

ALAPORAN PROYEK 5

FUNCTION & METHOD FORM



OLEH:
AHMAD ARJUN TRISULA
(NISN. 0082311714)

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



MATERI PERTEMUAN 10

Memahami Prosedur dan Fungsi



TAHUKAH KAMU...?

Pada kesempatan ini, kita akan membahas:

- Fungsi dengan Parameter
- Parameter dengan Nilai Default
- Fungsi yang Mengembalikan Nilai
- Memanggil Fungsi di dalam Fungsi

A. Menulis / Membuat Fungsi Pada PHP

Fungsi adalah sekumpulan intruksi yang dibungkus dalam sebuah blok. Fungsi dapat digunakan ulang tanpa harus menulis ulang instruksi di dalamnya. Fungsi pada PHP dapat dibuat dengan kata kunci **function**, lalu diikuti dengan nama fungsinya.

Membuat fungsi pada PHP dapat dilakukan dengan mudah, yaitu (1) menuliskan keyword **function** (2) kemudian diikuti dengan nama fungsi (3) diikuti dengan tanda kurung **()** sebagai tempat argumen, (4) kemudian diikuti dengan kurung kurawal **{ }** sebagai block statement yang akan dijalankan ketika fungsi dipanggil.

Function Name Arguments

```
function cetak ( $text, $callback ) {  
    echo $text;  
}
```

Statement

Pada kondisi tertentu nama fungsi ini tidak ditulis, lihat bagian V. Anonymous function atau closure

Penulisan nama fungsi harus mengikuti ketentuan sebagai berikut:

1. Harus diawali huruf atau underscore(_) kemudian dapat diikuti dengan huruf, angka, dan underscore
2. Case in-sensitive (**tidak** membedakan huruf kecil dan besar)

Banyak fungsi *build-in* dari php yang sering kita gunakan, seperti **print()**, **print_r()**, **unset()**, dll. Selain fungsi-fungsi tersebut, kita juga dapat membuat fungsi sendiri sesuai kebutuhan.

Contoh:

```
function namaFungsi(){  
    //...  
}
```

Kode intruksi dapat di tulis di dalam kurung kurawal (**{...}**).

Contoh:

```
function perkenalan(){  
    echo "Assalamulaikmu, ";  
    echo "Perkenalkan, nama kita Ahmadi<br/>";  
    echo "Senang berkenalan dengan anda<br/>";  
}
```

Fungsi yang sudah dibuat tidak akan menghasilkan apapun kalau tidak dipanggil. Kita dapat memanggil fungsi dengan menuliskan namanya.

Contoh:

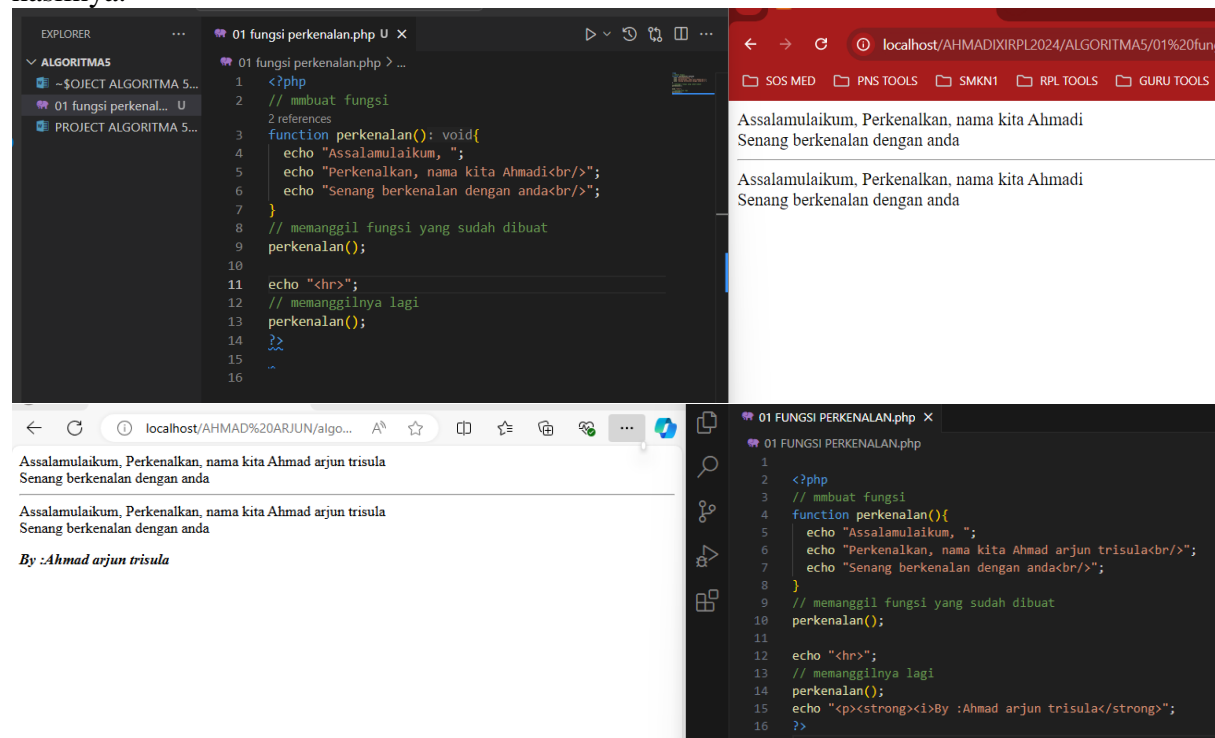
```
perkenalan();
```

Jadi, kode lengkapnya seperti ini:

```
<?php
// mmbuat fungsi
function perkenalan(){
    echo "Assalamualaikum, ";
    echo "Perkenalkan, nama kita Ahmadi<br/>";
    echo "Senang berkenalan dengan anda<br/>";
}
// memanggil fungsi yang sudah dibuat
perkenalan();

echo "<hr>";
// memanggilnya lagi
perkenalan();
?>
```

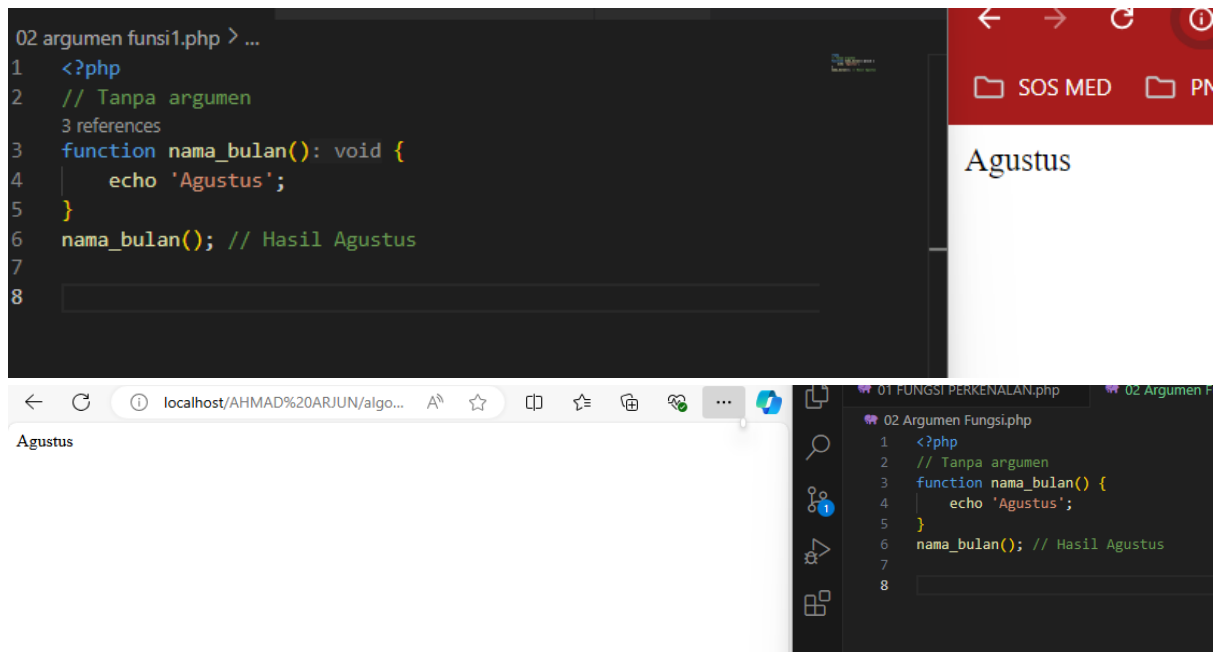
hasilnya:



1. Argumen Fungsi

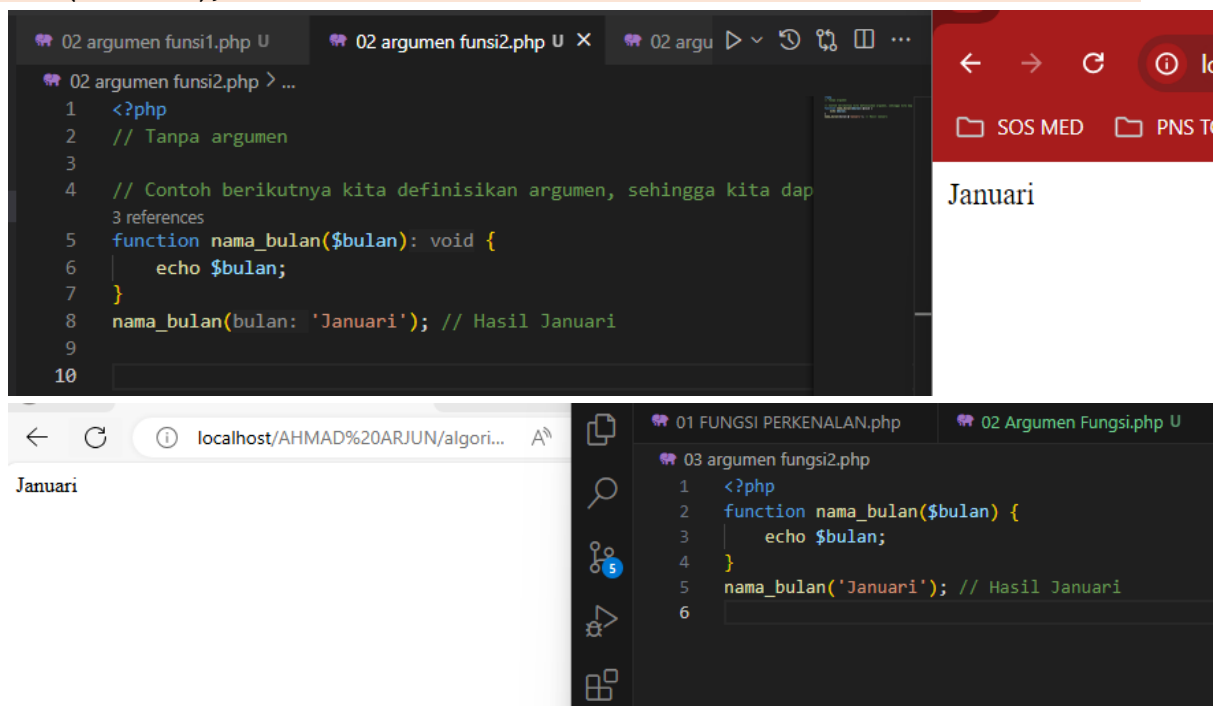
Argumen fungsi ditulis dalam tanda kurung dan dapat berupa tipe data apapun baik string, array, object, boolean, dsb..., selain itu argumen juga dapat dikosongkan, contoh:

```
<?php
// Tanpa argumen
function nama_bulan() {
    echo 'Agustus';
}
nama_bulan(); // Hasil Agustus
```



