

## **LAPORAN PROYEK 3**

# **DEBUGING, PENGGUNAAN OPERATOR, DAN STRUKTUR KONTROL PERCABANGAN PHP**



**OLEH:**  
**AHMAD ARJUN TRISULA**  
(NISN. 0082311714)

**REKAYASA PERANGKAT LUNAK**  
**SMK NEGERI 1 KARANG BARU**  
**PEMERINTAH PROVINSI ACEH**  
**2024**



## MATERI PERTEMUAN 7

# DEBUGGING DALAM PHP



### TAHUKAH KAMU...?

Debugging dalam pemrograman PHP yang harus kita ketahui:

- Penggunaan Echo;
- Penggunaan Print;
- Penggunaan Var\_dump;
- Penggunaan Exit;
- Penggunaan Die;

## DEBUGGING PHP

Pelajari cara debugging kode PHP dengan langkah-langkah mudah dan alat-alat yang efektif untuk menemukan serta memperbaiki bug dalam script PHP kamu..

Menguasai teknik debugging adalah keterampilan penting bagi pengembang PHP untuk mengidentifikasi dan memperbaiki kesalahan dalam kode. Artikel ini akan memandu kamu melalui proses debugging PHP dengan langkah-langkah yang praktis.

Debugging adalah Proses Penting Ketika mengembangkan aplikasi PHP, kamu akan sering menemukan bugs atau kesalahan. Debugging adalah proses menemukan dan memperbaiki bugs tersebut. Untuk itu, penting bagi kamu untuk mengetahui beberapa teknik dasar dan alat yang bisa membantu dalam proses debugging.

### Menggunakan Error Reporting

PHP menyediakan cara sederhana untuk menampilkan kesalahan dengan menggunakan fungsi `error_reporting`. Untuk menampilkan semua kesalahan, peringatan, dan pemberitahuan, kamu bisa menambahkan kode berikut di awal skrip:

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

Pastikan hanya menggunakan ini selama tahap pengembangan karena menampilkan kesalahan secara langsung pada produksi adalah praktik yang buruk.

### Membaca Log Kesalahan

PHP juga menyimpan log semua kesalahan, yang dapat kamu akses untuk mengetahui apa yang salah dengan aplikasi. Lokasi file log kesalahan dapat berbeda tergantung pada konfigurasi server, namun biasanya kamu bisa menemukannya di `php.ini` dengan mencari `error_log`.

### Memahami Pesan Kesalahan

Pesan yang disajikan dalam log umumnya mencakup informasi yang kamu butuhkan untuk memperbaiki masalah, seperti jenis kesalahan, lokasi file, dan nomor baris tempat kesalahan terjadi.

### Menggunakan Xdebug

Xdebug adalah ekstensi PHP yang menyediakan fitur debugging dan profiling yang kaya. Kamu dapat menginstal Xdebug dan mengkonfigurasinya untuk bekerja dengan IDE atau editor kode

pilihan kamu. Xdebug memberikan fasilitas seperti breakpoint, stack traces, dan viewing variabel yang sangat memudahkan proses debugging.

Untuk mengaktifkan Xdebug, kamu perlu melakukan beberapa pengaturan pada php.ini, seperti berikut:

```
zend_extension="/usr/local/lib/php/extensions/no-debug-non-zts-20121212/xdebug.so"
xdebug.remote_enable=1
xdebug.remote_handler=dbgp
xdebug.remote_host=localhost
xdebug.remote_port=9000
```

Pastikan untuk mengganti path zend\_extension dengan lokasi sesungguhnya dari ekstensi Xdebug di server kamu.

### Melakukan Debugging Manual dengan var\_dump() dan print\_r()

Kadang-kadang, kamu hanya perlu mengetahui nilai dari variabel tertentu pada titik tertentu dalam kode. Fungsi var\_dump() dan print\_r() bisa sangat membantu:

```
var_dump($variabel);
print_r($variabel);
```

Ini harus digunakan sebagai solusi sementara karena dapat membuat output menjadi kacau jika kamu mencetak banyak informasi.

### Mencoba Mengisolasi Masalah

Jika aplikasi PHP kamu terdiri dari banyak komponen, mencoba mengisolasi bagian kode yang bermasalah dapat sangat membantu. Kamu bisa melakukan ini dengan mengomentari bagian kode tertentu atau memisahkan logika ke dalam fungsi terpisah dan mengetesnya satu per satu.

Memiliki teknik debugging yang baik akan membantu kamu menjadi pengembang yang lebih efisien. Ingatlah selalu bahwa kesalahan adalah bagian dari proses pengembangan, dan dengan alat yang tepat, kamu dapat menemukan dan memperbaiki masalah tersebut dengan mudah. Selamat debugging!

## ECHO

Pelajari cara menggunakan fungsi echo pada PHP untuk menampilkan teks dengan langkah-langkah mudah yang cocok untuk pemula.

Dalam pemrograman PHP, salah satu kegiatan paling dasar yang akan kamu lakukan adalah menampilkan teks ke layar. Cara paling sederhana untuk melakukan ini adalah dengan menggunakan fungsi echo. Artikel ini akan memandu kamu melalui dasar-dasar penggunaan echo untuk membuat script PHP kamu dapat berkomunikasi dengan pengguna.

Echo pada PHP Echo adalah salah satu konstruk bahasa PHP yang digunakan untuk mengeluarkan satu atau lebih string. Ini bukan fungsi sebenarnya, sehingga penggunaannya tidak memerlukan tanda kurung. Dengan echo, kamu dapat menampilkan teks sederhana, variabel, maupun hasil dari ekspresi string.

### Menggunakan Echo untuk Teks Sederhana

Untuk menampilkan teks sederhana, tulis kata echo diikuti oleh teks yang ingin kamu tampilkan. Teks harus dikelilingi oleh tanda kutip, bisa menggunakan tanda kutip tunggal ( ' ') atau tanda kutip ganda ( " ").

Contoh:

```
echo "Halo Dunia!";
```

### Tanda Kutip Tunggal vs Tanda Kutip Ganda

Tanda kutip tunggal akan menampilkan teks apa adanya, sedangkan tanda kutip ganda akan memproses variabel di dalamnya dan menampilkan nilainya.

### Contoh Dengan Variabel:

```
$nama = 'Budi';  
echo "Halo, $nama!"; // Hasil: Halo, Budi!  
echo 'Halo, $nama!'; // Hasil: Halo, $nama!
```

### Menampilkan Variabel

echo dapat digunakan untuk menampilkan nilai dari variabel. Ini sangat berguna dalam skrip PHP dimana kamu membutuhkan umpan balik dinamis berdasarkan input pengguna atau data lainnya.

### Contoh:

```
$umur = 25;  
echo "Umur saya adalah $umur tahun.";
```

### Menggabungkan Teks dan Variabel

Kamu dapat menggabungkan teks dan variabel serta operator lainnya untuk membuat keluaran yang lebih kompleks.

### Contoh:

```
$depan = 'Ali';  
$belakang = 'Rahman';  
echo "Nama saya adalah " . $depan . " " . $belakang . " .";
```

### Menggunakan Echo untuk HTML

Echo juga sering digunakan untuk menulis HTML melalui PHP. Ini memungkinkan pembuatan halaman web dinamis.

### Contoh:

```
echo "<h1>Judul Halaman</h1>";  
echo "<p>Paragraf ini dibuat dengan PHP.</p>";
```

### Kesalahan Umum

Berikut adalah beberapa kesalahan umum yang terjadi ketika pemula menggunakan echo:

Melupakan tanda kutip di sekitar string.

Tidak menggunakan titik (.) untuk menggabungkan string dan variabel.

Menggunakan tanda kutip tunggal ketika ingin memproses variabel di dalam string.

### Ringkasan

Fungsi echo merupakan alat yang sangat penting dan sering digunakan dalam PHP. Dengan mengikuti contoh-contoh di atas, kamu sekarang harus bisa menampilkan teks, variabel, dan HTML dengan mudah dalam script PHP kamu. Ingatlah untuk mempraktekkan setiap contoh untuk memahami dengan lebih baik bagaimana echo bekerja dan menjadi nyaman dengan nuansa penggunaan tanda kutip dan penggabungan string.

### PRINT\_R

Pelajari cara menggunakan fungsi print\_r di PHP untuk mencetak informasi tentang variable, termasuk array dan objek, dengan contoh sederhana yang mudah dipahami.

Fungsi `print_r()` merupakan salah satu fungsi di PHP yang sering digunakan oleh pengembang untuk menampilkan informasi tentang variabel. Fungsi ini memberikan output yang mudah dibaca, terutama bila kamu ingin mengecek struktur data seperti array atau objek.

### Menggunakan Fungsi `print_r`

Untuk menggunakan fungsi `print_r()`, kamu hanya perlu memanggil fungsi tersebut dan memasukkan variabel yang ingin dicetak sebagai argumennya.

