 'cicak', 'domba'];
    $i=0;
    while($i< count($hewan)){
       echo $hewan[$i] . "<br>";
        $i++;
```

```
    }
?>
</body>
</html>
```

PHP Do ... While Loop

Loop do ... while akan selalu menjalankan blok kode sekali, kemudian akan memeriksa kondisinya, dan mengulangi loop selama kondisi yang ditentukan benar.

Sintaksis

```
do {
    kode yang akan dieksekusi;
} while (kondisi benar);
```

Contoh 1:

Contoh di bawah ini pertama-tama menetapkan variabel $x ke 1 ($ x = 1). Kemudian, loop do while akan menulis beberapa output, dan kemudian menambah variabel $ x dengan 1. Kemudian kondisinya diperiksa (apakah $ x kurang dari, atau sama dengan 5?), Dan loop akan terus berjalan selama $ x kurang dari, atau sama dengan 5:

```
<? php
$x = 1;

do {
    echo "angka sekarang adalah: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

Catatan:

Dalam do ... while, kondisi diuji setelah menjalankan pernyataan dalam loop. Ini berarti loop do ... while akan menjalankan pernyataannya setidaknya satu kali, walaupun kondisinya salah.

Lihat contoh di bawah ini:

Contoh ini menetapkan variabel $x ke 6, kemudian menjalankan loop, **dan kemudian kondisinya diperiksa**:

```
<? php
$x = 6;

do {
    echo "Angka Sekarang adalah: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

Do-While loop pada indexed Array

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
<?php
    $hewan = ['buaya','anjing', 'babi', 'cicak', 'domba'];
    $i=0;

    do {
        echo $hewan[$i] ."<br>";
        $i++;
    }
    while($i < count($hewan));
?>

</body>
</html>
```

PHP Conditional Statements

Sangat sering ketika Anda menulis kode, Anda ingin melakukan tindakan berbeda untuk kondisi yang berbeda. Anda dapat menggunakan pernyataan kondisional dalam kode Anda untuk melakukan ini.

Dalam PHP kami memiliki pernyataan kondisional berikut:

- if statement - mengeksekusi beberapa kode jika satu syarat benar
- if ... else statement - mengeksekusi beberapa kode jika suatu kondisi benar dan kode lain jika kondisi itu salah
- if ... else if ... else statement - mengeksekusi kode yang berbeda untuk lebih dari dua syarat
- beralih pernyataan - memilih salah satu dari banyak blok kode yang akan dieksekusi

If Statement

Pernyataan if mengeksekusi beberapa kode jika satu syarat benar.

```
Sintaksis
if (kondisi) {
    kode yang akan dieksekusi jika kondisinya benar;
}
```

Contoh

Keluaran: "Semoga harimu menyenangkan!" jika waktu saat ini (JAM) kurang dari 20:

```
<? php
$t = date ("H");
if ($t <"20") {
    echo "Semoga harimu menyenangkan!";
}
?>
```

If-Else Statement

Pernyataan **if ... else** mengeksekusi beberapa kode jika suatu kondisi benar dan kode lain jika kondisi itu salah.

Keluaran: "Semoga harimu menyenangkan!" jika waktu saat ini kurang dari 20, dan "Selamat malam!" jika tidak:

```
<? php
$ t = date ("H");

if ($ t <"20") {
    echo "Semoga harimu menyenangkan!";
} else {
    echo "Selamat malam!";
}
?>
```

PHP - if...else if...else Statement

Pernyataan if ... elseif ... else mengeksekusi kode yang berbeda untuk lebih dari dua syarat.

Sintaksis

```
if (kondisi) {
    kode yang akan dieksekusi jika kondisi ini benar;
} else if (kondisi) {
    kode yang akan dieksekusi jika kondisi pertama salah dan
kondisi ini benar;
} else {
    kode yang akan dieksekusi jika semua kondisinya salah;
```

```
}
```

Contoh

Keluaran: "Selamat pagi!" jika waktu saat ini kurang dari 10, dan "Semoga harimu menyenangkan!" jika waktu saat ini kurang dari 20. Jika tidak maka akan menampilkan "Selamat malam!":

```
<? php
$t = date ("H");

if ($t < "10") {
    echo "Selamat pagi!";
} else if ($t < "20") {
    echo "Semoga harimu menyenangkan!";
} else {
    echo "Selamat malam!";
}
?>
```