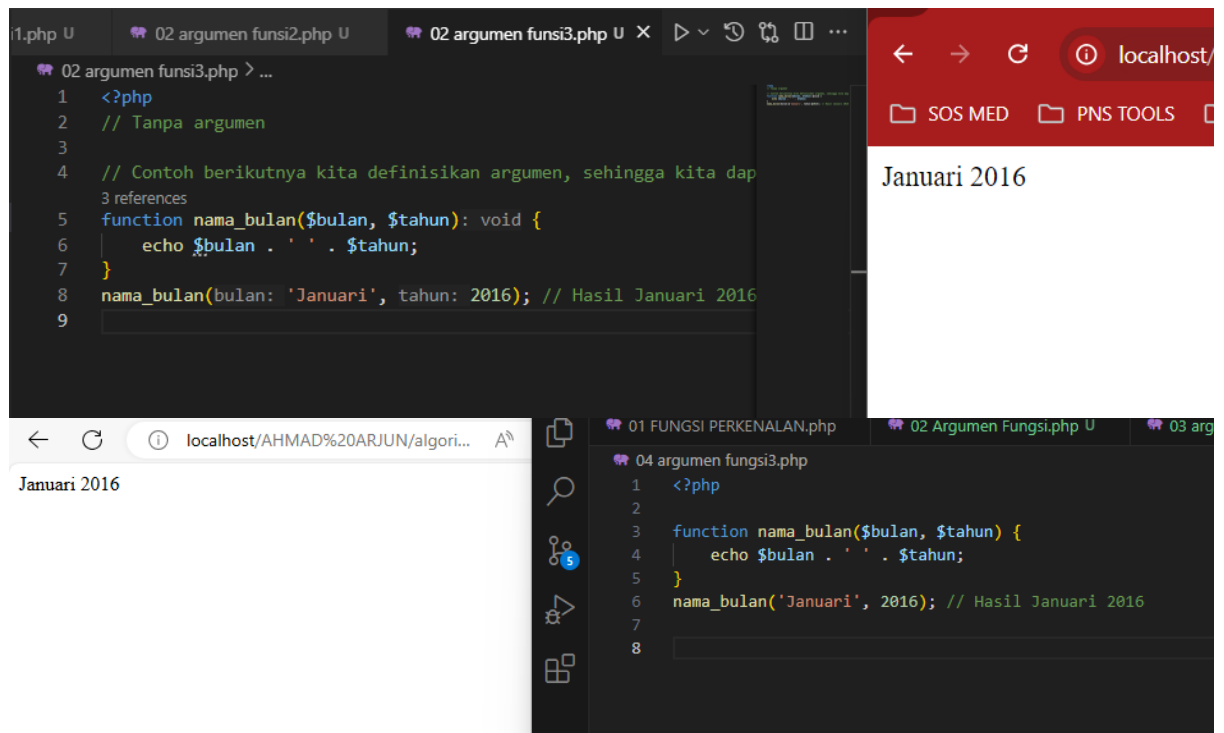
Contoh berikutnya kita definisikan argumen, sehingga kita dapat mencetak nama bulan sesuai dengan yang kita inginkan:

```
function nama_bulan($bulan) {
    echo $bulan;
}
nama_bulan('Januari'); // Hasil Januari
```



Lebih lanjut, argumen dari fungsi ini dapat kita definisikan lebih dari satu, caranya, pisahkan argumen dengan tanda koma, contoh:

```
function nama_bulan($bulan, $tahun) {
    echo $bulan . ' ' . $tahun;
}
nama_bulan('Januari', 2016); // Hasil Januari 2016
```

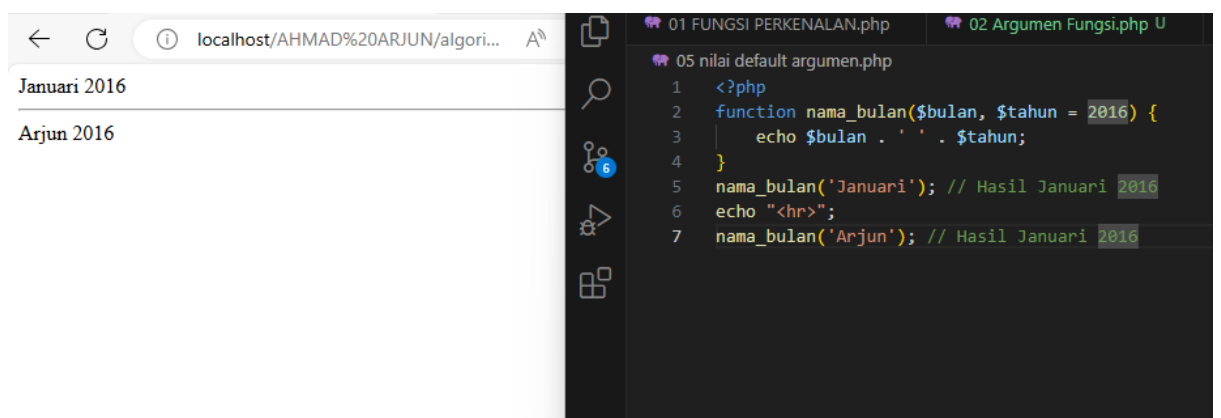
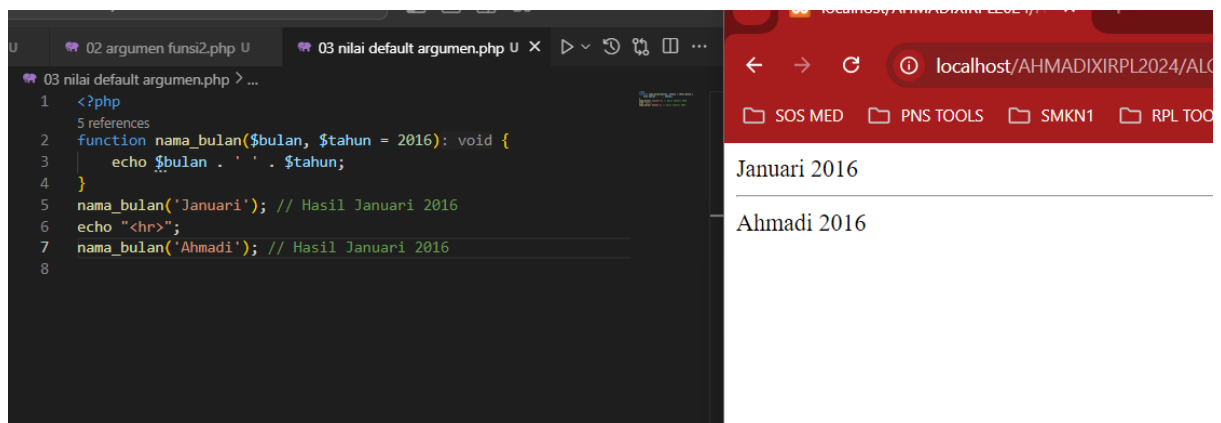


2. Nilai Default Argumen

Kita dapat mendefinisikan nilai default dari argumen, sehingga memudahkan pemanggilan fungsi karena tidak perlu menulis argumen terlalu banyak, contoh:

```
function nama_bulan($bulan, $tahun = 2016) {  
    echo $bulan . ' ' . $tahun;  
}  
nama_bulan('Januari'); // Hasil Januari 2016
```

Nilai default argumen ini bisa kita isi tipe data apa saja seperti boolean (true, false), null, array, object, dll



3. Memanggil Fungsi

Pemanggilan fungsi dilakukan dengan menulis nama fungsi tersebut, seperti pada contoh sebelumnya, kita memanggil fungsi dengan menulis `nama_bulan()` dan `nama_bulan('Januari', 2016)`. Jika fungsi memerlukan argumen, maka kita juga harus menulis argumen tersebut.

Catatan: Ketika melakukan pemanggilan fungsi, maka penulisan argumen harus lengkap, jika fungsi terdiri dari 3 argumen, maka kita harus menuliskan ketiganya, jika tidak maka akan muncul pesan error, KECUALI argumen tersebut memiliki nilai default (dibahas dibawah)

B. Nilai Kembalian – Return Value

1. Menggunakan return

Nilai kembalian ini maksudnya fungsi yang kita panggil tadi akan menghasilkan nilai tertentu, nilai tersebut bisa bertipe apa saja seperti: boolean, float, array, object, dll

Nilai kembalian ini dijalankan dengan menggunakan keyword `return`, contoh:

```
<?php
function nama_bulan($bulan) {
    $nama_bulan = array (1 => 'Januari', 2 => 'Februari', 3 => 'Maret');
    return $nama_bulan[$bulan];
}
// date('n') akan menghasilkan bulan sekarang dalam bentuk 1 digit, misal 3 untuk Januari
$bulan = nama_bulan(date('n')); // Hasil Maret
echo $bulan . ' ' . date('Y'); // Hasil Maret 2016
```

Keyword ini dapat diletakkan dimana saja di dalam fungsi dan ketika php menemukan keyword ini, maka seketika pemanggilan fungsi akan dihentikan dan PHP kembali ke baris dimana fungsi tadi dipanggil.

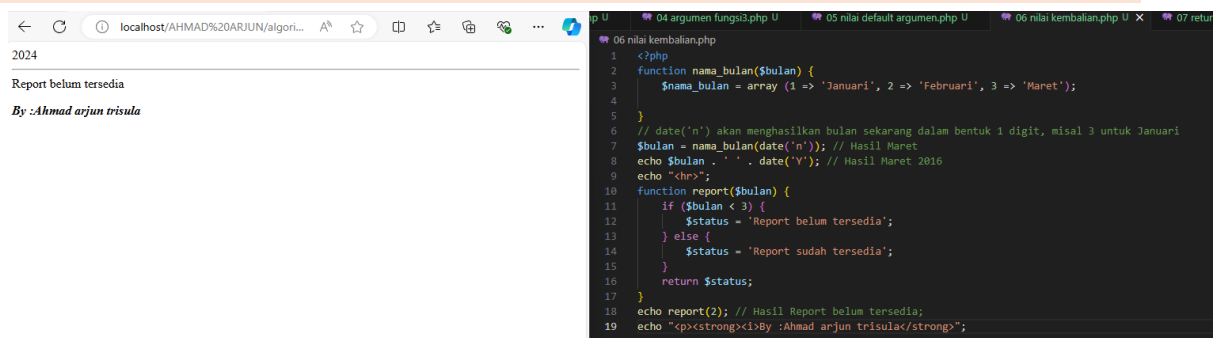
Karakteristik tersebut dapat memudahkan kita mengatur penulisan kode, sehingga, ketika menulis fungsi, kita harus selalu mempertimbangkan kemungkinan penggunaan `return` di tengah code, terutama ketika menggunakan conditional `if`

Contoh:

```
function report($bulan) {
    if ($bulan < 3) {
        $status = 'Report belum tersedia';
    } else {
        $status = 'Report sudah tersedia';
    }
    return $status;
}
echo report(2); // Hasil Report belum tersedia;
```

Untuk lebih efisien, kode tersebut dapat diubah menjadi:

```
function report($bulan) {
    if ($bulan < 3) {
        return 'Report belum tersedia';
    } else {
        return 'Report sudah tersedia';
    }
}
```



Pada script pertama, PHP akan membaca seluruh kode pada fungsi, sebaliknya untuk script kedua, ketika sampai `if` maka fungsi berhenti karena bertemu `return`,

Hal ini tentu akan mempercepat proses eksekusi terlebih jika script yang kita tulis panjang.

2. Return value lebih dari satu nilai

Return value HANYA memberikan nilai kembalian sebanyak satu nilai, misal pada contoh diatas hanya menghasilkan nama bulan, jika ingin menghasilkan nilai kembalian lebih dari satu, maka kita gunakan array, contoh:

```
<?php
function nama_bulan($bulan) {
    $nama_bulan = array (1 => 'Januari', 2 => 'Februari', 3 => 'Maret');
    $semester    = $bulan < 7 ? 1 : 2;
    return array('bulan' => $nama_bulan[$bulan], 'semester' => $semester);
}
$bulan = nama_bulan(3);
echo '<pre>'; print_r($bulan);
/* HASIL:
Array
(
    [bulan] => Maret
    [semester] => 1
) */
```

Hasil pengolahan nilai dari fungsi mungkin saja kita butuhkan untuk pemrosesan berikutnya. Oleh karena itu, kita harus membuat fungsi yang dapat mengembalikan nilai.

Pengembalian nilai dalam fungsi dapat menggunakan kata kunci **return**.

Contoh:

```
<?php
// membuat fungsi
function hitungUmur($thn_lahir, $thn_sekarang){
    $umur = $thn_sekarang - $thn_lahir;
    return $umur;
}

echo "Umur kita adalah ". hitungUmur(1986, 2023) ." tahun";
?>
```

Hasilnya:

The screenshot shows a web browser window at the top with the address bar displaying 'localhost/post/nilai-kembalian.php'. The page content shows 'Umur saya adalah 21 tahun'. Below the browser window, there is a code editor with two files open. The left file, '07 Return value lebih dari satu nilai.php', contains the PHP code for the 'nama_bulan' function and its usage, which outputs an array. The right file, '05 nilai default argumen.php', contains the PHP code for the 'hitungUmur' function and its usage, which outputs the age calculation. The browser's developer tools or a separate window below shows the output of the first script as an array: 'Array ([bulan] => Maret [semester] => 1)' and the output of the second script as 'Umur kita adalah 16 tahun'.