#### Sintaks

```
print_r($variable);
```

#### Contoh Penggunaan dengan Variabel Biasa

```
$nama = "Budi";  
print_r($nama);
```

#### Hasil:

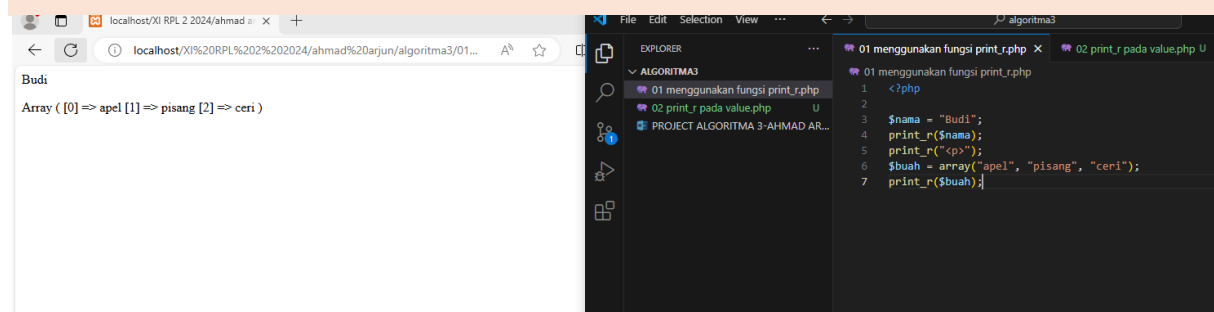
Budi

#### Contoh Penggunaan dengan Array

```
$buah = array("apel", "pisang", "ceri");  
print_r($buah);
```

#### Hasil:

```
Array  
(  
    [0] => apel  
    [1] => pisang  
    [2] => ceri  
)
```

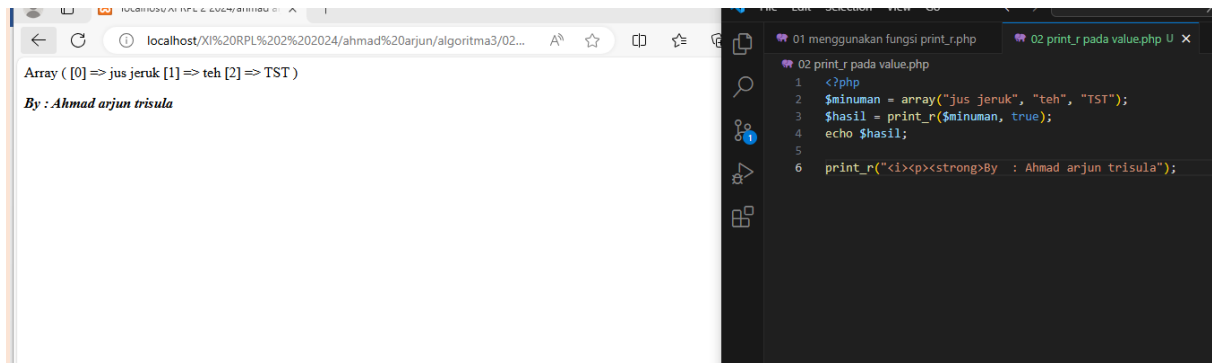


### Menampilkan Return Value dari `print_r`

Kamu bisa membuat `print_r()` mengembalikan informasi sebagai nilai dengan menetapkan parameter kedua menjadi `true`. Ini sangat berguna ketika kamu ingin menyimpan output `print_r()` ke dalam variabel.

#### Contoh Mengembalikan Nilai

```
<?php  
$minuman = array("jus jeruk", "teh", "TST");  
$hasil = print_r($minuman, true);  
echo $hasil;  
  
print_r("<i><p><strong>By : Ahmad arjun trisula");  
echo $hasil;
```



Output akan sama seperti contoh sebelumnya, namun sekarang disimpan dalam variabel \$hasil.

## Mencetak Objek

print\_r() juga bisa digunakan untuk mencetak informasi tentang objek. Kamu akan melihat properti dan nilai dalam bentuk yang terstruktur.

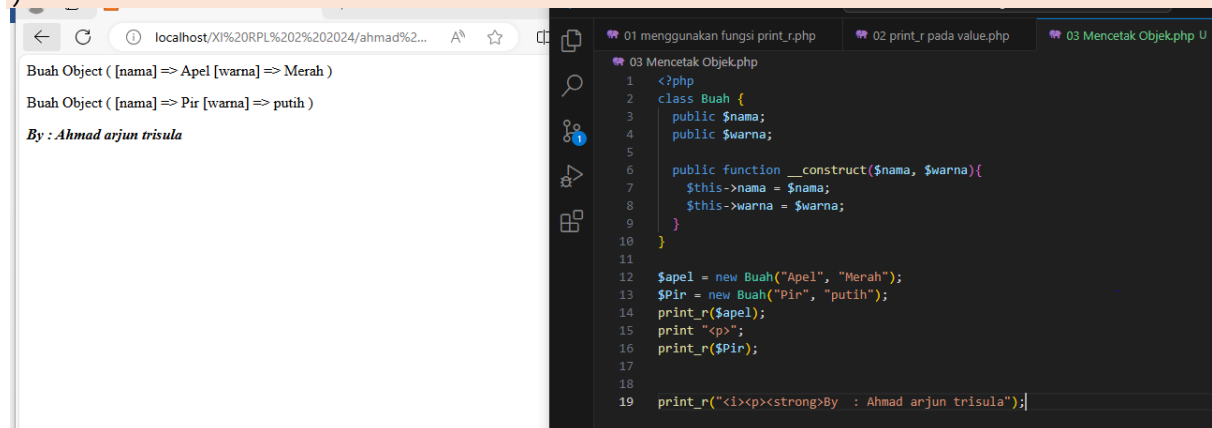
### Contoh dengan Objek

```
class Buah {  
    public $nama;  
    public $warna;  
  
    public function __construct($nama, $warna){  
        $this->nama = $nama;  
        $this->warna = $warna;  
    }  
}
```

```
$apel = new Buah("pir", "white");  
print_r($apel);
```

Hasil:

```
Buah Object  
(  
    [nama] => pir  
    [warna] => white  
)
```



## Kesalahan Umum

Seringkali pengguna baru bingung antara print\_r() dan echo. Ingatlah bahwa echo digunakan untuk mencetak string sederhana, sedangkan print\_r() lebih cocok untuk struktur data yang kompleks seperti array dan objek.

Dengan memanfaatkan print\_r(), kamu dapat dengan mudah memeriksa dan men-debug struktur data pada aplikasi PHP yang kamu kembangkan.

## Mengenal Fungsi var\_dump di PHP

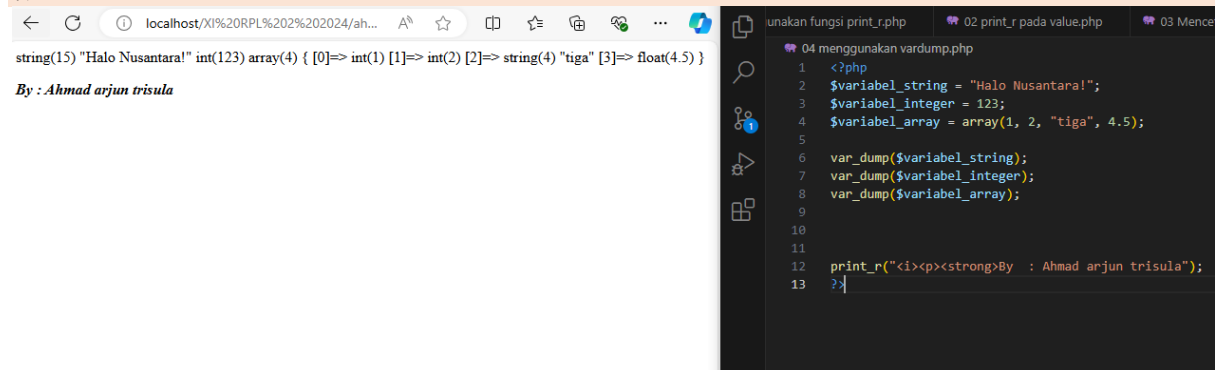
Fungsi `var_dump()` merupakan alat yang sangat berguna dalam pengembangan PHP. Fungsi ini digunakan untuk menampilkan informasi terperinci tentang satu atau lebih variabel. Informasi yang ditampilkan meliputi tipe data dan nilai dari variabel tersebut. Dalam praktiknya, `var_dump()` sering digunakan untuk men-debug dan memastikan bahwa data tersimpan dengan benar di dalam variabel.

### Menggunakan `var_dump()`

Untuk menggunakan `var_dump()`, kamu cukup menuliskannya diikuti dengan variabel yang ingin kamu periksa. Berikut adalah contoh penggunaannya:

```
<?php
$variabel_string = "Halo Dunia!";
$variabel_integer = 123;
$variabel_array = array(1, 2, "tiga", 4.5);

var_dump($variabel_string);
var_dump($variabel_integer);
var_dump($variabel_array);
?>
```



### Output

Jika kamu menjalankan skrip diatas, hasil yang akan muncul pada layar adalah sebagai berikut:

```
string(11) "Halo Dunia!"
int(123)
array(4) {
    [0]=>
        int(1)
    [1]=>
        int(2)
    [2]=>
        string(4) "tiga"
    [3]=>
        float(4.5)
}
```

Kesetiap jenis tipe data akan menampilkan detail yang berbeda. Untuk string, `var_dump()` akan menunjukkan panjang string tersebut dan isinya. Untuk integer dan float, akan ditunjukkan nilai numeriknya. Sementara itu, untuk array, `var_dump()` juga akan menampilkan struktur array, termasuk jumlah elemen, kunci, dan tipe data setiap elemennya.

### `var_dump()` dengan Variabel Bertipe Objek

`var_dump()` juga sangat berguna ketika bekerja dengan objek. Fungsi ini akan menampilkan informasi tentang properti dari objek beserta tipe datanya.

### Contoh dengan Objek

```
<?php
class Mobil {
    public $merk;
    public $model;
```

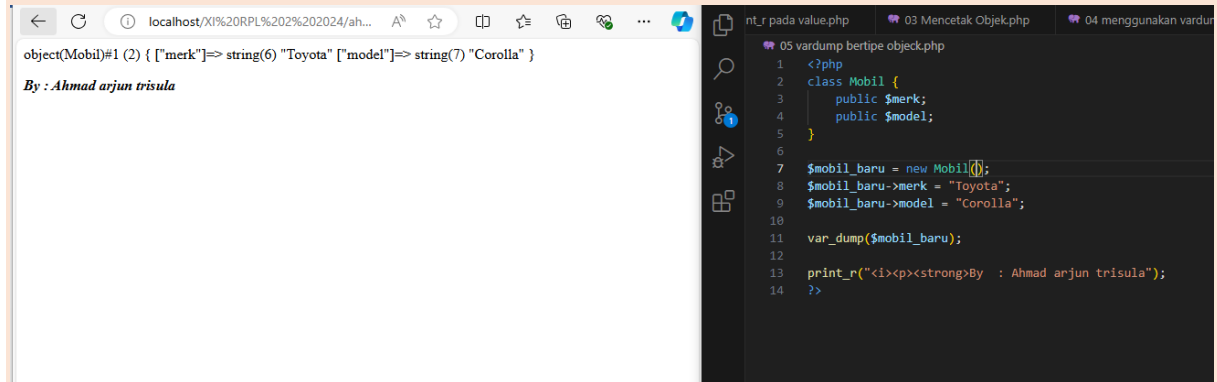
```

}

$mobil_baru = new Mobil();
$mobil_baru->merk = "Toyota";
$mobil_baru->model = "Corolla";

var_dump($mobil_baru);
?>

```



## Output untuk Objek

Outputnya akan memberikan informasi tentang objek Mobil yang meliputi kelasnya dan nilai properti yang dimiliki:

```

object(Mobil)#1 (2) {
    ["merk"]=>
    string(6) "Toyota"
    ["model"]=>
    string(7) "Corolla"
}

```

## Fitur Tambahan var\_dump()

Terakhir, var\_dump() memiliki beberapa fitur tambahan yang bisa membantu dalam debugging.