Contoh 2:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
    <?php
    $uang_jansutris = 1000;
    $laptop_gaming = 2000;
    $uang_koperasi = 3000;

    if($uang_jansutris > $laptop_gaming)
    {
        echo 'sanggup dibeli oleh jansutris sendiri';
    }

    else if($uang_koperasi > $laptop_gaming)
    {
        echo 'sanggup dibeli dengan pinjam uang dari
koperasi';
    }

    else{
        echo 'uang gak cukup, nabung dulu sana';
    }
    ?>
```

```
</body>
</html>
```

If-Bercabang (nested-if)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
    <?php
    $uang_programmer = 1000;
    $laptop_gaming = 2000;
    $uang_koperasi = 4000;

    if($uang_programmer > $laptop_gaming)
    {
        echo 'sanggup dibeli oleh programmer';
    }

    else if($uang_koperasi > $laptop_gaming)
    {
        echo 'sanggup dibeli oleh koperasi' . "<br>";

        if($uang_koperasi >= $laptop_gaming * 2)
        echo 'laptop yang sanggup terbeli ada dua' . "<br>";
    }

    else{
        echo 'uang gak cukup, nabung dulu sana' . "<br>";
    }
    ?>
</body>
</html>
```

PHP Switch Statement

Pernyataan **Switch** digunakan untuk melakukan tindakan yang berbeda berdasarkan kondisi yang berbeda.

Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
    <?php
  $nama = 'Moses';

  switch($nama)
  {
    case 'Rivaldo':
    echo 'kondisi terpenuhi,namaku Rivaldo';
    break;

    case 'Sotar dodo':
    echo 'kondisi terpenuhi,namaku Sotar dodo';
    break;

    case 'Moses':
        echo 'kondisi terpenuhi, namaku Moses ';
        break;

    case 'Arta Hutapea':
    echo 'kondisi terpenuhi,namaku Arta Hutapea';
    break;

    default:
    echo 'tidak ada pilihan yang benar';
  }
?>
</body>
</html>
```

Operator Logika

Operator Logika terdiri dari:

- ==
  operator pembanding kedua nilai yang sama tanpa memperhatikan kesamaan tipe-data)
- ===
  operator pembanding kedua nilai yang sama dengan memperhatikan kesamaan tipe-data)
- >
- <
- <=
- >=
- !=

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
    <?php
    $loginBerhasil = true;
    $loginGagal = false;

    $password  = '1234';
    $password2 = '1234';

        if($password == $password2){
        echo 'login berhasil' . "<br>";
        echo 'nilai login: ' . $loginBerhasil;
        }

        else{
            echo 'login gagal' . "<br>";
            echo 'nilai login: ' . $loginGagal;
        }
    ?>
</body>
</html>
```

AND dan OR Operator

Berfungsi menggabungkan 2 kondisi dalam single statement (pada 1 baris program). Perbedaannya adalah:

AND: Semua kondisi harus bernilai benar, maka hasil bernilai benar

OR: Cukup salah satu kondisi harus bernilai benar, maka hasil bernilai benar

Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
```

```
    <?php
    $uang_programmer = 1000;
    $laptop_gaming = 2000;
    $uang_koperasi = 4000;

    /* Operator && dan ||
    && semua kondisi harus bernilai benar => benar,
    || salah satu kondisi bernilai benar => benar
    */

    if($uang_programmer > $laptop_gaming || $uang_koperasi >
$laptop_gaming)
    {
        echo 'sanggup dibeli oleh programmer';
    }

    else{
        echo 'uang gak cukup, nabung dulu sana' . "<br>";
    }
    ?>
</body>
</html>
```

PHP Functions

Kekuatan nyata PHP berasal dari fungsinya.

PHP memiliki lebih dari 1000 fungsi bawaan, dan selain itu Anda dapat membuat fungsi kustom Anda sendiri.

Fungsi Bawaan (built-in function) PHP

PHP memiliki lebih dari 1000 fungsi bawaan yang dapat dipanggil langsung, dari dalam skrip, untuk melakukan tugas tertentu.

PHP User Defined Functions / Fungsi Buatan Pengguna PHP

Selain fungsi PHP bawaan, dimungkinkan untuk membuat fungsi Anda sendiri.

- Fungsi adalah blok pernyataan yang dapat digunakan berulang kali dalam suatu program.
- Suatu fungsi tidak akan dieksekusi secara otomatis ketika sebuah halaman dimuat.
- Suatu fungsi akan dieksekusi oleh jika fungsi tersebut dipanggil.

Buat Fungsi yang Ditentukan Pengguna dalam PHP

Deklarasi fungsi yang ditentukan pengguna dimulai dengan fungsi kata:

Sintaksis