C. Fungsi dengan Parameter

Supaya intruksi yang di dalam fungsi lebih dinamis, kita dapat menggunakan parameter untuk memasukkan sebuah nilai ke dalam fungsi. Nilai tersebut akan diolah di dalam fungsi.

Misalkan, pada contoh fungsi yang tadi, tidak mungkin nama yang dicetak adalah *Ahmadi* saja dan salam yang dipakai tidak selalu *assalamualaikum*. Maka, kita dapat menambahkan parameter menjadi seperti ini:

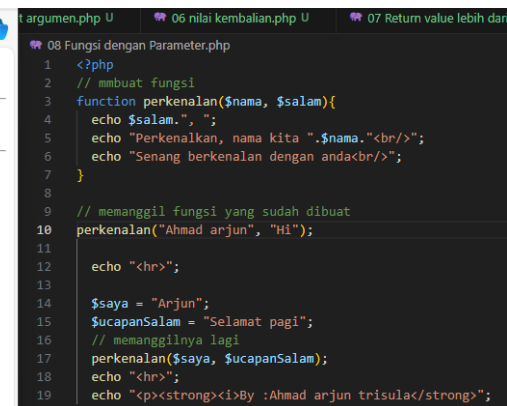
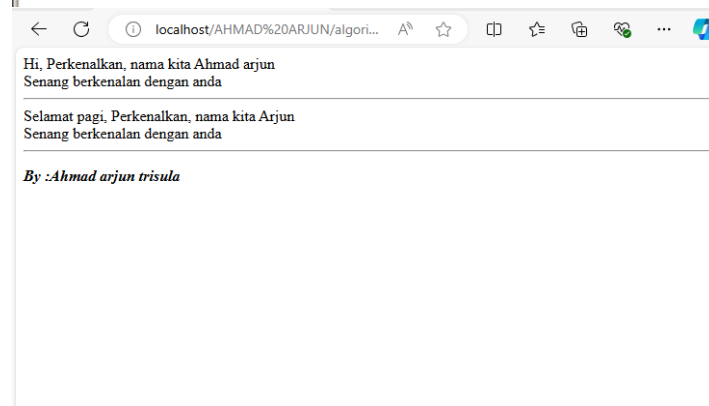
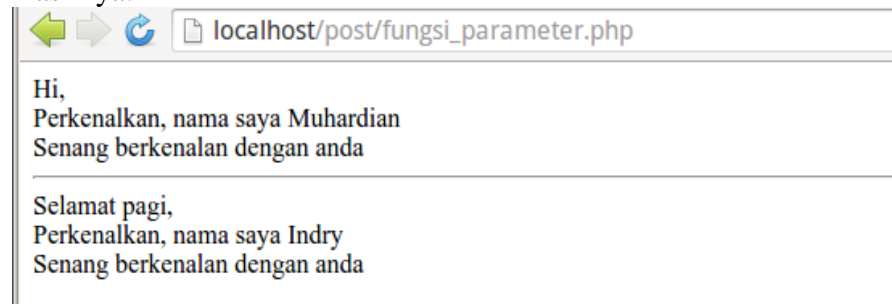
```
<?php
// mmbuat fungsi
function perkenalan($nama, $salam){
    echo $salam.", ";
    echo "Perkenalkan, nama kita ".$nama."<br/>";
    echo "Senang berkenalan dengan anda<br/>";
}

// memanggil fungsi yang sudah dibuat
perkenalan("Muhardian", "Hi");

echo "<hr>";

$saya = "Ahmadi";
$ucapanSalam = "Selamat pagi";
// memanggilnya lagi
perkenalan($saya, $ucapanSalam);
?>
```

Hasilnya:



Parameter dengan Nilai Default

Nilai *default* dapat kita berikan di parameter. Nilai *default* berfungsi untuk mengisi nilai sebuah parameter, kalau parameter tersebut tidak diisi nilainya.

Misalnya: kita lupa mengisi parameter *salam*, maka program akan *error*. Oleh karena itu, kita perlu memberikan nilai *default* supaya tidak *error*.

Contoh:

```
<?php
// mmbuat fungsi
function perkenalan($nama, $salam="Assalamualaikum"){
    echo $salam.", ";
    echo "Perkenalkan, nama kita ".$nama."<br/>";
    echo "Senang berkenalan dengan anda<br/>";
}
```

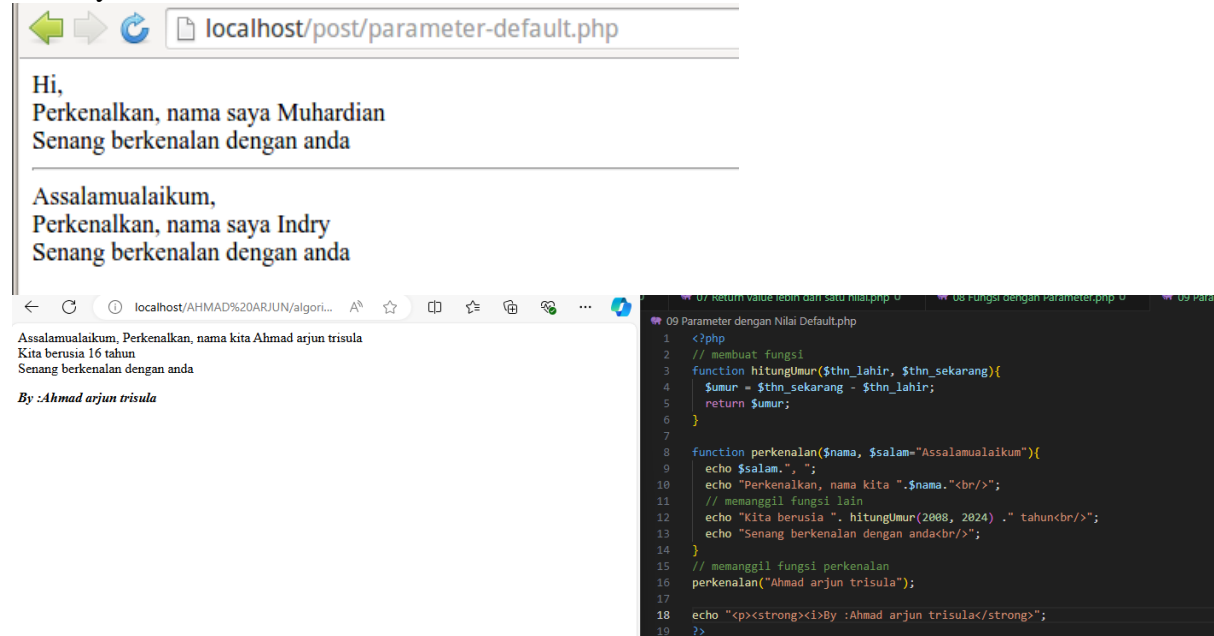


```
// memanggil fungsi yang sudah dibuat
perkenalan("Muhardian", "Hi");

echo "<hr>";

$kita = "Indry";
$ucapanSalam = "Selamat pagi";
// memanggilnya lagi tanpa mengisi parameter salam
perkenalan($saya);
?>
```

Hasilnya:



D. Memanggil Fungsi di dalam Fungsi

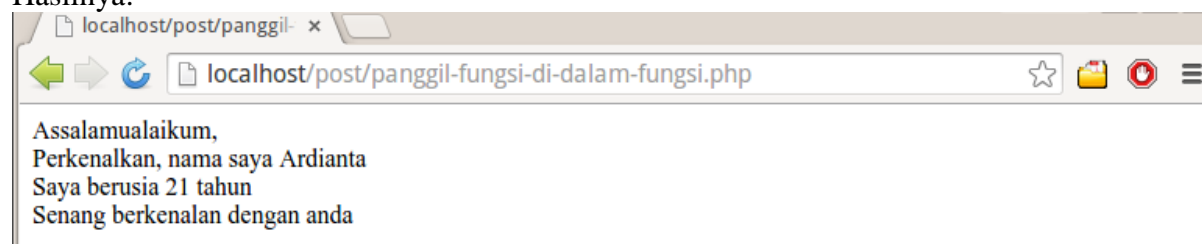
Fungsi yang sudah kita buat, dapat juga dipanggil di dalam fungsi lain.

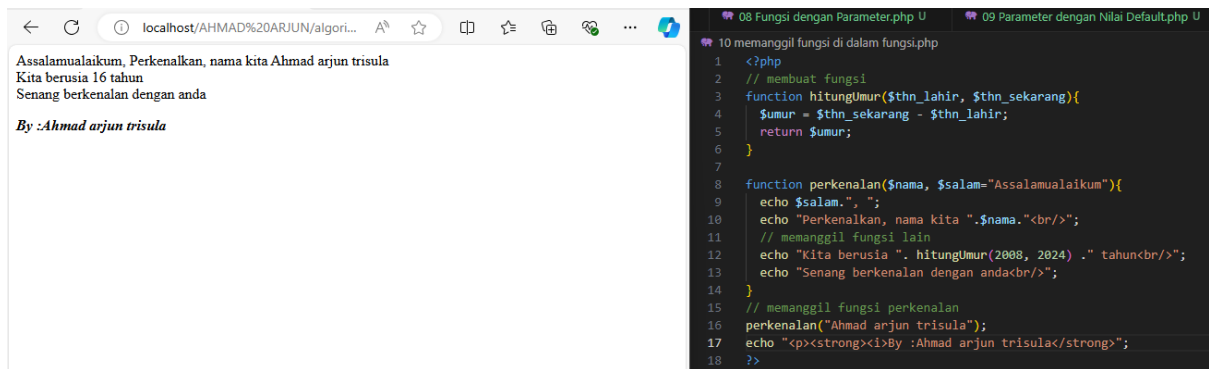
Contoh:

```
<?php
// membuat fungsi
function hitungUmur($thn_lahir, $thn_sekarang){
    $umur = $thn_sekarang - $thn_lahir;
    return $umur;
}

function perkenalan($nama, $salam="Assalamualaikum"){
    echo $salam.", ";
    echo "Perkenalkan, nama kita ".$nama."<br/>";
    // memanggil fungsi lain
    echo "Kita berusia ". hitungUmur(1986, 2023) ." tahun<br/>";
    echo "Senang berkenalan dengan anda<br/>";
}
// memanggil fungsi perkenalan
perkenalan("Ahmadi");
?>
```

Hasilnya:





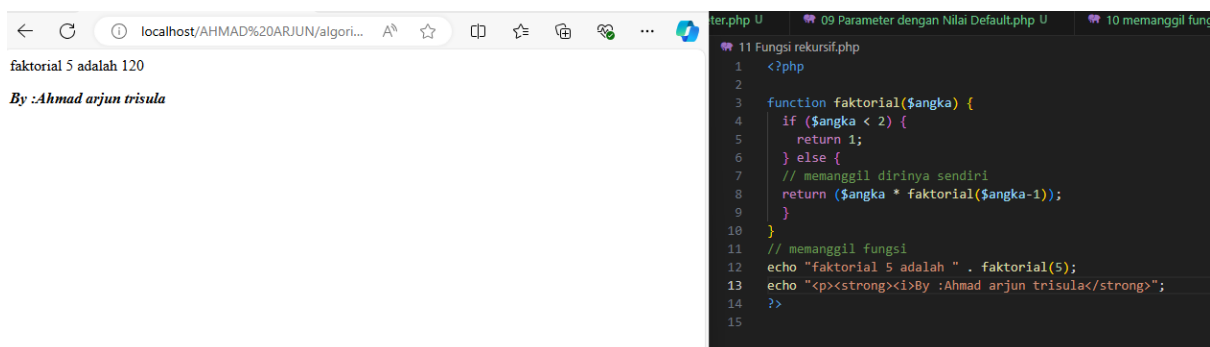
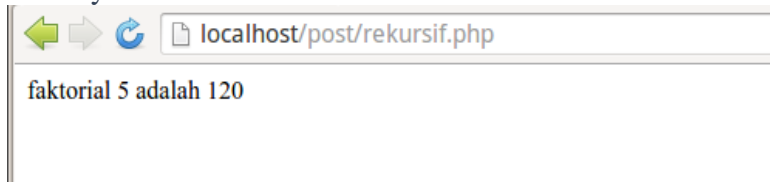
E. Fungsi rekursif

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri. Fungsi ini biasanya digunakan untuk menyelesaikan masalah seperti faktorial, bilangan fibonacci, pemrograman dinamis, dll. Contoh fungsi rekursif:

```
<?php

function faktorial($angka) {
    if ($angka < 2) {
        return 1;
    } else {
        // memanggil dirinya sendiri
        return ($angka * faktorial($angka-1));
    }
}
// memanggil fungsi
echo "faktorial 5 adalah " . faktorial(5);
?>
```

Hasilnya:



F. Jenis Fungsi Pada PHP

Fungsi pada PHP dibagi menjadi dua yaitu *built-in function* yang merupakan fungsi bawaan PHP dan *user-defined function*, dimana kita membuat fungsi sendiri.