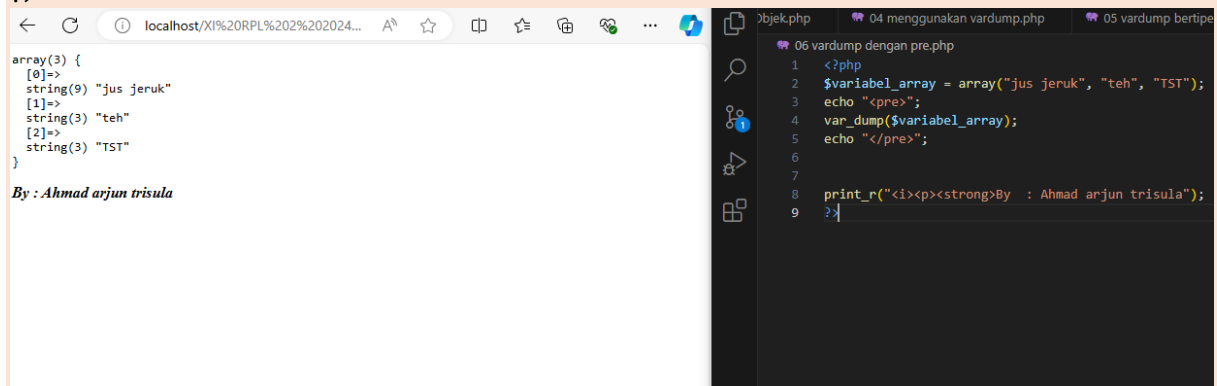
## Menggunakan var\_dump() dengan pre

Untuk membuat output var\_dump() lebih mudah dibaca, terutama ketika menampilkan array atau objek yang besar, kamu bisa menggunakan tag HTML <pre> untuk merapikan format output.

```

<?php
$variabel_array = array("apel", "pisang", "ceri");
echo "<pre>";
var_dump($variabel_array);
echo "</pre>";
?>

```



Eencode dalam tag <pre> akan membuat struktur array lebih terorganisir dan mudah diperiksa.

## Data Boolean dan NULL

Ketika menggunakan var\_dump() dengan tipe data boolean atau nilai NULL, fungsi ini akan dengan jelas menunjukkan nilai TRUE atau FALSE, serta NULL.

```

<?php

```



```
$var_boolean = true;
$var_null = NULL;
```

```
var_dump($var_boolean);
var_dump($var_null);
?>
```

bool(true)  
NULL  
*By : Ahmad arjun trisula*

```
an vardump.php 05 vardump bertipe object.php 06 vardump deng
07 Data Boolean dan NULL.php
1 <?php
2 $var_boolean1 = true;
3 $var_null = NULL;
4
5 var_dump($var_boolean1);
6 print "<p>";
7 var_dump($var_null);
8
9
10 print_r("<i><p><strong>By : Ahmad arjun trisula");
11 ?>
```

### Output:

bool(true)  
NULL

Fungsi `var_dump()` adalah alat yang sangat berharga dalam pengembangan PHP, terutama saat proses debugging kode. Seringkali dapat membantu kamu dengan cepat menemukan masalah dengan data dalam skrip PHP kamu. Ingatlah untuk membersihkan semua pemanggilan `var_dump()` sebelum menerapkan kode kamu ke lingkungan produksi.

## EXIT

Belajar cara menggunakan fungsi `exit` di PHP untuk menghentikan eksekusi skrip dengan panduan langkah demi langkah yang mudah dipahami.

### Menggunakan Fungsi Exit di PHP

Pernahkah kamu ingin menghentikan eksekusi skrip PHP di tengah jalan? PHP menyediakan sebuah fungsi bernama `exit` yang memungkinkanmu untuk melakukan itu. Fungsi `exit` bisa sangat berguna saat kamu ingin mengakhiri skrip saat kondisi tertentu terpenuhi, atau saat kamu telah menyelesaikan tugas yang dibutuhkan dan tidak perlu melanjutkan ke proses selanjutnya.

### Apa Itu Fungsi Exit?

Fungsi `exit` dalam PHP digunakan untuk menghentikan eksekusi skrip. Fungsi ini menerima satu parameter opsional yang dapat digunakan untuk menampilkan pesan atau mengembalikan kode status tertentu kepada sistem operasi jika skrip dijalankan dari command-line.

### Contoh Penggunaan Dasar

Berikut adalah contoh penggunaan yang paling dasar dari fungsi `exit`:

```
echo 'Ini akan tampil di browser.';
exit;
echo 'Ini tidak akan tampil di browser.';
```

Dalam contoh di atas, teks “Ini akan tampil di browser.” akan muncul, tapi teks “Ini tidak akan tampil di browser.” tidak akan pernah dieksekusi karena skrip dihentikan oleh fungsi `exit`.

### Menggunakan Pesan Keluaran

Kamu juga dapat menggunakan fungsi `exit` untuk menampilkan pesan sebelum keluar:

```
exit('Proses telah selesai.');
```

Pesan “Proses telah selesai.” akan ditampilkan di browser sebelum skrip dihentikan.

## Exit dengan Kode Status

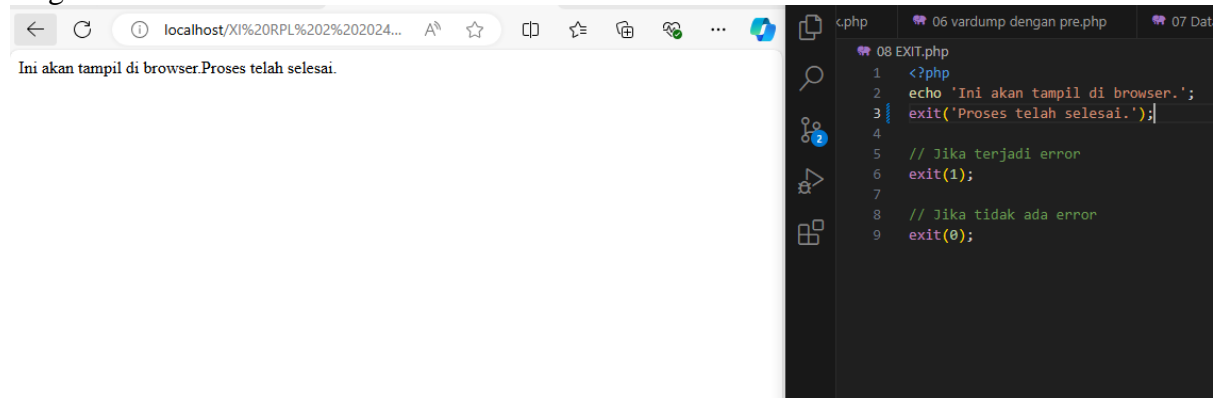
Jika kamu menjalankan skrip PHP dari command-line, exit dapat digunakan untuk mengembalikan kode status. Ini berguna untuk memberi tahu shell tentang hasil eksekusi skrip.

```
// Jika terjadi error
exit(1);

// Jika tidak ada error
exit(0);
```

## Perbedaan antara Exit dan Die

Kamu mungkin juga menemui fungsi die yang sering digunakan dan memiliki fungsi yang sama dengan exit. Sebenarnya, die merupakan alias dari exit, dan keduanya bisa digunakan secara bergantian.



## Contoh Menggunakan Die

```
die('Skrip dihentikan.');
```

Penggunaan die seperti contoh di atas sama seperti jika kamu menggunakan exit dengan parameter pesan.

Memanfaatkan fungsi exit dapat sangat membantumu dalam mengontrol alur dari skrip PHP yang kamu buat. Cukup ingat bahwa setelah fungsi exit dipanggil, tidak ada lagi kode yang akan dieksekusi setelahnya. Gunakanlah fungsi ini dengan bijak untuk menghindari berhentinya skrip sebelum menyelesaikan semua tugasnya.

## DIE

Pelajari cara menggunakan fungsi die() untuk menghentikan skrip PHP kamu dengan contoh yang mudah dipahami.

## Menghentikan Skrip dengan Fungsi die() di PHP

Dalam pemrograman PHP, terkadang kamu perlu menghentikan eksekusi skrip sebelum ia mencapai akhir. Fungsi die() adalah cara yang efektif untuk melakukan ini, khususnya ketika menghadapi error atau situasi yang memerlukan berhentinya program.

### Penggunaan Dasar fungsi die()

Fungsi die() bisa kamu gunakan untuk menghentikan eksekusi skrip PHP dan menampilkan pesan ke pengguna. Hal ini bermanfaat saat terdapat kondisi error atau ketika sebuah kondisi spesifik tidak terpenuhi dan kamu perlu keluar dari skrip.

Contoh penggunaan dasar:

```
if (file_exists('file_important.txt')) {
    echo "File ditemukan.";
} else {
    die("Error: File tidak ditemukan.");
}
```

```
}
```

Pada contoh di atas, jika file `file_important.txt` tidak ditemukan, maka skrip akan berhenti dan menampilkan pesan “Error: File tidak ditemukan.”

### Menambahkan Status HTTP

Kadang kamu ingin mengirim status HTTP tertentu saat skrip dihentikan. Ini bisa dilakukan dengan menggunakan header sebelum memanggil `die()`.

```
if (!user_is_logged_in()) {  
    header('HTTP/1.1 403 Forbidden');  
    die("Error: Kamu tidak memiliki akses.");  
}
```

Dalam contoh di atas, kita mengirim header HTTP 403 Forbidden yang mengindikasikan bahwa akses ditolak sebelum skrip dihentikan.

### Menggabungkan dengan fungsi `exit()`

Fungsi `exit()` dalam PHP hampir sama fungsinya dengan `die()`. Keduanya bisa digunakan secara bergantian. `exit()` sering digunakan ketika ingin memberikan kode keluar tertentu, sementara `die()` sering digunakan untuk menampilkan pesan error.