```
function functionName () {
    kode yang akan dieksekusi;
}
```

Catatan:

Nama fungsi harus dimulai dengan huruf atau garis bawah. Nama fungsi TIDAK peka huruf besar kecil (NOT case-sensitive).

Tips: Beri nama fungsi yang mencerminkan fungsi itu!

Pada contoh di bawah ini, kita membuat fungsi bernama "**writeMsg ()**". Kurung kurawal buka **({)** menunjukkan awal kode fungsi, dan kurung kurawal penutup **(})** menunjukkan akhir dari fungsi. Fungsi ini menampilkan **"Halo dunia!".** Untuk memanggil fungsi, cukup tulis namanya diikuti dengan **tanda kurung ()**:

Contoh 1:

```
<? php
function writeMsg () {
    echo "Halo dunia!";
}

writeMsg (); // panggil fungsi
?>
```

Contoh2:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
    <?php
        function panggilan_masuk()
        {
            echo 'Hallo bisa bicara dengan jansutris?';
        }

        function jawaban()
        {
```

```
            echo 'Bisa. ini dengan jansutris sendiri';
        }

        function baris_Baru()
        {
            echo "<br>";
        }

    panggilan_masuk();
    baris_Baru();
    jawaban();

    baris_Baru();
    panggilan_masuk();
    baris_Baru();
    jawaban();

?>
</body>
</html>
```

Fungsi dengan Argument

- Informasi dapat diteruskan ke fungsi melalui argumen. Argumen seperti variabel.

- Argumen ditentukan setelah nama fungsi, di dalam tanda kurung. Anda dapat menambahkan argumen sebanyak yang Anda inginkan, cukup pisahkan dengan koma.

Contoh berikut memiliki fungsi dengan satu argumen ($fname). Saat fungsi **familyName()** dipanggil, kita juga ***passing*** nama (**mis. Jani),** dan nama tersebut digunakan di dalam fungsi tersebut, yang menampilkan beberapa nama depan yang berbeda, tetapi nama belakang yang sama:

```
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

Contoh berikut memiliki fungsi dengan dua argumen (**$fname** dan **$year**):

```php
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege", 1975);
familyName("Stale", 1978);
familyName("Kai Jim", 1983);
?>
```

Contoh 2:

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
    <?php
        function belanjaan($bahan, $jumlah)
        {
 $text = "daftar belanjaan: " .$bahan . " sebanyak ". $jumlah
. "KG";
            echo $text . '<br>';
        }

        function jarak()
        {
            echo '<br>';
        }

        belanjaan('bayam', 5);
        jarak();

        belanjaan('bawang merah', 2);
        jarak();

        belanjaan('daging babi', 10);
        jarak();

?>
</body>
</html>
```

Nilai Argumen Default PHP

Contoh berikut menunjukkan cara menggunakan parameter default. Jika kita memanggil fungsi **setHeight()** tanpa argumen, dibutuhkan nilai default sebagai argumen:

Contoh 3:

```
<?php declare(strict_types=1); // strict requirement ?>
<!DOCTYPE html>
<html>
<body>

<?php
function setHeight(int $minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight();
setHeight(135);
setHeight(80);
?>

</body>
</html>
```

Fungsi PHP - Mengembalikan nilai (Return Values)

Untuk membiarkan suatu fungsi mengembalikan nilai, gunakan pernyataan **return**:

Contoh 1:

```
<?php declare(strict_types=1); // strict requirement
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

Deklarasi Jenis Pengembalian PHP

PHP 7 juga mendukung Deklarasi Tipe untuk pernyataan pengembalian. Seperti dengan deklarasi tipe untuk argumen fungsi, dengan mengaktifkan **strict type**, itu akan melempar "Kesalahan Fatal" pada tipe data yang tidak sama.

Untuk mendeklarasikan tipe untuk fungsi kembali, tambahkan titik dua (:) dan tipe tepat sebelum braket keriting pembuka **({)** saat mendeklarasikan fungsi.

Dalam contoh berikut ini kami menentukan jenis kembali untuk fungsi:

Contoh :2:

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : float {
    return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

Anda dapat menentukan jenis pengembalian yang berbeda, dari jenis argumen, tetapi pastikan kembali adalah jenis yang benar:

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : int {
    return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
?>
```

Contoh 3:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
    <?php

        function penjumlahan($angka1, $angka2)
        {
            $sum = $angka1 + $angka2;
            return $sum;
        }

        function baris_Baru()
        {
            echo '<br>';
        }

        echo "Hasil Penjumlahan: " . penjumlahan(5, 3);
        baris_Baru();

        $perkalian = penjumlahan(5, 3) * 4;
        echo "Hasil Perkalian: " . $perkalian;
```

```
?>
</body>
</html>
```

Scope

Anda dapat mengakses nilai dari variable, walaupun variable tersebut tidak berada didalam fungsi. Anda dapat menggunakan keywords **global** atau **$GLOBALS ['nama_variabel']**

Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
    <?php
    $a =2;
    $b =3;

    function menghitung()
    {
      //  cara pertama menggunakan method global
      //silahkan un-comment baris global $a, $b dibawah

     // global $a, $b;

        $c = $GLOBALS['a']+ $GLOBALS['b'];
        return $c;
    }

    echo menghitung();
    ?>
</body>
</html>
```

Anonymous function

Anonymous function adalah fungsi yang tidak memilki nama namun ditampung dalam sebuah variabel.

Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
<?php
    $pesan = function(){
    echo 'selamat datang ';
    };

    $pesan();
 ?>
</body>
</html>
```

Anonymous function dengan parameter

Contoh 2:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
<?php
    $pesan = function($pengguna){
    echo $pengguna;
    };

    echo 'User Aktif: ' . '<br>';
    echo $pesan('Jansutris');
 ?>
</body>
</html>
```

Anonymous function dengan parameter menggunakan metode return

Contoh 3:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
```

```
    <title>PHP itu Mudah</title>
</head>
<body>
<?php
    $pesan = function($pengguna){
    $user = $pengguna;
    return $user;
    };

    echo 'Selamat datang ' . $pesan('jansutris');
 ?>
</body>
</html>
```

CallBack Function dengan method: **$callback()**

Memanggil fungsi setelah fungsi sebelumnya dijalankan.

Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
<?php
    function berteriak($callback)
    {
        echo "Hallooooo... <br>";
         $callback();
    }

    $panggil = function()
    {
        echo 'aku gak punya nama';
    };

    berteriak ($panggil);
 ?>
</body>
</html>
```

CallBack Function dengan method: **$ call_user_func()**

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
<?php
    function berteriak($callback)
    {
        echo "Hallooooo... <br>";
        // $callback();
        call_user_func($callback, 'Selamat datang');
    }

    $panggil = function($text)
    {
        echo $text;
    };

    berteriak ($panggil);
 ?>

</body>
</html>
```

Menguji fungsi Callback() dengan method is_Callable()

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>PHP itu Mudah</title>
</head>
<body>
<?php
    function berteriak($callback)
    {
        echo "Hallooooo... <br>";
        // $callback();
    if(is_callable($callback)) //menguji apakah dia fungsi
    {
        call_user_func($callback, 'dia berhasil memanggil
fungsi');
    }
        else{
            echo 'dia bukan memanggil fungsi';
        }
```

```
    }

    $panggil = function($text)
    {
        echo $text;
    };

    berteriak ($panggil);
 ?>
</body>
</html>
```

Debug program dengan: die dan var_dump()

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <?php
    $car = "Toyota";
   //var_dump menampilkan jumlah array,index, tipe data
   //dari sebuah variabel

    var_dump($car);
    echo '<br>';

    /* die() adalah syntax untuk debug program
untuk mengetahui apakah program anda bekeja hingga baris
terentu dengan cara menghentikan ekskusi program pada baris
deklarasi 'die()' sambil menampilkan nilai dari variabel
*/
    die($car);
    echo 'satu dua tiga';
    ?>
</body>
</html>
```

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

Martin Fowler