Banyak sekali fungsi yang disediakan php, seperti substr(), dll yang dapat kita gunakan langsung, list lengkapnya dapat dilihat pada halaman: PHP: Function and Method Listing – Manual. Terkait fungsi ini ada dua hal pokok yang dapat kita lakukan, yaitu:

1. Mendefinisikan fungsi sendiri, jika fungsi yang kita inginkan belum disediakan oleh PHP

2. Memanggil fungsi, baik *built-in function* dan *user-defined function* cara memanggilnya sama yaitu menuliskan nama fungsi kemudian diikuti tanda kurung, misal kita memanggil fungsi cetak, maka kita menuliskan: cetak()

Kita tidak perlu menghafal semua fungsi tersebut dan ketika memerlukan tidak perlu mencarinya disana, karena tetap akan sulit mencarinya, sebaliknya, gunakan saja google.

G. Fungsi Alias Pada PHP

Untuk alasan tertentu seperti penamaan fungsi yang lebih relevan dengan tugas fungsi tersebut, PHP menyediakan fungsi baru yang tugasnya sama persis dengan fungsi lama, yang disebut fungsi alias (Function Aliases)

Fungsi baru tersebut tidak memiliki kode sendiri, melainkan ketika dipanggil, dia memanggil fungsi yang lama. Contoh dari fungsi alias ini adalah: die(), key_exists() dan join() yang merupakan alias dari fungsi exit(), array_key_exists() dan implode(), contoh penggunaan fungsi ini dapat dibaca disini.

Dari contoh tersebut terlihat bahwa die(), key_exists dan join lebih pendek dan lebih memiliki arti, untuk list lengkap dari alias ini, dapat dilihat pada halaman: PHP: List of Function Aliases – Manual. Beberapa orang mempermasalahkan performa dari fungsi alias ini, karena fungsinya yang menjalankan fungsi lain, namun sebenarnya tidak masalah menggunakan fungsi ini, karena perbedaan speed nya sangat tidak signifikan.

H. Anonymous Function

Anonymous function atau disebut juga *closure* dapat diartikan fungsi tanpa nama (anonymous). Fungsi ini umumnya digunakan pada fungsi-fungsi yang membutuhkan *callback* (fungsi yang dipanggil oleh fungsi lainnya).

Fungsi yang membutuhkan callback ini bisa *built-in function* seperti preg_replace_callback, array_map, array_walk, dll maupun *user-defined function*

Contoh berikut ini diambil dari tulisan sebelumnya:

```
<?php
$kendaraan = array('Mobil', 'Motor', 'Sepeda');
$upper = array_map('toupper', $kendaraan);
function toupper($array_val) {
    return strtoupper($array_val);
}
echo '<pre>'; print_r($upper);
```

Nah, seperti disampaikan sebelumnya, tujuan dibuatnya function adalah agar dapat digunakan kembali (*re-use*), karena fungsi toupper() HANYA digunakan sekali, maka fungsi tersebut dapat kita gabungkan ke dalam fungsi array_map sehingga bentuknya menjadi anonymous function:

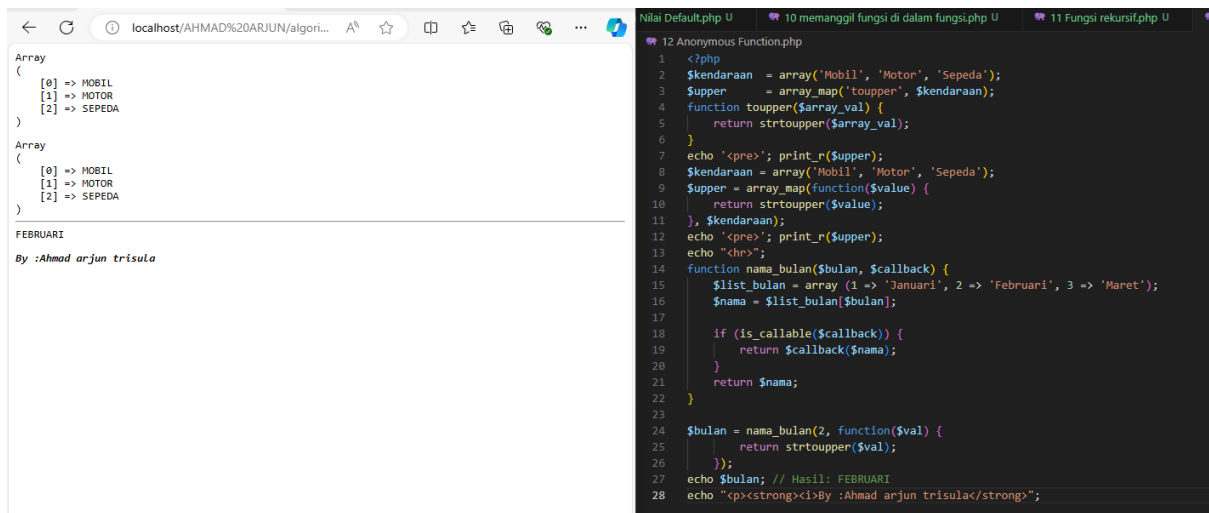
```
<?php
$kendaraan = array('Mobil', 'Motor', 'Sepeda');
$upper = array_map(function($value) {
    return strtoupper($value);
}, $kendaraan);
echo '<pre>'; print_r($upper);
```

Contoh lain pada *user-defined function*:

```
function nama_bulan($bulan, $callback) {
    $list_bulan = array (1 => 'Januari', 2 => 'Februari', 3 => 'Maret');
    $nama = $list_bulan[$bulan];

    if (is_callable($callback)) {
        return $callback($nama);
    }
    return $nama;
}

$bulan = nama_bulan(2, function($val) {
    return strtoupper($val);
});
echo $bulan; // Hasil: FEBRUARI
```



ARGUMEN ATAU PARAMETER FUNGSI

Pelajari cara menggunakan argumen atau parameter dalam fungsi PHP untuk membuat kode yang lebih modular dan fleksibel.

Dalam pemrograman PHP, fungsi berperan vital dalam membantu kita memecah kode menjadi bagian-bagian yang lebih kecil dan termanage. Agar fungsi dapat bekerja dengan berbagai jenis data dan lebih fleksibel, kita menggunakan argumen atau parameter. Mari kita pahami lebih lanjut tentang konsep ini.

Apa itu Argumen Fungsi?

Argumen fungsi adalah nilai yang kamu berikan kepada fungsi ketika kamu memanggilnya. Nilai-nilai ini digunakan oleh fungsi untuk melakukan operasi atau tugas tertentu. Argumen seringkali diperlukan oleh fungsi untuk bisa bekerja dengan benar.

Contoh:

```
function sapa($nama) {  
    echo "Halo, " . $nama . "!";  
}  
  
sapa("Budi");
```

Dalam contoh di atas, "Budi" merupakan argumen dari fungsi sapa.

Mendefinisikan Parameter

Parameter adalah variabel yang digunakan dalam deklarasi fungsi untuk menerima argumen yang diberikan saat fungsi dipanggil.

Contoh mendefinisikan fungsi dengan parameter:

```
function tambah($angka1, $angka2) {  
    return $angka1 + $angka2;  
}  
  
echo tambah(5, 10); // Output: 15
```

Fungsi tambah memiliki dua parameter: \$angka1 dan \$angka2.

Tipe Parameter

PHP mendukung tipe penentuan parameter (type hinting), yang memungkinkan kita mendefinisikan tipe data dari parameter.

Contoh dengan tipe parameter:

```
function setNilai(int $nilai) {
```

```
    echo $nilai;
}
```

```
setNilai(10); // Output: 10
```

Dengan menambahkan int sebelum \$nilai, kita menentukan bahwa parameter tersebut harus berupa integer.

Argumen Default

Kamu bisa menentukan nilai default untuk parameter. Ini sangat berguna saat kamu ingin argumen bersifat opsional.

Contoh argumen default:

```
function salam($waktu = "pagi") {
    echo "Selamat $waktu!";
}
```

```
salam(); // Output: Selamat pagi!
salam("sore"); // Output: Selamat sore!
```

Pada fungsi salam, \$waktu secara default memiliki nilai "pagi".

Passing by Reference

Secara default, argumen dalam PHP di-passing by value. Namun, PHP juga memungkinkan kita untuk pass argumen by reference, yang berarti kamu bisa mengubah nilai asli dari variabel tersebut melalui fungsi.

Contoh passing by reference:

```
function tambahSatu(&$nilai) {
    $nilai++;
}
```

```
$angka = 10;
tambahSatu($angka);
echo $angka; // Output: 11
```

Perhatikan simbol & sebelum \$nilai; ini menandakan bahwa argumen di-pass by reference.

Jumlah Argumen Dinamis

Dalam PHP, fungsi dapat menerima jumlah argumen yang variatif dengan menggunakan fungsi bawaan seperti func_get_args dan func_num_args.

Contoh jumlah argumen dinamis:

```
function jumlahkan() {
    $total = 0;
    $args = func_get_args();

    foreach ($args as $arg) {
        $total += $arg;
    }

    return $total;
}
```

```
echo jumlahkan(1, 2, 3, 4); // Output: 10
```

Fungsi jumlahkan di atas bisa mengambil jumlah argumen yang tidak terbatas dan menjumlahkannya.

Memahami penggunaan argumen dan parameter sangat penting dalam pemrograman PHP. Dengan menggunakan argumen dan parameter dengan baik, kamu bisa membuat kode yang lebih bersih, modular, dan mudah untuk dipahami.

← ↻ ⓘ localhost/AHMAD%20ARJUN/algori... A

Halo, Budi!

15

10

Selamat pagi!Selamat sore!

11

10

By :Ahmad arjun trisula

13 ARGUMEN ATAU PARAMETER FUNGSI.php

```
4 }
5
6 sapa("Budi");
7 echo "<hr>";
8 function tambah($angka1, $angka2) {
9     return $angka1 + $angka2;
10 }
11
12 echo tambah(5, 10); // Output: 15
13 echo "<hr>";
14 function setNilai(int $nilai) {
15     echo $nilai;
16 }
17
18 setNilai(10); // Output: 10
19 echo "<hr>";
20 function salam($waktu = "pagi") {
21     echo "Selamat $waktu!";
22 }
23
24 salam(); // Output: Selamat pagi!
25 salam("sore"); // Output: Selamat sore!
26 echo "<hr>";
27 function tambahSatu(&$nilai) {
28     $nilai++;
29 }
30
31 $angka = 10;
32 tambahSatu($angka);
33 echo $angka; // Output: 11
34 echo "<hr>";
35
36 function jumlahkan() {
37     $total = 0;
38     $args = func_get_args();
39
40     foreach ($args as $arg) {
41         $total += $arg;
42     }
43
44     return $total;
45 }
46
47 echo jumlahkan(1, 2, 3, 4); // Output: 10
48 echo "<hr>";
49 echo "<p><strong><i>By :Ahmad arjun trisula</i></strong>";
```

DEFAULT ARGUMEN

Pelajari cara menggunakan argumen default dalam fungsi PHP dengan penjelasan sederhana dan contoh kode yang mudah dipahami.

Menggunakan argumen default dalam fungsi PHP memudahkan kita dalam pemrograman karena memungkinkan fungsi untuk memiliki nilai parameter yang telah ditentukan sebelumnya. Ini berguna ketika kamu ingin fungsi dapat dipanggil dengan jumlah argumen yang tidak tetap. Dalam panduan singkat ini, kita akan melihat cara menentukan dan menggunakan argumen default dalam fungsi PHP.