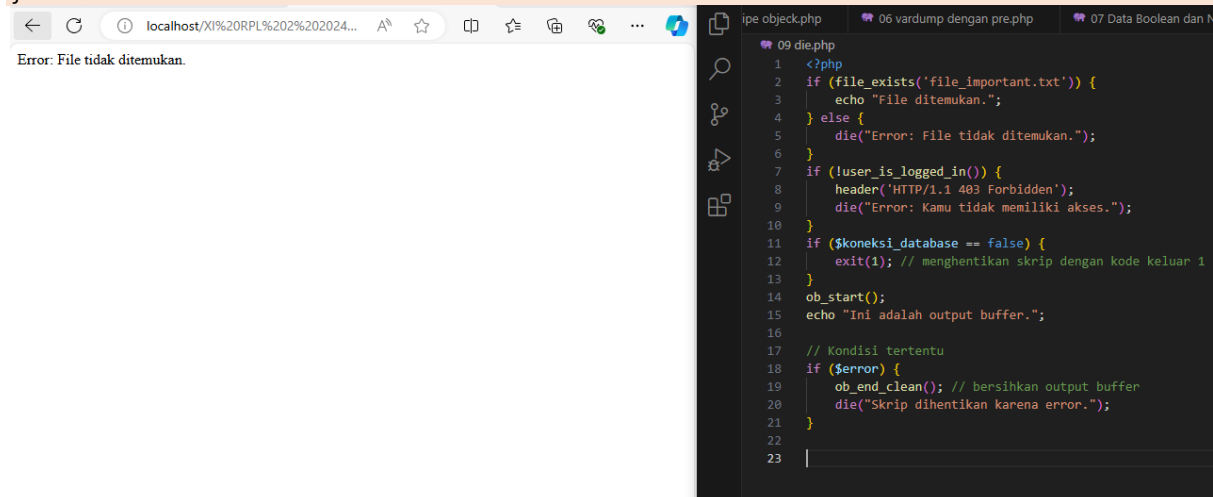
Berikut contoh penggunaan `exit()` dengan kode keluar (exit code):

```
if ($koneksi_database == false) {  
    exit(1); // menghentikan skrip dengan kode keluar 1  
}
```

### Memahami Output Buffering

Ketika menggunakan `die()` atau `exit()`, penting untuk memahami konsep output buffering. Jika output buffering aktif, kamu mungkin perlu membersihkannya sebelum menghentikan skrip.

```
ob_start();  
echo "Ini adalah output buffer.";  
  
// Kondisi tertentu  
if ($error) {  
    ob_end_clean(); // bersihkan output buffer  
    die("Skrip dihentikan karena error.");  
}
```



### Kesimpulan

Fungsi `die()` sangat berguna dalam mengontrol alur eksekusi skrip PHP kamu. Gunakan fungsi ini dengan bijak untuk menangani error atau kondisi yang membutuhkan skrip untuk dihentikan. Ingatlah untuk memberikan informasi yang bermanfaat kepada pengguna tentang mengapa skrip dihentikan. Selain itu, sesuaikan penggunaan status HTTP dan output buffering sesuai dengan situasi yang kamu hadapi.

### A Practical Guide to PHP's `print_r()` and `var_dump()` Functions

When developing PHP applications, it's essential to understand how to debug and inspect variables, arrays, and objects. This tutorial will provide a practical guide to using PHP's `print_r()` and `var_dump()` functions, which are commonly used for this purpose. We will discuss the differences between the two functions, their syntax, and some practical examples. Both `print_r()` and `var_dump()` are built-in PHP functions that can be used to display structured information about variables. This information can be helpful when you need to debug your code or inspect the contents of a variable.

#### Understanding `print_r()`

The `print_r()` function in PHP displays human-readable information about a variable. It recursively prints the contents of arrays and objects, making it an excellent choice for quickly inspecting complex data structures.

The syntax for the `print_r()` function is as follows:

```
print_r(mixed $expression [, bool $return = false ]);
```

Where:

- `$expression` is the variable you want to inspect.
- `$return` is an optional boolean parameter. When set to `true`, the function will return the output as a string instead of printing it. By default, this parameter is set to `false`, which means the function will print the output directly.

#### Example of `print_r()`

Let's take a look at a simple example of using `print_r()` to inspect an array:

```
$array = array(  
    'name' => 'John Doe',  
    'age' => 30,  
    'skills' => array('PHP', 'JavaScript', 'HTML')  
);  
  
print_r($array);
```

Running this code will produce the following output:

```
Array  
(  
    [name] => John Doe  
    [age] => 30  
    [skills] => Array  
        (  
            [0] => PHP  
            [1] => JavaScript  
            [2] => HTML  
        )  
)
```

#### Understanding `var_dump()`

While `print_r()` is useful for displaying human-readable information about a variable, `var_dump()` goes one step further by providing more detailed information, including data types and sizes. This function is particularly helpful when debugging code, as it helps you understand the structure and content of variables more thoroughly.

The syntax for the `var_dump()` function is as follows:

```
var_dump(mixed $expression [, mixed $... ]);
```

Where `$expression` is the variable you want to inspect. You can also pass multiple variables separated by commas, and `var_dump()` will output information for each variable.

## Example of `var\_dump()`

Let's take a look at a simple example of using `var\_dump()` to inspect the same array as in the previous example:

```
$array = array(  
    'name' => 'John Doe',  
    'age' => 30,  
    'skills' => array('PHP', 'JavaScript', 'HTML')  
);  
  
var_dump($array);
```

Running this code will produce the following output:

```
array(3) {  
    ["name"]=>  
    string(8) "John Doe"  
    ["age"]=>  
    int(30)  
    ["skills"]=>  
    array(3) {  
        [0]=>  
        string(3) "PHP"  
        [1]=>  
        string(10) "JavaScript"  
        [2]=>  
        string(4) "HTML"  
    }  
}
```

As you can see, `var\_dump()` provides more detailed information about the variable, including data types and sizes. This can be especially helpful when dealing with large and complex data structures.

## Conclusion

In this tutorial, we covered the practical use of PHP's `print\_r()` and `var\_dump()` functions for inspecting and debugging variables. Both functions are essential tools for any PHP developer, and understanding how to use them effectively can significantly improve your productivity and code quality. When looking to [hire PHP developers](#), ensure they are familiar with these functions and their practical applications.



## MATERI PERTEMUAN 7

### 7 Jenis Operator dalam PHP



#### TAHUKAH KAMU...?

Ada 6 Jenis operator dalam pemrograman PHP yang harus kita ketahui:

- Operator Aritmatika;
- Operator Penugasan atau Assignment;
- Operator Increment & Decrement;
- Operator Relasi atau pembandingan;
- Operator Logika;
- Operator Bitwise;
- Operator Ternary.

### 1. Operator Aritmatika

Operator aritmatika merupakan operator untuk melakukan operasi aritmatika. Operator aritmatika terdiri dari:

Nama Operator	Simbol
1) Penjumlahan	+
2) Pengurangan	-
3) Perkalian	*
4) Pemangkatan	**
5) Pembagian	/
6) Sisa Bagi	%

Contoh:

```
<?php

$a = 5;
$b = 2;

// penjumlahan
$c = $a + $b;
echo "$a + $b = $c";
echo "<hr>";

// pengurangan
$c = $a - $b;
echo "$a - $b = $c";
echo "<hr>";

// Perkalian
$c = $a * $b;
echo "$a * $b = $c";
echo "<hr>";

// Pembagian
$c = $a / $b;
echo "$a / $b = $c";
echo "<hr>";

// modulus Sisa bagi
$c = $a % $b;
echo "$a % $b = $c";
echo "<hr>";

// Pangkat
$c = $a ** $b;
echo "$a ** $b = $c";
```

```
echo "<hr>";  
?>
```

Mula-mula kita punya dua variabel, yaitu **\$a** dan **\$b** dengan nilai awal sebagai berikut:

```
$a = 5;  
$b = 2;
```

Kemudian kita menggunakan operator aritmatika untuk melakukan operasi terhadap dua nilai atau variabel tersebut.

Lalu hasilnya disimpan di dalam variabel **\$c**.

Maka hasilnya:

O ya, perhatikan juga simbol-simbol operator yang dipakai.

Pada matematika, perkalian biasanya menggunakan simbol **x**. Namun di dalam pemrograman perkalian menggunakan simbol **\***.

```
// salah  
$c = 7 x 4;  
  
// benar  
$c = 7 * 4;
```

Lalu untuk operator **%** (modulo), ini adalah operator untuk menghitung sisa bagi.

Misalnya:

```
// kita punya 5  
$a = 5;  
  
// lalu kita bagi dengan 2  
$c = $a / 2;
```

Maka variabel **\$c** akan bernilai **1**, karena **5** dibagi **2** sisanya **1**.

Biar lebih mudah, coba bayangkan seperti ini:

Kamu punya permen **5** biji, lalu dibagi berdua dengan adikmu. Biar adil, sama-sama dapat dua biji. Nah, ada sisanya satu yang belum dibagi.

Sisa inilah yang menjadi hasil modulo.

## Modulus (%)

Modulus adalah operator yang memberikan sisa dari pembagian dua nilai.

```
$a = 10;  
$b = 3;  
$hasil = $a % $b; // Hasilnya adalah 1
```

## Eksponensial (\*\*)

Eksponensial menggunakan operator **\*\*** untuk mencari hasil pemangkatan suatu bilangan.

```
$a = 2;  
$b = 3;  
$hasil = $a ** $b; // Hasilnya adalah 8 (2 pangkat 3)
```

## Prioritas Operator

Operator aritmatika memiliki tingkat prioritas tertentu dalam eksekusi. Secara default, PHP mengikuti urutan operasi matematika dasar (PEMDAS/PEDMAS) yaitu:

### Parentheses (tanda kurung)

Exponential (eksponensial)

Multiplication dan Division (perkalian dan pembagian)

Addition dan Subtraction (penjumlahan dan pengurangan)

Pastikan kamu mengelompokkan perhitungan dengan tanda kurung untuk menjamin hasil yang benar.

```
$hasil = (5 + 5) * 2; // Hasilnya adalah 20, bukan 15
```

Gunakan operator aritmatika dengan benar untuk menyelesaikan berbagai perhitungan matematika dalam program PHP kamu. Dengan memahami operator-operator ini, kamu bisa mengambil langkah selanjutnya untuk membangun logika program yang lebih kompleks.

## 2. Operator Penugasan (Assignment)

Operator berikutnya yang harus kamu ketahui adalah operator penugasan atau *assignment*.

Yap! dari namanya saja sudah bisa ditebak.

Operator ini adalah operator untuk memberikan tugas kepada variabel.

Biasanya digunakan untuk mengisi nilai.

Contoh:

```
$a = 32;
```

Sama dengan (=) adalah operator penugasan untuk mengisi nilai.

Selain sama dengan, terdapat juga beberapa operator penugasan seperti:

Nama Operator	Symbol
1) Pengisian Nilai	=
2) Pengisian dan Penambahan	+=
3) Pengisian dan Pengurangan	-=
4) Pengisian dan Perkalian	*=
5) Pengisian dan Pemangkatan	**=
6) Pengisian dan Pembagian	/=
7) Pengisian dan Sisa bagi	%=
8) Pengisian dan Peggabungan (string)	.=

### Apa bedanya dengan operator aritmatika?

Operator penugasan digunakan untuk mengisi nilai dan juga menghitung dengan operasi aritmatika. Sedangkan operator aritmatika hanya berfungsi untuk menghitung saja.