Mendefinisikan Argumen Default

Untuk menetapkan nilai default untuk parameter fungsi, kamu perlu mendefinisikannya dalam deklarasi fungsi. Nilai default tersebut akan digunakan jika ketika fungsi dipanggil, argumen untuk parameter tersebut tidak disertakan.

Sintaks Argumen Default

Berikut ini adalah sintaks dasar dalam mendefinisikan argumen default dalam fungsi PHP:

```
function namaFungsi($parameter = 'nilaiDefault') {  
    // kode fungsi di sini  
}
```

Contoh penggunaan Argumen Default

Misalnya, kita memiliki sebuah fungsi untuk menyapa seseorang, tetapi jika tidak ada nama yang diberikan, fungsi akan memberi sapaan umum “Halo, pengunjung!”

```
function sapa($nama = 'pengunjung') {  
    echo "Halo, $nama!";  
}  
  
sapa(); // Output: Halo, pengunjung!  
sapa('Dewi'); // Output: Halo, Dewi!
```

Memanggil Fungsi Dengan Argumen Default

Saat memanggil fungsi yang memiliki argumen default, kamu bisa melewati argumen tersebut jika ingin menggunakan nilai yang telah ditetapkan sebelumnya.

Memanggil tanpa Argumen

Jika kita tidak menyertakan argumen saat memanggil fungsi, nilai default akan diterapkan.

```
sapa(); // menggunakan nilai default yang didefinisikan dalam fungsi
```

Memanggil dengan Argumen

Kamu juga bisa menyertakan argumen baru jika ingin menggantikan nilai default.

```
sapa('Budi'); // menggunakan argumen yang diberikan, "Budi"
```

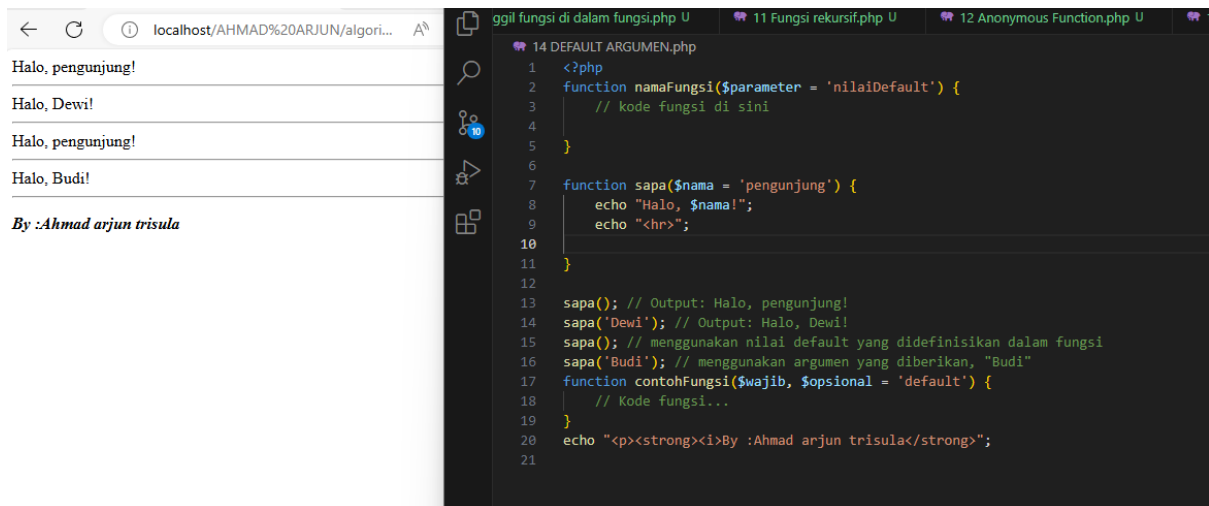
Aturan dalam Menggunakan Argumen Default

Ada aturan penting yang harus diingat saat menggunakan argumen default dalam fungsi PHP:

Argumen Non-default Setelah Default

Jika kamu memiliki beberapa parameter, dan beberapa di antaranya adalah argumen default, pastikan untuk menempatkan argumen default setelah semua parameter non-default.

```
function contohFungsi($wajib, $opsional = 'default') {  
    // Kode fungsi...  
}
```



Dengan mengikuti panduan dasar ini, kamu sekarang dapat menggunakan argumen default dalam fungsi PHP untuk membuat kode yang lebih fleksibel dan mudah dipelihara. Ingatlah untuk menempatkan argumen default dengan benar dalam deklarasi fungsi untuk menghindari kesalahan saat pemanggilan fungsi. Selamat mencoba!

RETURN FUNCTION

Pelajari penggunaan fungsi return dalam PHP untuk mengembalikan nilai dari fungsi dengan contoh kode yang mudah diikuti.

Dalam pemrograman dengan PHP, fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu dan dapat digunakan berkali-kali dalam program. Salah satu aspek penting dalam menggunakan fungsi adalah memahami cara kerja pengembalian nilai menggunakan return. Keyword return digunakan untuk mengakhiri eksekusi fungsi dan secara opsional mengembalikan nilai ke lokasi pemanggilan fungsi. Mari kita bahas cara kerja return dalam fungsi PHP.

Pengertian Fungsi Return

Fungsi return sangat penting dalam pemrograman karena memberikan keluaran dari sebuah fungsi. Ketika return dieksekusi, fungsi akan selesai dan mengembalikan nilai yang ditentukan ke baris kode yang memanggil fungsi tersebut. Nilai yang dikembalikan ini bisa berupa tipe data apapun seperti string, integer, array, atau bahkan objek.

Menggunakan Return dalam Fungsi

Untuk menggunakan return dalam fungsi, kamu cukup menambahkan keyword return diikuti dengan nilai yang ingin dikembalikan. Berikut ini contoh penggunaan return dalam sebuah fungsi sederhana:

```
function jumlahkan($a, $b) {  
    $hasil = $a + $b;  
    return $hasil;  
}  
  
echo jumlahkan(5, 10); // Output: 15
```

Pada contoh di atas, fungsi jumlahkan mengembalikan nilai \$hasil yang merupakan penjumlahan dari dua parameter \$a dan \$b.

Penggunaan Return untuk Mengakhiri Fungsi

Selain mengembalikan nilai, return juga dapat digunakan untuk segera mengakhiri eksekusi fungsi tanpa mengembalikan nilai apapun. Hal ini berguna ketika kamu ingin keluar dari fungsi karena sebuah kondisi tertentu.

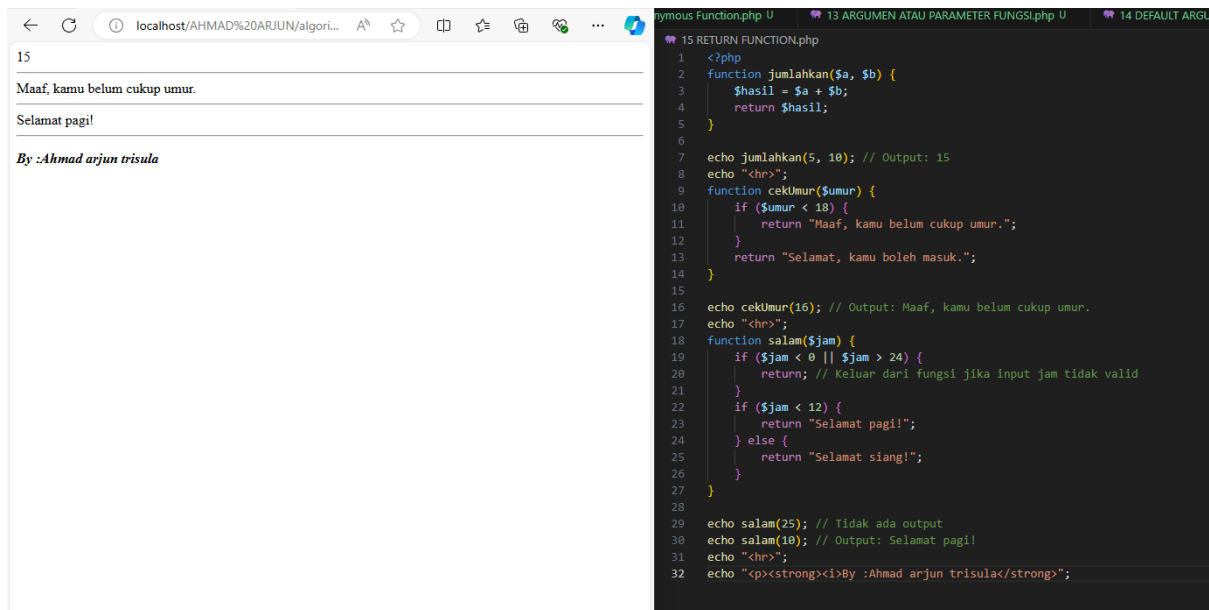
```
function cekUmur($umur) {  
    if ($umur < 18) {  
        return "Maaf, kamu belum cukup umur.";  
    }  
    return "Selamat, kamu boleh masuk.";  
}  
  
echo cekUmur(16); // Output: Maaf, kamu belum cukup umur.
```


Pada contoh di atas, jika umur pengguna kurang dari 18 tahun, fungsi akan langsung mengembalikan pesan dan menghentikan eksekusinya.

Return Tanpa Nilai

Kadang kala, return digunakan tanpa menyertakan nilai apa pun, hanya untuk keluar dari fungsi. Meskipun tidak mengembalikan nilai, ini tetap berguna untuk mengontrol alur program.

```
function salam($jam) {  
    if ($jam < 0 || $jam > 24) {  
        return; // Keluar dari fungsi jika input jam tidak valid  
    }  
    if ($jam < 12) {  
        return "Selamat pagi!";  
    } else {  
        return "Selamat siang!";  
    }  
}  
  
echo salam(25); // Tidak ada output  
echo salam(10); // Output: Selamat pagi!
```



INCLUDE DAN REQUIRE FUNCTION

Pelajari cara menggunakan perintah include dan require dalam PHP untuk memanggil file eksternal dalam skrip kamu, beserta perbedaan dan contohnya.

Menggunakan include dan require di PHP

Dalam pemrograman PHP, kadang kamu perlu menggunakan isi dari sebuah file PHP lain di dalam skrip yang sedang kamu kerjakan. Untuk tujuan ini, PHP menyediakan dua perintah yang sangat berguna: include dan require.

include

Perintah include digunakan untuk memasukkan isi file lain ke dalam file yang sedang dieksekusi. Jika file yang ingin di-include tidak ditemukan, PHP akan memberi peringatan (warning) namun skrip akan tetap dijalankan.

Contoh penggunaan:

```
include 'header.php';
```

Dengan kode di atas, isi dari header.php akan disertakan di tempat perintah include dipanggil.

Kapan menggunakan include?

Gunakan include ketika kamu ingin menyertakan file yang keberadaannya tidak krusial untuk eksekusi skrip keseluruhan. Artinya, skrip masih dapat berjalan walaupun file yang di-include-kan itu tidak ada atau terjadi kesalahan saat memanggilnya.

require

Sementara require sangat mirip dengan include, perbedaan utamanya adalah jika file yang di-require tidak ditemukan, PHP akan menghentikan eksekusi skrip dan memberikan pesan error (fatal error).

Contoh penggunaan:

```
require 'config.php';
```

Pada contoh tersebut, config.php merupakan file penting sehingga harus ada dan bisa diakses oleh skrip. Jika tidak, skrip tidak akan dijalankan.

Kapan menggunakan require?

Gunakan require ketika file yang kamu ingin sertakan adalah krusial bagi seluruh aplikasi, seperti konfigurasi, fungsi, atau kelas yang harus ada agar aplikasi dapat berjalan dengan semestinya.

include_once dan require_once

PHP juga menyediakan `include_once` dan `require_once`, yang masing-masing bekerja serupa dengan `include` dan `require`. Perbedaanannya adalah, perintah ini akan memeriksa apakah file tersebut sudah pernah di-include atau di-require sebelumnya dalam skrip yang sama. Jika ya, file tersebut tidak akan disertakan lagi. Ini berguna untuk mencegah masalah ketika berusaha menyertakan file yang sama lebih dari satu kali.

Contoh penggunaan `include_once`:

```
include_once 'library.php';
```

Contoh penggunaan `require_once`:

```
require_once 'constants.php';
```

Dengan menggunakan `*_once`, kamu bisa memastikan tidak akan terjadi duplikat kode akibat menyertakan file yang sama lebih dari satu kali.

DATE FUNCTION

Belajar cara menggunakan fungsi tanggal dan waktu dalam PHP untuk manipulasi dan penampilan data tanggal.