Sebagai contoh:

```
$speed = 83;

// ini operator aritmatika
$speed = $speed + 10;

// maka nilai speed akan samadengan 83 + 10 = 93

// ini operator penugasan
$speed += 10;

// sekarang nilai speed akan menjadi 93 + 10 = 103
```

Output:

```
dian@petanikode:~$ php -a
Interactive mode enabled

php > $speed = 83;
php > echo $speed;
83
php > $speed = $speed + 10; ← Menggunakan operator aritmatika
php > echo $speed;
93
php > $speed += 10; ← Menggunakan operator penugasan
php > echo $speed;
103
php > █
```

Perhatikan operasi ini:



```
$speed = $speed + 10;  
// dan  
$speed += 10;
```

Kedua operasi tersebut merupakan operasi yang sama. Hanya saja yang atas menggunakan operator aritmatika dan yang bawah menggunakan operator penugasan. Bisa dibilang, operator penugasan adalah bentuk yang lebih sederhana dari ekspresi seperti di atas. Penggunaan operator penugasan akan sering kita temukan saat membuat program.

### 3. Operator Increment & Decrement

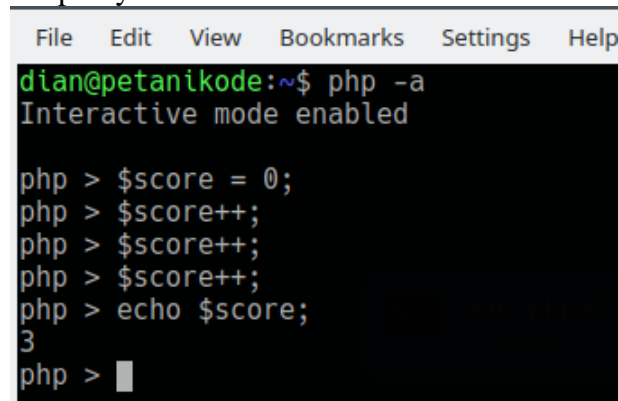
Operator *increment* dan *decrement* merupakan operator yang digunakan untuk menambah **+1** (tambah satu) dan mengurangi **-1** (kurangi dengan satu).

Operator *increment* menggunakan simbol **++**, sedangkan *decrement* menggunakan simbol **--**.

Contoh:

```
$score = 0;  
  
$score++;  
$score++;  
$score++;  
  
echo $score;
```

Outputnya:



```
dian@petanikode:~$ php -a  
Interactive mode enabled  
  
php > $score = 0;  
php > $score++;  
php > $score++;  
php > $score++;  
php > echo $score;  
3  
php > █
```

Nilai **\$score** akan menjadi **3**, karena kita melakukan *increment* sebanyak 3x.

### 4. Operator Relasi

Operator relasi adalah operator untuk membandingkan dua buah nilai. Hasil operasi dari operator relasi akan menghasilkan nilai dengan tipe data *boolean*, yaitu **true** (benar) dan **false** (salah).

Berikut ini daftar operator relasi:

Nama Operator	Simbol
1) Lebih Besar	>
2) Lebih Kecil	<
3) Sama Dengan	== atau ===
4) Tidak Sama dengan	!= atau !==
5) Lebih Besar Sama dengan	>=
6) Lebih Kecil Sama dengan	<=

Mari kita coba dalam program: **relasi.php**

```
<?php  
  
$a = 6;  
$b = 2;  
  
// menggunakan operator relasi lebih besar  
$c = $a > $b;  
echo "$a > $b: $c";
```

```

echo "<hr>";

// menggunakan operator relasi lebih kecil
$c = $a < $b;
echo "$a < $b: $c";
echo "<hr>";

// menggunakan operator relasi lebih sama dengan
$c = $a == $b;
echo "$a == $b: $c";
echo "<hr>";

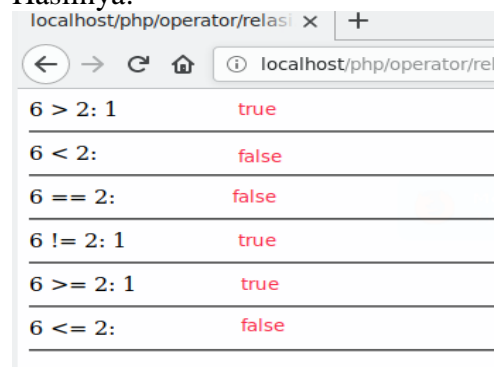
// menggunakan operator relasi lebih tidak sama dengan
$c = $a != $b;
echo "$a != $b: $c";
echo "<hr>";

// menggunakan operator relasi lebih besar sama dengan
$c = $a >= $b;
echo "$a >= $b: $c";
echo "<hr>";

// menggunakan operator relasi lebih kecil sama dengan
$c = $a <= $b;
echo "$a <= $b: $c";
echo "<hr>";

```

Hasilnya:



6 > 2: 1	true
6 < 2:	false
6 == 2:	false
6 != 2: 1	true
6 >= 2: 1	true
6 <= 2:	false

Perhatikan!

Di sana kita mendapatkan nilai **1** untuk **true** sedangkan **false** tidak ditampilkan atau **0**. Apakah ini salah?

Tidak, memang seperti itulah sifat dari fungsi **echo** di PHP.

Nilai dengan tipe data boolean biasanya tidak untuk ditampilkan. Biasanya digunakan untuk pembuatan kondisi pada percabangan.

Contohnya seperti ini:

```

<?php
$total_belanja = 150000;

if($total_belanja > 100000){
    echo "Anda dapat hadiah!";
}

```

## 5. Operator Logika

Jika kamu pernah belajar logika matematika, kamu pasti tidak akan asing dengan operator ini. Operator logika adalah operator untuk melakukan operasi logika seperti AND, OR, dan NOT.

Operator logika terdiri dari:

Nama Operator	Simbol
1) Logika AND	&&
2) Logika OR	
3) Negasi/kebalikan/ NOT	!

Mari kita coba dalam program: **logika.php**

```
<?php

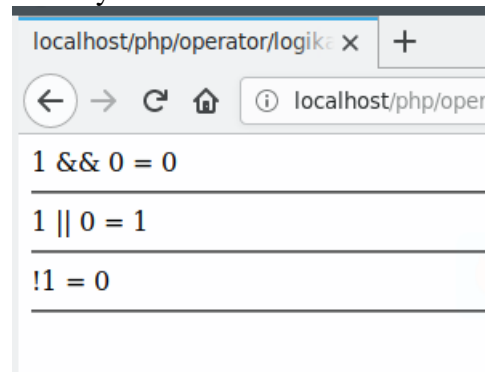
$a = true;
$b = false;

// variabel $c akan bernilai false
$c = $a && $b;
printf("%b && %b = %b", $a,$b,$c);
echo "<hr>";

// variabel $c akan bernilai true
$c = $a || $b;
printf("%b || %b = %b", $a,$b,$c);
echo "<hr>";

// variabel $c akan bernilai false
$c = !$a;
printf("!%b = %b", $a, $c);
echo "<hr>";
```

Hasilnya:



### Perhatikan!

Pada contoh di atas, kita menggunakan fungsi **printf()** untuk mencetak memformat dan mencetak teks.

Namun akan tetap menampilkan **1** untuk **true** dan **0** untuk **false**.

Operator logika sama seperti operator relasi, ia akan menghasilkan nilai dengan tipe data *boolean*.

Perhatikanlah hasil yang di dapatkan ketika menggunakan operator **&&** (AND), **||** (OR), dan **!** (NOT).

Operator **&&** akan menghasilkan **true** apabila nilai kiri dan kanan bernilai **true**. Sedangkan operator **||** akan menghasilkan **false** saat nilai kiri dan kanan bernilai **false**.

Coba cek kembali hukum logika AND, OR, dan NOT.

AND		Hasil
true	true	true
true	false	false
false	true	false
false	false	false

OR		Hasil
true	true	true
true	false	true
false	true	true
false	false	false

NOT	Hasil
true	false
false	true

## 6. Operator Bitwise

Operator bitwise merupakan operator yang digunakan untuk operasi bit (biner).

Operator ini terdiri dari:

Nama	Simbol
AND	&
OR	
XOR	^

Negasi/kebalikan	~
Left Shift	<<
Right Shift	>>

Operator ini berldiuntuk tipe data **int**, **long**, **short**, **char**, dan **byte**.

Operator ini akan menghitung dari bit-ke-bit.

Misalnya, kita punya variabel **a = 60** dan **b = 13**.

Bila dibuat dalam bentuk biner, akan menjadi seperti ini:

a = 00111100

b = 00001101

Kemudian, dilakukan operasi bitwise

**Operasi AND**

a = 00111100

b = 00001101

a & b = 00001100

**Operasi OR**

a = 00111100

b = 00001101

a | b = 00111101

**Operasi XOR**

a = 00111100

b = 00001101

a ^ b = 00110001

**Opearsi NOT (Negasi/kebalikan)**

a = 00111100

~a = 11000011

Konsepnya memang hampir sama dengan opeartor Logika. Bedanya, Bitwise digunakan untuk biner.

Untuk lebih jelasnya mari kita coba dalam program.

```
<?php

$a = 60;
$b = 13;

// bitwise AND
$c = $a & $b;
echo "$a & $b = $c";
echo "<br>";

// bitwise OR
$c = $a | $b;
echo "$a | $b = $c";
echo "<br>";

// bitwise XOR
$c = $a ^ $b;
echo "$a ^ $b = $c";
echo "<br>";

// Shift Left
$c = $a << $b;
echo "$a << $b = $c";
echo "<br>";

// Shift Right
$c = $a >> $b;
echo "$a >> $b = $c";
echo "<br>";
```

Hasilnya:

```
localhost/php/operator/bitwise x +
localhost/php
60 & 13 = 12
60 | 13 = 61
60 ^ 13 = 49
60 << 13 = 491520
60 >> 13 = 0
```

## 7. Operator Ternary

Operator ternary adalah operator untuk membuat sebuah kondisi. Simbol yang digunakan adalah tanda tanya (?) dan titik dua (:).



Pada contoh di atas, “Kamu suka aku” adalah pertanyaan atau kondisi yang akan diperiksa. Kalau jawabannya benar, maka iya. Sebaliknya akan tidak. Untuk lebih jelasnya, mari kita coba...

```
<?php
$suka = true;

// menggunakan operator ternary
$jawab = $suka ? "iya": "tidak";

// menampilkan jawaban
echo $jawab;
```

Hasilnya:

```
Mozilla Firefox
localhost/php/operator/terna x +
localhost/php/operator/ternary.php ...
iya
```

Cobalah untuk mengganti nilai variabel **\$suka** menjadi **false**, maka hasil outputnya akan **tidak**.