PHP menyediakan berbagai fungsi untuk bekerja dengan tanggal dan waktu yang memungkinkan kamu untuk melakukan manipulasi dan penampilan data tanggal dengan mudah. Artikel ini akan membahas cara menggunakan beberapa fungsi dasar tanggal pada PHP.

Mengambil Tanggal dan Waktu Saat Ini

Untuk mendapatkan tanggal dan waktu saat ini, kamu dapat menggunakan fungsi `date()` yang disertai dengan format string yang menentukan representasi outputnya.

```
echo date("Y-m-d H:i:s"); // output contoh: 2023-03-15 14:00:00
```

Dalam contoh di atas, “Y” merepresentasikan tahun, “m” bulan, “d” hari, “H” jam dalam format 24 jam, “i” menit, dan “s” detik.

Format Tanggal

PHP memungkinkan kamu untuk menformat tanggal sesuai yang kamu inginkan. Beberapa karakter yang sering digunakan untuk menformat tanggal adalah:

- d - Representasi hari dalam bentuk dua digit (01 sampai 31)
 - m - Representasi bulan dalam bentuk dua digit (01 sampai 12)
 - Y - Representasi tahun dalam bentuk empat digit
 - l (huruf ‘L’ kecil) - Representasi nama hari dalam seminggu
- Contoh penggunaan format:

```
echo date("l, d-m-Y"); // output contoh: Rabu, 15-03-2023
```

Mengatur Zona Waktu

Secara default, PHP menggunakan zona waktu yang diset pada konfigurasi server. Namun, kamu bisa mengubahnya dengan fungsi `date_default_timezone_set()`.

```
date_default_timezone_set('Asia/Jakarta');  
echo date("Y-m-d H:i:s"); // output akan sesuai dengan zona waktu Jakarta
```

Zona waktu harus sesuai dengan penulisan yang diakui oleh PHP, misal ‘America/New_York’, ‘Europe/Paris’, dan lain-lain.

Manipulasi Tanggal

PHP juga menyediakan fungsi `strtotime()` untuk manipulasi tanggal dan waktu. Fungsi ini mengkonversi sebuah string yang berisi tanggal dan waktu ke Unix timestamp.

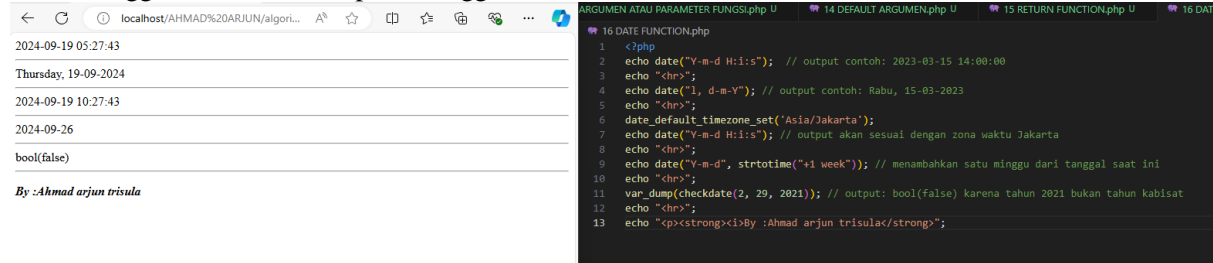
```
echo date("Y-m-d", strtotime("+1 week")); // menambahkan satu minggu dari tanggal saat ini
```

Memeriksa Keabsahan Tanggal

Untuk memeriksa apakah sebuah tanggal itu valid, gunakan fungsi `checkdate()`. Ini berguna ketika kamu bekerja dengan input tanggal dari pengguna.

```
var_dump(checkdate(2, 29, 2021)); // output: bool(false) karena tahun 2021 bukan tahun kabisat
```

Dengan memanfaatkan fungsi-fungsi ini, kamu dapat dengan mudah bekerja dengan tanggal dalam aplikasi PHP kamu. Mulai dari menampilkan tanggal dan waktu saat ini, menformat output tanggal, mengatur zona waktu, hingga melakukan manipulasi tanggal sesuai kebutuhan.



TIME FUNCTION

Pelajari cara menggunakan fungsi waktu di PHP untuk bekerja dengan tanggal dan waktu dengan mudah melalui contoh penggunaan yang sederhana.

Mengelola waktu merupakan salah satu fungsi penting dalam pemrograman web. Di PHP, ada beberapa fungsi yang dapat digunakan untuk mendapatkan dan memanipulasi informasi waktu. Artikel ini akan membimbing kamu melalui penggunaan dasar waktu di PHP.

Fungsi time()

Fungsi `time()` di PHP mengembalikan waktu saat ini dalam format UNIX timestamp, yaitu jumlah detik sejak tanggal 1 Januari 1970 (epoch time). Contoh penggunaannya sangat sederhana:

```
<?php
$sekarang = time();
echo $sekarang;
?>
```

Fungsi date()

Untuk menampilkan timestamp dalam format yang lebih terbaca oleh manusia, kita bisa menggunakan fungsi `date()`. Fungsi ini membutuhkan dua parameter: format tanggal dan waktu yang diinginkan, serta timestamp yang akan diformat.

```
<?php
echo date('Y-m-d H:i:s');
?>
```

Format Tanggal yang Sering Digunakan

- Y - Tahun (4 digit)
- m - Bulan (2 digit dengan leading zero)
- d - Hari dalam bulan (2 digit dengan leading zero)
- H - Jam dalam format 24 jam (2 digit dengan leading zero)
- i - Menit (2 digit dengan leading zero)
- s - Detik (2 digit dengan leading zero)

Menyetel Zona Waktu

Secara default, PHP mengambil nilai zona waktu dari konfigurasi server. Akan tetapi, kamu bisa menyetel zona waktu secara manual dengan fungsi `date_default_timezone_set()`.

```
<?php
date_default_timezone_set('Asia/Jakarta');
echo date('Y-m-d H:i:s');
?>
```

Manipulasi Tanggal dan Waktu dengan strtotime()

Fungsi `strtotime()` memungkinkan kamu untuk mengubah teks berbahasa Inggris yang menggambarkan tanggal dan waktu menjadi timestamp UNIX. Fungsi ini sangat berguna ketika kamu perlu melakukan operasi penambahan atau pengurangan pada nilai tanggal dan waktu.

```
<?php
$tomorrow = strtotime("tomorrow");
echo date('Y-m-d', $tomorrow);

$nextWeek = strtotime("+1 week");
echo date('Y-m-d', $nextWeek);
?>
```

Menggunakan `DateTime` dan `DateInterval`

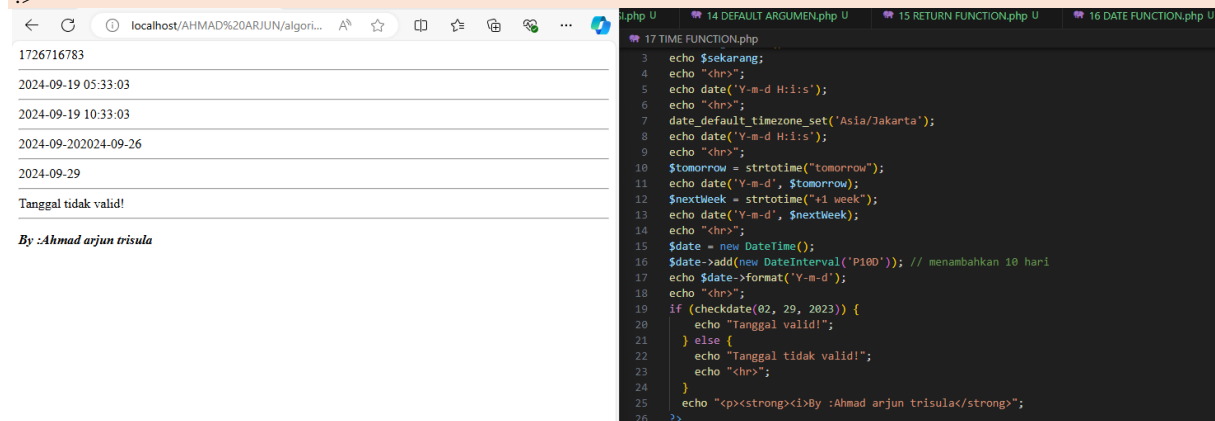
Kelas `DateTime` menyediakan metode-metode yang lebih objektif untuk pengelolaan tanggal dan waktu. Kamu dapat memanipulasi tanggal dengan menggunakan kelas `DateInterval`.

```
<?php
$date = new DateTime();
$date->add(new DateInterval('P10D')); // menambahkan 10 hari
echo $date->format('Y-m-d');
?>
```

Mengecek Tanggal Valid dengan `checkdate()`

Kadang kita perlu memastikan apakah suatu kombinasi tahun, bulan, dan tanggal merupakan tanggal yang valid. Fungsi `checkdate()` bisa digunakan untuk tujuan tersebut.

```
<?php
if (checkdate(02, 29, 2023)) {
    echo "Tanggal valid!";
} else {
    echo "Tanggal tidak valid!";
}
?>
```



Dengan pengetahuan dasar fungsi-fungsi waktu PHP ini, kamu bisa mengatasi berbagai kebutuhan terkait tanggal dan waktu dalam aplikasi web kamu. Ingatlah untuk selalu mencari dokumentasi yang relevan jika kamu membutuhkan fungsi yang lebih spesifik atau lanjutan.

STRTOTIME FUNCTION

Pelajari cara menggunakan fungsi `strtotime` di PHP untuk mengubah format tanggal menjadi timestamp serta contoh penggunaannya.

Mempelajari Fungsi `strtotime()` di PHP

Fungsi `strtotime()` adalah salah satu fungsi penting dalam PHP yang digunakan untuk mengubah format tanggal yang diberikan dalam bentuk string menjadi timestamp Unix. Timestamp Unix adalah jumlah detik yang telah berlalu sejak tanggal 1 Januari 1970. Fungsi ini sangat berguna saat kamu perlu melakukan operasi aritmatika tanggal atau mengubah format tanggal.

Penggunaan Dasar strtotime()

Untuk menggunakan fungsi strtotime(), kamu cukup memberikan string tanggal yang ingin diubah ke dalam fungsi tersebut. Berikut adalah contoh penggunaan dasarnya:

```
<?php
$timestamp = strtotime("now");
echo $timestamp; // Menampilkan timestamp untuk waktu saat ini

$timestamp = strtotime("10 September 2000");
echo $timestamp; // Menampilkan timestamp untuk tanggal 10 September 2000
?>
```

Format Tanggal yang Didukung

strtotime() mendukung banyak format tanggal yang berbeda, termasuk:

"now" untuk waktu saat ini

"10 September 2000" untuk tanggal tertentu

"next Friday" untuk hari Jumat selanjutnya

"+1 day" untuk menambahkan satu hari ke waktu saat ini

"+1 week 2 days 4 hours 2 seconds" untuk operasi kompleks

Contoh penggunaan format tanggal:

```
<?php
$nextWeek = strtotime("+1 week");
echo date("Y-m-d", $nextWeek); // Menampilkan tanggal satu minggu dari waktu saat ini
?>
```

Penanganan Zona Waktu

Kamu mungkin perlu menetapkan atau mengubah zona waktu di script PHP untuk memastikan strtotime() memberikan waktu yang benar.

```
<?php
date_default_timezone_set('Asia/Jakarta');
$localTime = strtotime("now");
echo $localTime; // Menampilkan timestamp dengan zona waktu Asia/Jakarta
?>
```