## OPERATOR STRING

Pelajari berbagai metode string dalam PHP untuk manipulasi dan pemrosesan teks dengan mudah dan efektif. Artikel ini menjelaskan penggunaan metode-metode tersebut dengan contoh yang sederhana.

Sastra digital memiliki peran sentral dalam pemrograman, dan PHP, sebagai salah satu bahasa pemrograman populer untuk pengembangan web, menyediakan sejumlah metode string yang memudahkan pengolahan teks. Metode string di PHP memungkinkan kamu untuk melakukan berbagai operasi pada teks, seperti pencarian, penggantian, pemotongan, dan formatting. Berikut ini adalah beberapa metode string yang sering digunakan dalam PHP.

### strlen

Digunakan untuk mendapatkan panjang sebuah string.

```
$text = "Hello World!";  
echo strlen($text); // Outputs: 12
```

### **str\_word\_count**

Menghitung jumlah kata dalam string.

```
$text = "Hello World!";  
echo str_word_count($text); // Outputs: 2
```

### **strrev**

Memutar balik urutan karakter dalam string.

```
$text = "Hello World!";  
echo strrev($text); // Outputs: "!dlrow olleH"
```

### **strtoupper dan strtolower**

Mengubah semua karakter dalam string menjadi huruf besar (strtoupper) atau huruf kecil (strtolower).

```
echo strtoupper("Hello World!"); // Outputs: "HELLO WORLD!"  
echo strtolower("Hello World!"); // Outputs: "hello world!"
```

### **ucwords dan lcfirst**

ucwords mengkapitalkan huruf pertama dari setiap kata dalam string, sedangkan lcfirst mengubah huruf pertama dari string menjadi huruf kecil.

```
echo ucwords("hello world!"); // Outputs: "Hello World!"  
echo lcfirst("Hello"); // Outputs: "hello"
```

### **trim, ltrim, dan rtrim**

Menghilangkan whitespace (atau karakter lain) dari awal dan/atau akhir string. trim menghilangkan dari kedua sisi, ltrim hanya dari sisi kiri, dan rtrim hanya dari sisi kanan.

```
$text = " Hello World! ";  
echo "[" . trim($text) . "]; // Outputs: "[Hello World!]"  
echo "[" . ltrim($text) . "]; // Outputs: "[Hello World! ]"  
echo "[" . rtrim($text) . "]; // Outputs: "[ Hello World!]"
```

### **strpos dan strrpos**

Menemukan posisi pertama (strpos) atau terakhir (strrpos) dari suatu substring dalam string.

```
$text = "Hello World!";  
echo strpos($text, "World"); // Outputs: 6  
echo strrpos($text, "o"); // Outputs: 7
```

### **str\_replace**

Menggantikan semua kemunculan substring dalam string dengan string lain.

```
$text = "Hello World!";  
echo str_replace("World", "PHP", $text); // Outputs: "Hello PHP!"
```

### **substr**

Mengembalikan sebagian string berdasarkan posisi awal dan panjang yang ditentukan.

```
$text = "Hello World!";  
echo substr($text, 0, 5); // Outputs: "Hello"  
echo substr($text, 6); // Outputs: "World!"
```

### **strcmp dan strcasecmp**

Melakukan perbandingan dua string dengan case-sensitive (strcmp) atau case-insensitive (strcasecmp).

```
echo strcmp("Hello World!", "Hello World!"); // Outputs: 0  
echo strcasecmp("hello world!", "HELLO WORLD!"); // Outputs: 0
```

## **str\_split**

Mengkonversi string menjadi array.

```
$text = "Hello";  
print_r(str_split($text, 2)); // Outputs: Array ( [0] => He [1] => ll [2] => o )
```

Dengan memahami dan menggunakan metode-metode string ini, kamu bisa mengolah data teks di PHP dengan lebih mudah dan efisien. Penggunaan yang benar dari fungsi-fungsi ini akan membuat kode kamu lebih bersih, cepat, dan dapat diandalkan. Selamat mencoba dan eksplorasi lebih lanjut!

## **OPERATOR NUMBERS**

Pelajari berbagai metode untuk memanipulasi angka dalam PHP termasuk pembulatan, format, dan operasi matematika dasar dengan penjelasan yang mudah dimengerti.

Manipulasi angka adalah salah satu aspek penting dalam pemrograman. Di PHP, kamu bisa menggunakan berbagai fungsi bawaan untuk memanipulasi angka untuk berbagai kebutuhan seperti perhitungan matematika, pembulatan, dan format angka. Berikut adalah beberapa metode yang dapat kamu gunakan untuk bekerja dengan angka di PHP.

### **Pembulatan Angka**

PHP menyediakan beberapa fungsi untuk membulatkan angka. Ini sangat berguna, misalnya, ketika kamu ingin membulatkan hasil perhitungan.

#### **round()**

Fungsi round() digunakan untuk membulatkan angka ke bilangan bulat terdekat atau ke presisi desimal tertentu.

```
echo round(3.4); // Hasil: 3  
echo round(3.5); // Hasil: 4  
echo round(3.6); // Hasil: 4  
echo round(3.678, 2); // Hasil: 3.68
```

#### **ceil()**

Fungsi ceil() membulatkan angka ke atas ke bilangan bulat terdekat.

```
echo ceil(3.3); // Hasil: 4  
echo ceil(3.9); // Hasil: 4
```

#### **floor()**

Fungsi floor() membulatkan angka ke bawah ke bilangan bulat terdekat.

```
echo floor(3.3); // Hasil: 3  
echo floor(3.9); // Hasil: 3
```

### **Format Angka**

Untuk menampilkan angka dengan format tertentu, PHP memiliki fungsi number\_format().

#### **number\_format()**

Fungsi ini memformat angka dengan memisahkan ribuan dan menentukan jumlah desimal.

```
echo number_format(1000); // Hasil: 1,000  
echo number_format(1000.75, 2, '.', ','); // Hasil: 1,000.75
```

### **Operasi Matematika Dasar**

PHP memiliki operator matematika standar seperti penambahan, pengurangan, perkalian, dan pembagian.

```
$angka1 = 10;  
$angka2 = 5;
```

```
// Penambahan
echo $angka1 + $angka2; // Hasil: 15

// Pengurangan
echo $angka1 - $angka2; // Hasil: 5

// Perkalian
echo $angka1 * $angka2; // Hasil: 50

// Pembagian
echo $angka1 / $angka2; // Hasil: 2
```

### Fungsi Matematika Lainnya

PHP memiliki fungsi matematika bawaan lainnya yang juga berguna dalam pengolahan angka.

#### **abs()**

Menghitung nilai absolut atau nilai positif dari angka.

```
echo abs(-4.2); // Hasil: 4.2
```

#### **pow()**

Menghitung nilai pangkat dari angka.

```
echo pow(2, 3); // Hasil: 8
```

#### **sqrt()**

Menghitung akar kuadrat dari angka.

```
echo sqrt(16); // Hasil: 4
```

#### **max() dan min()**

Mendapatkan nilai tertinggi dan terendah dari sejumlah angka atau array.

```
echo max(2, 3, 1); // Hasil: 3
echo min(array(2, 3, 1)); // Hasil: 1
```

Dengan memahami fungsi-fungsi ini, kamu bisa melakukan berbagai operasi matematika dan manipulasi angka di PHP dengan lebih mudah. Kombinasikan fungsi-fungsi tersebut sesuai kebutuhan program yang sedang kamu kembangkan.





## MATERI PERTEMUAN 8

### Memahami Percabangan dan Structur Kontrolnya



#### TAHUKAH KAMU...?

Ada 6 Jenis percabangan dalam pemrograman PHP yang harus kita ketahui:

- Percabangan If
- Percabangan If/Else
- Percabangan If/Elseif/Else
- Percabangan Switch/Case
- Percabangan dengan Operator Ternary
- Percabangan Bersarang

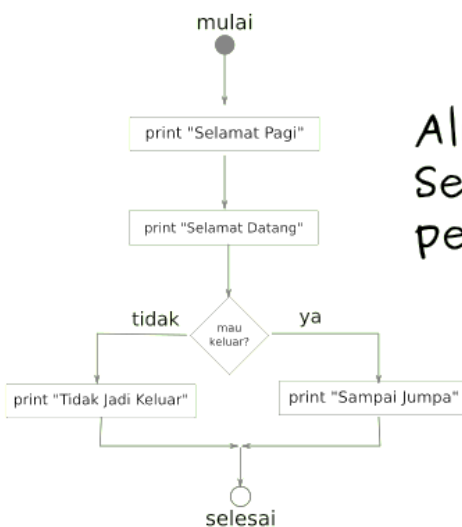
#### A. Percabangan, Logika Dalam Pemrograman

Percabangan adalah sebutan untuk alur program yang bercabang. Pada diagram alur, kita sering menggambar alur program seperti ini:



Alur Program  
Sederhana, tanpa  
percabangan

Apabila kita ingin menambahkan percabangan, kita akan membuatnya seperti ini:



Alur Program  
Sederhana dengan  
percabangan

Pada kesempatan ini, kita akan pelajari tentang percabangan sampai tuntas dan membuat beberapa contoh program.

## Percabangan If

Bentuk yang paling sederhana dari percabangan adalah “If” saja. Biasanya digunakan saat hanya ada satu tindakan yang harus dilakukan.

Bentuknya seperti ini:

```
<?php

if (<kondisi>){
    // eksekusi kode ini
}
```

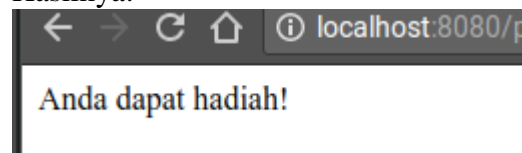
Jika kondisi benar, maka eksekusi kode yang ada di dalamnya. **<kondisi>** bisa kita isi dengan nilai *boolean* atau kita bisa buat pernyataan untuk menghasilkan nilai *boolean*.

Contoh:

```
<?php
$total_belanja = 150000;

if($total_belanja > 100000){
    echo "Anda dapat hadiah!";
}
```

Hasilnya:



Perhatikan contoh di atas!

Teks **Anda dapat hadiah!** hanya akan ditampilkan saat kondisi variabel **\$total\_belanja** bernilai di atas **100000**. Kalau di bawah **100000**, tidak akan menampilkan apa-apa.

Kondisi yang digunakan pada contoh adalah:

```
$total_belanja > 100000
```

Kondisi atau pernyataan ini akan bernilai **true** dan **false**. Jika **true** (benar), maka kode yang ada di dalamnya akan dieksekusi. Namun, apabila **false** maka tidak akan mengeksekusinya.