Memformat Ulang Tanggal

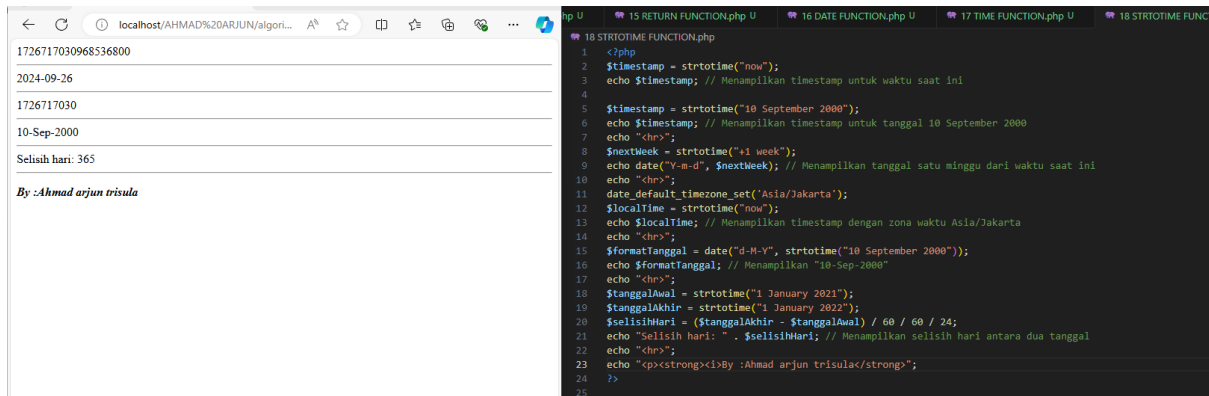
Setelah mendapatkan timestamp, kamu dapat memformat ulang tanggal sesuai kebutuhanmu menggunakan fungsi date().

```
<?php
$formatTanggal = date("d-M-Y", strtotime("10 September 2000"));
echo $formatTanggal; // Menampilkan "10-Sep-2000"
?>
```

Menghitung Selisih Waktu

strtotime() sangat berguna ketika kamu harus menghitung selisih waktu. Sebagai contoh, untuk menentukan selisih hari antara dua tanggal:

```
<?php
$tanggalAwal = strtotime("1 January 2021");
$tanggalAkhir = strtotime("1 January 2022");
$selisihHari = ($tanggalAkhir - $tanggalAwal) / 60 / 60 / 24;
echo "Selisih hari: " . $selisihHari; // Menampilkan selisih hari antara dua tanggal
?>
```



Dengan memahami cara kerja strtotime(), kamu akan mendapatkan alat yang sangat fleksibel untuk manipulasi dan penghitungan tanggal di PHP. Praktekkan beberapa contoh di atas untuk memperkuat pemahamanmu tentang fungsi ini.

ISSET FUNCTION

Pelajari cara menggunakan fungsi isset pada PHP untuk mengecek keberadaan variabel dengan panduan langkah demi langkah yang mudah dipahami.

Pemrograman PHP seringkali melibatkan pengecekan terhadap variabel yang dapat mempengaruhi alur eksekusi program. Salah satu fungsi yang berguna untuk tujuan ini adalah isset(). Fungsi ini digunakan untuk memverifikasi apakah sebuah variabel telah di-set atau tidak, artinya variabel tersebut telah didefinisikan dan bukan NULL.

Penggunaan Fungsi isset()

Untuk memahami cara kerja isset(), kita dapat melihat contoh penggunaannya:

```
<?php
$var = "Halo, dunia!";

if (isset($var)) {
    echo "Variabel 'var' telah di-set.";
} else {
    echo "Variabel 'var' belum di-set.";
}
?>
```

Pada contoh di atas, isset(\$var) akan menghasilkan true karena variabel \$var telah di-set dengan nilai string “Halo, dunia!”.

Menchecek Beberapa Variabel Sekaligus

Kamu juga bisa mengecek beberapa variabel sekaligus dalam satu panggilan isset():

```
<?php
$var1 = "PHP";
$var2 = "isset";

if (isset($var1, $var2)) {
    echo "Kedua variabel telah di-set.";
} else {
    echo "Salah satu atau kedua variabel belum di-set.";
}
?>
```

Fungsi ini akan mengembalikan true hanya jika semua variabel yang diberikan telah di-set.

Penggunaan dengan Array

isset() juga dapat digunakan untuk mengecek apakah suatu kunci atau indeks ada pada array:

```
<?php
$array = array("nama" => "John", "umur" => 25);
```

```

if (isset($array["nama"])) {
    echo "Kunci 'nama' ada di array.";
} else {
    echo "Kunci 'nama' tidak ada di array.";
}

if (isset($array["alamat"])) {
    echo "Kunci 'alamat' ada di array.";
} else {
    echo "Kunci 'alamat' tidak ada di array.";
}
?>

```

Pada contoh ini, `isset($array["nama"])` akan mengembalikan `true` karena ada elemen dengan kunci “nama” dalam array `$array`. Sebaliknya, `isset($array["alamat"])` akan mengembalikan `false`.

Mengecek Nilai NULL

Perlu diingat bahwa `isset()` akan mengembalikan `false` jika variabel telah di-set tapi nilainya `NULL`:

```

<?php
$var3 = NULL;

if (isset($var3)) {
    echo "Variabel 'var3' telah di-set.";
} else {
    echo "Variabel 'var3' belum di-set atau `NULL`.";
}
?>

```

Kombinasi dengan Fungsi Lain

Kadang-kadang, `isset()` digunakan bersama dengan fungsi `empty()` untuk memastikan variabel tidak hanya di-set, tetapi juga memiliki nilai yang berarti:

```

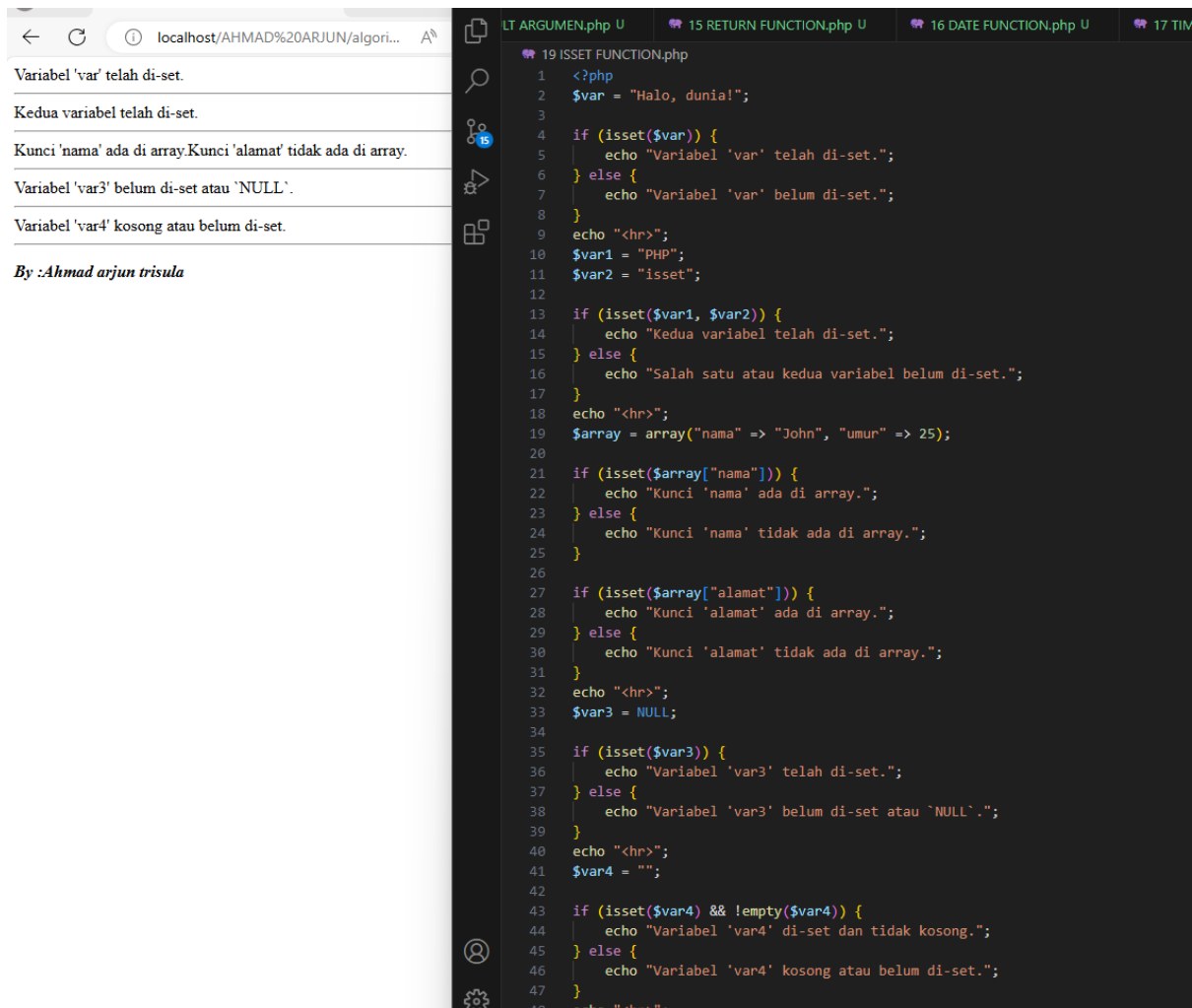
<?php
$var4 = "";

if (isset($var4) && !empty($var4)) {
    echo "Variabel 'var4' di-set dan tidak kosong.";
} else {
    echo "Variabel 'var4' kosong atau belum di-set.";
}
?>

```

Fungsi `empty()` akan mengecek apakah variabel adalah empty string, 0, `NULL`, array kosong, atau belum di-set sama sekali.

Dengan memahami penggunaan `isset()`, kamu akan lebih mudah menangani situasi dimana programmu memerlukan kepastian terhadap keadaan variabel sebelum dilakukan proses lebih lanjut.



EMPTY FUNCTION

Pelajari cara menggunakan fungsi empty() dalam PHP untuk memeriksa apakah suatu variabel kosong atau tidak.

Mengecek Variabel Kosong dengan Fungsi empty dalam PHP

Dalam pengembangan web menggunakan PHP, sering kali kita perlu memeriksa apakah suatu variabel kosong atau tidak. PHP menyediakan fungsi bawaan yang bernama empty untuk memudahkan proses ini. Fungsi ini sangat bermanfaat, terutama saat kamu bekerja dengan form input, data yang diterima dari pengguna, atau kondisi yang membutuhkan validasi.

Penggunaan Fungsi empty

Fungsi empty akan mengembalikan true jika variabel yang diperiksa tidak ada, atau nilai dari variabel tersebut dianggap kosong. Menurut dokumentasi PHP, sebuah variabel dianggap kosong jika ia mengandung nilai:

"" (string kosong)

0 (0 sebagai integer)

0.0 (0 sebagai float)

"0" (0 sebagai string)

null

false

array kosong

Sintaks

bool empty (mixed \$var)

Contoh Penggunaan

Pertimbangkan contoh berikut:

```

$var1 = "";
$var2 = 0;
$var3 = false;
$var4 = array();

if(empty($var1)) {
    echo 'Svar1 kosong';
}

if(empty($var2)) {
    echo 'Svar2 kosong';
}

if(empty($var3)) {
    echo 'Svar3 kosong';
}

if(empty($var4)) {
    echo 'Svar4 kosong';
}

```

The screenshot shows a web browser window at the top with the URL `localhost/AHMAD%20ARJUN/algorithm.php`. The browser's output area displays the text: `Svar1 kosongSvar2 kosongSvar3 kosongSvar4 kosong`. Below the browser window, a code editor shows the PHP code used to generate this output. The code is as follows:

```

1 <?php
2 $var1 = "";
3 $var2 = 0;
4 $var3 = false;
5 $var4 = array();
6
7 if(empty($var1)) {
8     echo 'Svar1 kosong';
9 }
10
11 if(empty($var2)) {
12     echo 'Svar2 kosong';
13 }
14
15 if(empty($var3)) {
16     echo 'Svar3 kosong';
17 }
18
19 if(empty($var4)) {
20     echo 'Svar4 kosong';
21 }
22 echo "<p><strong><i>By :Ahmad arjun trisula</strong>";
23
24

```

Hasil dari setiap if statement akan mencetak variabel yang bersangkutan adalah kosong, karena setiap variabel memenuhi kriteria dari fungsi empty.

Perbedaan antara empty, isset, dan is_null

Ketika bekerja dengan variabel, ada beberapa fungsi lain yang juga sering digunakan, yaitu `isset` dan `is_null`. Fungsi-fungsi ini memiliki kegunaan yang mirip tapi berbeda dari `empty`.

`isset` menguji apakah sebuah variabel sudah didefinisikan atau tidak dan apakah nilainya tidak null. Jadi, jika sebuah variabel sudah didefinisikan dan nilainya bukan null, fungsi ini akan mengembalikan `true`.

`is_null` akan mengembalikan `true` hanya jika variabel belum diinisialisasi atau nilainya adalah null.

Pemahaman yang baik tentang kapan harus menggunakan `empty`, `isset`, dan `is_null` akan membantu kamu dalam pengecekan kondisi variabel dengan cara yang lebih efisien dan efektif.