## Percabangan If/Else

Percabangan If/Else memiliki dua pilihan. Jika **<kondisi>** bernilai **false**, maka blok else akan dikerjakan. Contoh:

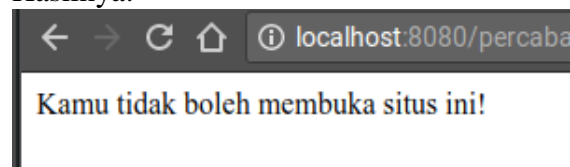
```
<?php

$umur = 13;

if ($umur < 18 ){
    echo "Kamu tidak boleh membuka situs ini!";
} else {
    echo "Selamat datang di website kami!";
}

?>
```

Hasilnya:



Sekarang coba ubah nilai **\$umur** menjadi **19**:

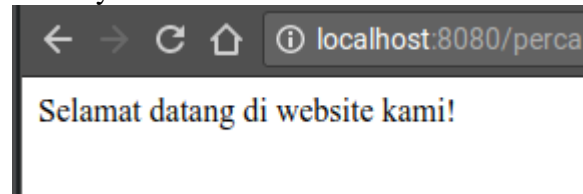
```
<?php

$umur = 19;

if ($umur < 18 ){
    echo "Kamu tidak boleh membuka situs ini!";
} else {
    echo "Selamat datang di website kami!";
}

?>
```

Hasilnya:



### Percabangan If/Elseif/Else

Percabangan If/Elseif/Else memiliki lebih dari dua pilihan kondisi.

Contoh:

```
<?php

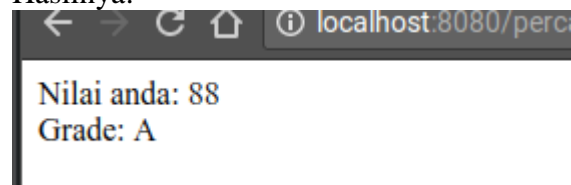
$nilai = 88;

if ($nilai > 90) {
    $grade = "A+";
} elseif($nilai > 80){
    $grade = "A";
} elseif($nilai > 70){
    $grade = "B+";
} elseif($nilai > 60){
    $grade = "B";
} elseif($nilai > 50){
    $grade = "C+";
} elseif($nilai > 40){
    $grade = "C";
} elseif($nilai > 30){
    $grade = "D";
} elseif($nilai > 20){
    $grade = "E";
} else {
    $grade = "F";
}

echo "Nilai anda: $nilai<br>";
echo "Grade: $grade";

?>
```

Hasilnya:



Coba ubah variabel **\$nilai** menjadi **54** dan perhatikanlah hasilnya!

### Percabangan Switch/Case

Percabangan Switch/Case adalah bentuk lain dari percabangan If/Elseif/Else.

Format penulisannya seperti ini:

```
<?php
```

```

switch($variabel){
    case <konidisi>:
        // eksekusi kode ini
        break;
    case <kondisi2>:
        // eksekusi kode ini
        break;
    default:
        // eksekusi kode ini
}
?>

```

### Contoh:

```

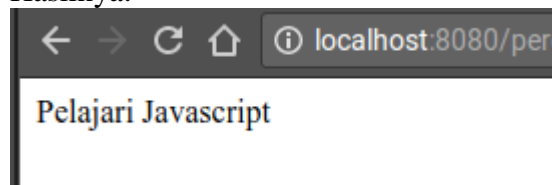
<?php
$level = 3;

switch($level){
    case 1:
        echo "Pelajari HTML";
        break;
    case 2:
        echo "Pelajari CSS";
        break;
    case 3:
        echo "Pelajari Javascript";
        break;
    case 4:
        echo "Pelajari PHP";
        break;
    default:
        echo "Kamu bukan programmer!";
}
?>

```

Ada 5 pilihan dalam kondisi di atas. Pilihan **default** akan dipilih apabila nilai variabel **\$level** tidak ada dalam pilihan **case**.

Hasilnya:



Coba ubah nilai variabel **\$level** dan perhatikanlah hasilnya.

### Percabangan dengan Operator Ternary

Percabangan menggunakan operator ternary adalah bentuk sederhana dari percabangan If/Else.

Formatnya seperti ini:

```

<?php
<kondisi> ? benar : salah;

?>

```

### Contoh:

```

<?php
$suka = true;
$suka ? echo "Dijuga suka kamu": echo "Baiklah!";

?>

```

Atau bisa juga dibuat seperti ini:

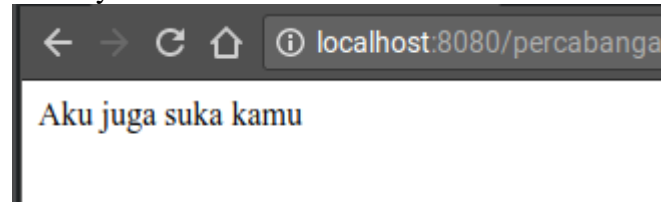
```
<?php
```

```
$suka = true;
echo $suka ? "Dijuga suka kamu": "Baiklah!";

?>
```

Artinya: jika variabel **\$suka** bernilai **true** maka cetak "**Dijuga suka kamu**". Tapi kalau bernilai **false**, maka cetak "**Baiklah!**".

Hasilnya:



### Percabangan Bersarang

Percabangan bersarang artinya ada percabangan di dalam percabangan (*nested*).

Contoh:

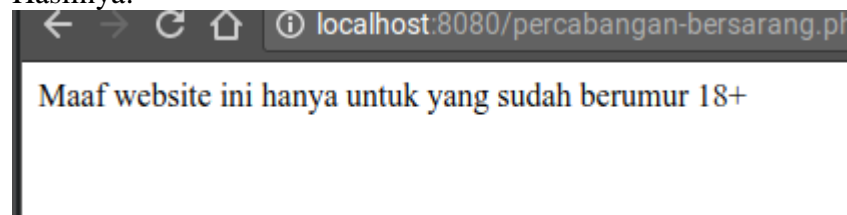
```
<?php
```

```
$umur = 17;
$menikah = false;

if($umur > 18){
    if($menikah){
        echo "Selamat datang pak!";
    } else {
        echo "Selamat datang wahai pemuda!";
    }
} else {
    echo "Maaf website ini hanya untuk yang sudah berumur 18+";
}

?>
```

Hasilnya:



## STRUKTUR KONTROL PERCABANGAN

Pelajari tentang Struktur Kontrol pada PHP, termasuk penggunaan pernyataan if, else, switch, while, dan for untuk mengatur alur program.

Struktur kontrol merupakan salah satu konsep penting dalam pemrograman PHP yang memungkinkan kamu untuk mengatur alur program. Struktur kontrol ini termasuk pernyataan kondisional dan perulangan yang menentukan jalannya kode berdasarkan kondisi tertentu atau melakukan iterasi beberapa kali.

### Pernyataan If

Pernyataan if digunakan untuk menjalankan kode hanya jika kondisi tertentu terpenuhi.

```
if ($kondisi) {  
    // Kode dijalankan jika kondisi terpenuhi  
}
```

Contoh penggunaan:

```
$nilai = 75;  
if ($nilai > 70) {  
    echo "Selamat, kamu lulus!";  
}
```

### Pernyataan Else

Untuk menentukan blok kode yang akan dijalankan jika kondisi if tidak terpenuhi, kamu bisa menggunakan pernyataan else.

```
if ($kondisi) {  
    // Kode dijalankan jika kondisi terpenuhi  
} else {  
    // Kode dijalankan jika kondisi tidak terpenuhi  
}
```

Contoh penggunaan:

```
$nilai = 60;  
if ($nilai > 70) {  
    echo "Selamat, kamu lulus!";  
} else {  
    echo "Maaf, kamu harus mengulang.";  
}
```

### Pernyataan Elseif

Ketika kamu memiliki beberapa kondisi yang harus diperiksa, gunakan pernyataan elseif.

```
if ($kondisi1) {  
    // Kode dijalankan jika kondisi1 terpenuhi  
} elseif ($kondisi2) {  
    // Kode dijalankan jika kondisi1 tidak terpenuhi dan kondisi2 terpenuhi  
} else {  
    // Kode dijalankan jika tidak ada kondisi yang terpenuhi  
}
```

Contoh penggunaan:

```
$nilai = 80;  
if ($nilai >= 90) {  
    echo "Nilai kamu A";  
} elseif ($nilai >= 80) {  
    echo "Nilai kamu B";  
} else {  
    echo "Nilai kamu C";  
}
```

### Pernyataan Switch

Pernyataan switch digunakan sebagai alternatif untuk pernyataan if multipercabangan yang panjang dan rumit.

```
switch ($var) {  
    case 'nilai1':  
        // Kode dijalankan jika $var sama dengan nilai1  
        break;  
    case 'nilai2':  
        // Kode dijalankan jika $var sama dengan nilai2  
        break;  
    default:  
        // Kode dijalankan jika $var tidak sama dengan semua case  
}
```

Contoh penggunaan:

```
$grade = 'B';  
switch ($grade) {  
    case 'A':  
        echo "Luar biasa!";  
        break;  
    case 'B':  
    case 'C':  
        // Kode untuk B dan C  
}
```

```
    echo "Bagus!";  
    break;  
    default:  
        echo "Tetap semangat!";  
}
```

## PENGAMBILAN KEPUTUSAN DENGAN BOOLEAN

Pelajari tentang tipe data boolean di PHP, termasuk cara kerja, definisi, serta contoh penggunaan boolean dalam pemrograman PHP.

PHP, seperti bahasa pemrograman lainnya, menggunakan tipe data untuk mengidentifikasi jenis nilai yang bisa disimpan dalam variabel. Salah satu tipe data dasar yang ada dalam PHP adalah tipe data boolean. Artikel ini akan membahas apa itu tipe data boolean dan bagaimana cara menggunakannya dalam PHP.

### Apa Itu Tipe Data Boolean?

Tipe data boolean merepresentasikan dua kemungkinan nilai: true atau false. Dalam pemrograman, boolean sering digunakan untuk pengambilan keputusan dan control flow, misalnya dalam struktur kondisional seperti if...else dan while.

### Mendefinisikan Boolean

Untuk mendefinisikan variabel boolean di PHP, kamu cukup memberikan nilai true atau false kepada variabel tersebut. Ingat bahwa nilai boolean tidak case-sensitive di PHP, yang berarti TRUE, True, dan true sama saja.

```
$benar = true;  
$salah = false;
```

### Boolean dalam Pengambilan Keputusan

Sangat umum menggunakan tipe data boolean dalam statement kondisional. Berikut contoh sederhana:

```
$login_sukses = true;  
  
if ($login_sukses) {  
    echo "Selamat datang pengguna!";  
} else {  
    echo "Login gagal, silakan coba lagi.";  
}
```

Dalam contoh di atas, jika \$login\_sukses bernilai true, maka program akan menampilkan “Selamat datang pengguna!”.  
“.

### Konversi ke Boolean

PHP secara otomatis akan mengkonversi nilai ke tipe boolean jika diperlukan. Ini terjadi dalam kontrol struktur seperti if, while, dan expression lain yang mengharapkan nilai boolean. Berikut adalah aturan konversi ke boolean:

Nilai 0 (zero), 0.0 (zero float), string kosong " " dan string "0", array kosong, variabel khusus NULL, serta objek yang belum di-assign akan dianggap sebagai false.

Nilai selain yang disebutkan di atas akan dianggap sebagai true.

Contoh konversi otomatis:

```
$nilai = 0;  
  
if ($nilai) {  
    echo "Variabel ini bernilai true";  
} else {  
    echo "Variabel ini bernilai false";  
}  
  
// Keluaran akan "Variabel ini bernilai false" karena nilai 0 dianggap false.
```

## Boolean dan Fungsi

Fungsi-fungsi dalam PHP juga sering mengembalikan nilai boolean untuk mengindikasikan keberhasilan atau kegagalan operasi. Sebagai contoh, fungsi file-handling `file_put_contents()` mengembalikan `false` jika gagal menulis ke file, dan jumlah byte yang ditulis jika berhasil. Kamu bisa menggunakan nilai boolean ini untuk mengecek hasil operasi:

```
$isiFile = "Contoh teks di dalam file.";
$status = file_put_contents("example.txt", $isiFile);

if ($status !== false) {
    echo "Penulisan file berhasil!";
} else {
    echo "Penulisan file gagal.";
}
```

Pemahaman tentang tipe data boolean adalah fondasi penting dalam pemrograman PHP. Dengan kemampuan untuk menilai kondisi dan mengontrol aliran program, kamu dapat menulis kode yang bersih, efisien, dan mudah dibaca.

## IF DAN ELSE

Pelajari cara menggunakan struktur kontrol `if` dan `else` dalam PHP untuk membuat keputusan dalam kode dengan penjelasan dan contoh yang mudah dipahami.

Mempelajari struktur kontrol seperti `if` dan `else` sangat penting dalam pemrograman PHP karena membantu kamu membuat keputusan berdasarkan kondisi tertentu.

### Pengenalan If dan Else

Dalam PHP, `if` dan `else` digunakan untuk mengeksekusi kode yang berbeda tergantung pada hasil evaluasi suatu kondisi. Jika kondisi bernilai `true`, kode di dalam blok `if` akan dijalankan. Sebaliknya, jika kondisi bernilai `false`, kode di dalam blok `else` akan dijalankan.

### Penulisan Sintaks If

```
if (kondisi) {
    // kode yang akan dijalankan jika kondisi bernilai true
}

Penulisan Sintaks Else
if (kondisi) {
    // kode yang akan dijalankan jika kondisi bernilai true
} else {
    // kode yang akan dijalankan jika kondisi bernilai false
}
```

### Contoh Penggunaan If

Kamu bisa menggunakan `if` untuk membuat keputusan sederhana seperti mengecek nilai dari variabel.

```
$umur = 20;

if ($umur >= 17) {
    echo "Kamu sudah cukup umur.";
}
```

Pada contoh di atas, PHP akan menampilkan “Kamu sudah cukup umur.” jika variabel `$umur` lebih dari atau sama dengan 17.

### Contoh Penggunaan If dan Else

Kamu juga bisa menambahkan `else` untuk menangani situasi ketika kondisi `if` tidak terpenuhi.

```
$umur = 16;

if ($umur >= 17) {
    echo "Kamu sudah cukup umur.";
} else {
```



```
    echo "Kamu belum cukup umur.";
}
```

Dalam contoh ini, PHP akan menampilkan “Kamu belum cukup umur.” karena variabel \$umur kurang dari 17.

### **Kombinasi Dengan ElseIf**

Kadang-kadang, kamu mungkin ingin mengecek lebih dari dua kondisi. Untuk ini, PHP menyediakan elseif.

```
$suhu = 30;

if ($suhu < 20) {
    echo "Cuaca dingin.";
} elseif ($suhu <= 30) {
    echo "Cuaca hangat.";
} else {
    echo "Cuaca panas.";
}
```

Dalam contoh di atas, akan ditampilkan “Cuaca hangat.” karena nilai \$suhu adalah 30, yang memenuhi kondisi kedua.

### **Penutup**

Menggunakan if dan else dalam PHP adalah cara dasar untuk melakukan pengambilan keputusan dalam kode. Melalui contoh-contoh di atas, kamu dapat memahami cara kerja struktur kontrol ini dan mulai mengimplementasikannya dalam kode PHPmu. Ingat, praktik membuat sempurna, jadi terus berlatih dan eksperimen dengan kondisi yang berbeda-beda untuk meningkatkan keahlian pemrogramanmu.

## ELSEIF

Pelajari penggunaan kondisional elseif dalam PHP untuk membuat keputusan dalam kode dengan panduan langkah demi langkah yang mudah diikuti.

Dalam pemrograman, sering kali kamu memerlukan struktur kontrol untuk mengeksekusi kode berbeda berdasarkan kondisi tertentu. PHP menyediakan kondisional elseif untuk menangani situasi ini. Penggunaan elseif memungkinkan pengecekan kondisi tambahan setelah if, dan sebelum else.

### Pengertian Elseif

Kondisional elseif merupakan gabungan dari if dan else. Kode dalam blok elseif akan dijalankan hanya jika kondisi if sebelumnya gagal (yaitu menghasilkan false) dan kondisi elseif-nya sendiri berhasil (menghasilkan true).

### Sintaks Elseif

Sintaks dasar dari elseif dalam PHP adalah sebagai berikut:

```
if (kondisi1) {  
    // kode dieksekusi jika kondisi1 benar  
} elseif (kondisi2) {  
    // kode dieksekusi jika kondisi1 salah dan kondisi2 benar  
} else {  
    // kode dieksekusi jika kondisi1 dan kondisi2 salah  
}
```

Kamu bisa memiliki banyak blok elseif sesuai dengan kebutuhan.

### Contoh Penggunaan Elseif

Misalnya, kamu ingin menampilkan pesan yang berbeda berdasarkan usia pengguna:

```
$usia = 20;  
  
if ($usia < 13) {  
    echo "Kamu masih anak-anak.";   
} elseif ($usia >= 13 && $usia <= 19) {  
    echo "Kamu adalah remaja.";   
} elseif ($usia > 19 && $usia <= 30) {  
    echo "Kamu adalah dewasa muda.";   
} else {  
    echo "Kamu adalah dewasa.";   
}
```

Pada contoh di atas, PHP akan mengecek setiap kondisi satu per satu. Jika \$usia kurang dari 13, pesan “Kamu masih anak-anak.” akan ditampilkan. Jika tidak, ia akan mengecek kondisi berikutnya, dan seterusnya, sampai menemukan kondisi yang benar atau menjalankan blok else jika tidak ada satupun kondisi elseif yang benar.

### Tips Penggunaan Elseif

Pastikan untuk tidak melewatkan tanda ( dan ) pada kondisi.

Gunakan operasi perbandingan yang tepat, seperti ==, !=, >, <, >=, <=.

Kondisi yang lebih spesifik biasanya ditempatkan di awal, dan yang lebih umum di akhir.

Hindari terlalu banyak elseif karena dapat membuat kode sulit dibaca. Pertimbangkan untuk menggunakan switch atau refaktor kodemu.

Dengan memahami cara kerja elseif, kamu bisa membuat kode PHP-mu lebih efisien dan mudah dibaca. Gunakanlah sesuai kebutuhan dan jangan ragu untuk bereksperimen dalam menulis kondisional.

## SWITCH CASE

Pelajari penggunaan switch case dalam PHP untuk menangani berbagai kondisi dengan mudah dan efisien. Contoh kode dan penjelasan yang sederhana akan memudahkan pemahaman.

PHP menyediakan berbagai struktur kontrol untuk membuat keputusan dalam kode. Salah satunya adalah switch case, yang merupakan alternatif dari penggunaan beruntun if-else yang bisa menjadi rumit dan sulit dibaca. Berikut kami jelaskan bagaimana menggunakan switch case di PHP.

### Menggunakan Switch Case

switch case memungkinkan kamu untuk membandingkan satu nilai terhadap banyak nilai atau kasus. Sintaks dasarnya adalah sebagai berikut:

```
switch ($variable) {  
    case 'value1':  
        // kode yang akan dijalankan jika $variable == 'value1'  
        break;  
    case 'value2':  
        // kode yang akan dijalankan jika $variable == 'value2'  
        break;  
    // Tambahkan lebih banyak case sesuai kebutuhan  
    default:  
        // kode yang akan dijalankan jika tidak ada case yang cocok  
}
```

### Break

Perintah break penting untuk menghentikan eksekusi lebih lanjut dari blok switch setelah menemukan case yang cocok. Tanpa break, PHP akan terus mengeksekusi kasus-kasus berikutnya meski sudah menemukan yang cocok.

### Default

Bagian default adalah opsional dan dijalankan apabila tidak ada case yang sesuai dengan nilai variabel yang diberikan. Anggap saja sebagai “else” pada if-else.

### Contoh Penggunaan

Sebagai contoh penggunaan praktis, bayangkan kamu memiliki aplikasi yang membutuhkan sistem grading:

```
$nilai = 85;  
  
switch (true) {  
    case ($nilai >= 90):  
        echo "Grade A";  
        break;  
    case ($nilai >= 80):  
        echo "Grade B";  
        break;  
    case ($nilai >= 70):  
        echo "Grade C";  
        break;  
    case ($nilai >= 60):  
        echo "Grade D";  
        break;  
    default:  
        echo "Grade E";  
}
```

Dalam contoh ini, switch case digunakan untuk memeriksa rentang nilai dan menampilkan grade sesuai dengan nilai yang didapat.

### Kesalahpahaman Umum

Salah satu kesalahpahaman tentang switch adalah bahwa ia hanya dapat menangani perbandingan sama dengan (`=`). Sebenarnya, kamu bisa menggunakan ekspresi untuk setiap case, seperti yang terlihat pada contoh grading di atas, dengan menempatkan `true` sebagai ekspresi dalam switch.

Dengan mengetahui cara kerja switch case di PHP, kamu bisa menulis kode yang lebih bersih dan efisien untuk menggantikan statemen if-else yang berlebihan.