HEADER FUNCTION

Pelajari cara menggunakan header dalam PHP untuk mengatur respons HTTP, alihkan halaman, dan lebih lanjut dengan contoh yang mudah dipahami.

Mengelola Header HTTP dengan PHP

Dalam pembuatan situs web, mengatur header HTTP sangat penting karena memberi instruksi kepada browser bagaimana harus menangani konten yang dikirimkan oleh server. PHP memiliki fungsi bawaan yang memungkinkan kita untuk memanipulasi header ini dengan mudah.

Header Location: Pengalihan Halaman

Salah satu kegunaan utama fungsi `header()` dalam PHP adalah untuk mengarahkan pengguna ke halaman yang berbeda, yang dikenal sebagai pengalihan HTTP. Contoh di bawah ini menunjukkan bagaimana mengarahkan pengguna ke halaman lain:

```
<?php
header("Location: http://www.contoh.com/halaman_baru.php");
exit; // Selalu panggil exit setelah header location untuk menghentikan eksekusi skrip
?>
```

Penting untuk memanggil `exit` atau `die()` setelah fungsi `header()` agar skrip PHP tidak terus dieksekusi setelah pengalihan.

Mengatur Konten Tipe Header

Kamu juga bisa menggunakan header untuk memberitahu browser tentang tipe konten yang dikirim. Ini berguna, misalnya, ketika kamu ingin mengirimkan file untuk di-download atau menampilkan gambar.

```
<?php
header("Content-Type: application/pdf");
// Kode untuk mengirimkan file PDF
?>
```

Mengontrol Cache dengan Header

Mengatur header agar konten tidak disimpan dalam cache browser dapat sangat berguna, terutama untuk mengontrol konten yang sensitif atau sering berubah.

```
<?php
header("Cache-Control: no-cache, no-store, must-revalidate");
header("Pragma: no-cache");
header("Expires: 0");
?>
```

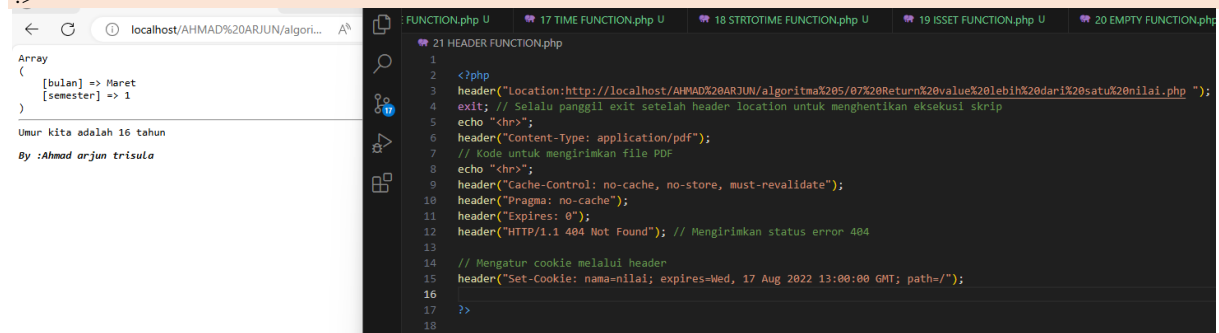
Header di atas akan memberitahu browser untuk tidak menyimpan konten di cache.

Menggunakan Lainnya Header

PHP mengizinkan pengaturan berbagai header HTTP untuk keperluan lainnya seperti autentikasi, pengaturan cookies, dan lain-lain.

```
<?php
header("HTTP/1.1 404 Not Found"); // Mengirimkan status error 404

// Mengatur cookie melalui header
header("Set-Cookie: nama=nilai; expires=Wed, 17 Aug 2022 13:00:00 GMT; path=/");
?>
```



Menggunakan fungsi `header()` dengan benar sangat penting dalam pengembangan web untuk melakukan kontrol yang lebih baik atas apa yang dikirimkan kepada pengguna.

Penting untuk diingat bahwa kamu harus memanggil fungsi `header()` sebelum ada output apapun yang dikirimkan oleh script. Hal ini karena header HTTP harus dikirim ke browser sebelum konten apapun. Jika sudah ada output yang dikirim, PHP akan menghasilkan error "headers already sent". Jadi perhatikan urutan pemanggilan dalam script-mu.



MATERI PERTEMUAN 11

Memahami Method Form PHP



TAHUKAH KAMU...?

Pada kesempatan ini, kita akan membahas:

- Pengertian GET dan POST pada PHP dan HTTP
- Metode GET
- Method POST
- Kapan Menggunakan Method GET dan POST pada PHP
- \$_REQUEST pada PHP

1. Pengertian GET dan POST pada PHP dan HTTP

Dalam dunia internet, protokol yang umum digunakan adalah protokol HTTP, protokol ini memiliki beberapa metode request (request method) diantaranya adalah dari GET dan POST, jadi GET dan POST ini berdiri sendiri tidak berhubungan dengan bahasa pemrograman seperti PHP dan ASP, sehingga jika kita membicarakan GET dan POST pada PHP, maka sebenarnya kita sedang membicarakan GET dan POST pada HTTP. Dalam berkomunikasi, PHP menggunakan protokol HTTP, oleh karena itu PHP juga menyediakan sarana untuk berinteraksi dengan kedua metode request tersebut yaitu: (1) menyimpan data GET dan POST dan (2) mengirim data GET dan POST.

2. Metode GET

Dalam bahasa Inggris kita akrab dengan istilah GETting, dari istilah tersebut dapat diartikan bahwa metode GET pada HTTP ditujukan untuk mengambil (get) data dari server. Pada metode ini umumnya data berbentuk query string yang dikirim via url, data tersebut berupa pasangan `key=value` yang dipisahkan dengan tanda `&`. Data tersebut digabung dengan url utama yang dipisahkan dengan tanda `?`.

Sebelum dikirim, terlebih dahulu data diproses sehingga memenuhi standar format URL. URL hanya boleh memuat **huruf** (besar dan kecil), **angka**, dan beberapa karakter lain dalam ASCII Character Set seperti `“-._~`), karakter di luar itu akan diubah ke format tertentu yang diawali tanda `%` kemudian diikuti dengan 2 digit hexadecimal, contoh:

Karakter	URL Encoded
?	%3F
@	%40
=	%3D

Angka pada kolom (URL Encoded) merupakan nilai hexadecimal dari character ASCII, disamping itu URL juga tidak boleh memuat spasi, sehingga spasi akan diubah menjadi tanda `+` atau `%20`. Semua proses tersebut disebut **url encoding**. Metode GET adalah metode yang datanya dikirim melalui URL, data yang dikirim di URL berupa rangkaian pasangan nama dan nilai yang dipisahkan oleh ampersand (`&`). URL dengan data GET akan terlihat sebagai berikut:

```
https://ahmadmuslim.info/data.php?name=alfian&age=21
```

Apabila kita lihat dari URL diatas, disana terdapat nama untuk file PHP yaitu `data.php`, Nah File ini lah yang kita tuju pada Formulir GET. Berikutnya ada pembatas antara File PHP dan juga Variable yang dikirimkan yaitu Tanda Tanya (`?`), dan yang terakhir yaitu Variable yang dikirimkan beserta isi datanya yang dipisah dengan Tanda Ampersand (`&`). Variable yang dikirimkan adalah `name` dan `age`.

Nah, sekarang kita praktekkan bagaimana penggunaan Method GET ini pada Codingan kita, yang kita butuhkan adalah syntax PHP yang berisi Form dan memiliki Method GET.

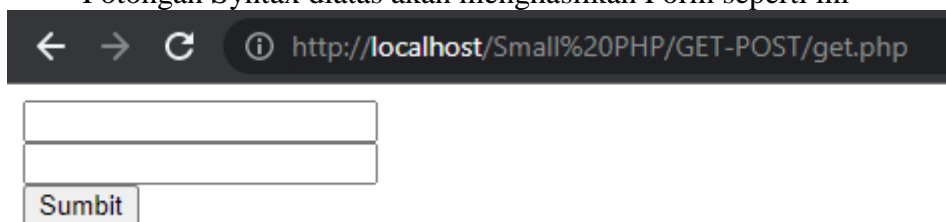
```
<html lang="en">

<head>
  <title>Method GET</title>
</head>

<body>
  <form action="" method="GET">
    <input type="text" name="nama"><br />
    <input type="number" name="umur"><br />
    <input type="submit" name="submit" value="Sumbit">
  </form>

  <?php
  if ($_GET) {
    echo "Nama: " . $_GET["nama"];
    echo "<br/>";
    echo "Umur: " . $_GET["umur"];
  }
  ?>
</body>
</html>
```

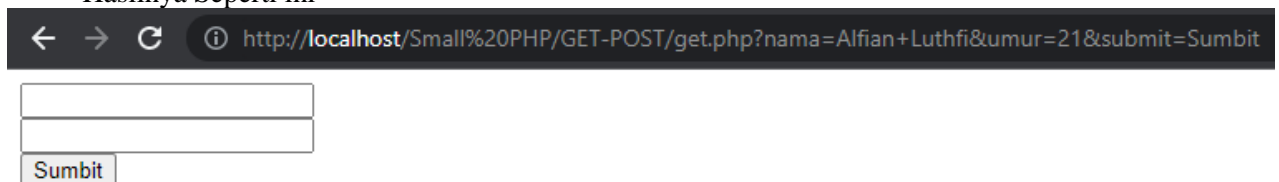
Potongan Syntax diatas akan menghasilkan Form seperti ini



← → ↻ ⓘ http://localhost/Small%20PHP/GET-POST/get.php

Misalkan kita mengisi Form diatas dengan Nama dan Umur kita. Maka Codingan kita akan mengambil Data yang kita isi menggunakan \$_GET lalu ditampilkan dengan echo.

Hasilnya Seperti ini



← → ↻ ⓘ http://localhost/Small%20PHP/GET-POST/get.php?nama=Alfian+Luthfi&umur=21&submit=Sumbit

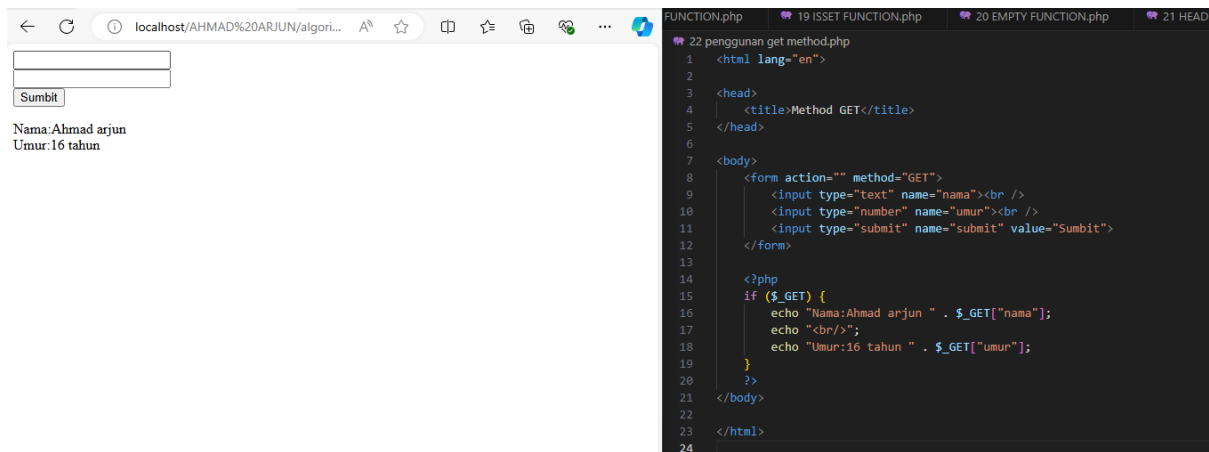
Nama: Alfian Luthfi
Umur: 21

Bisa dilihat di URL diatas. Ada tulisan nama=Alfian+Luthfi&umur=21. Seperti yang dijelaskan tadi, Method GET ini menggunakan URL, jadi kita bisa mengganti Outputnya dengan mengganti isi URL nya. Misal sekarang kita ganti nilai umur pada URL jadi 19 jadi nama=Alfian+Luthfi&umur=19. Maka Outputnya juga akan berubah seperti ini



← → ↻ ⓘ http://localhost/Small%20PHP/GET-POST/get.php?nama=Alfian+Luthfi&umur=19&submit=Sumbit

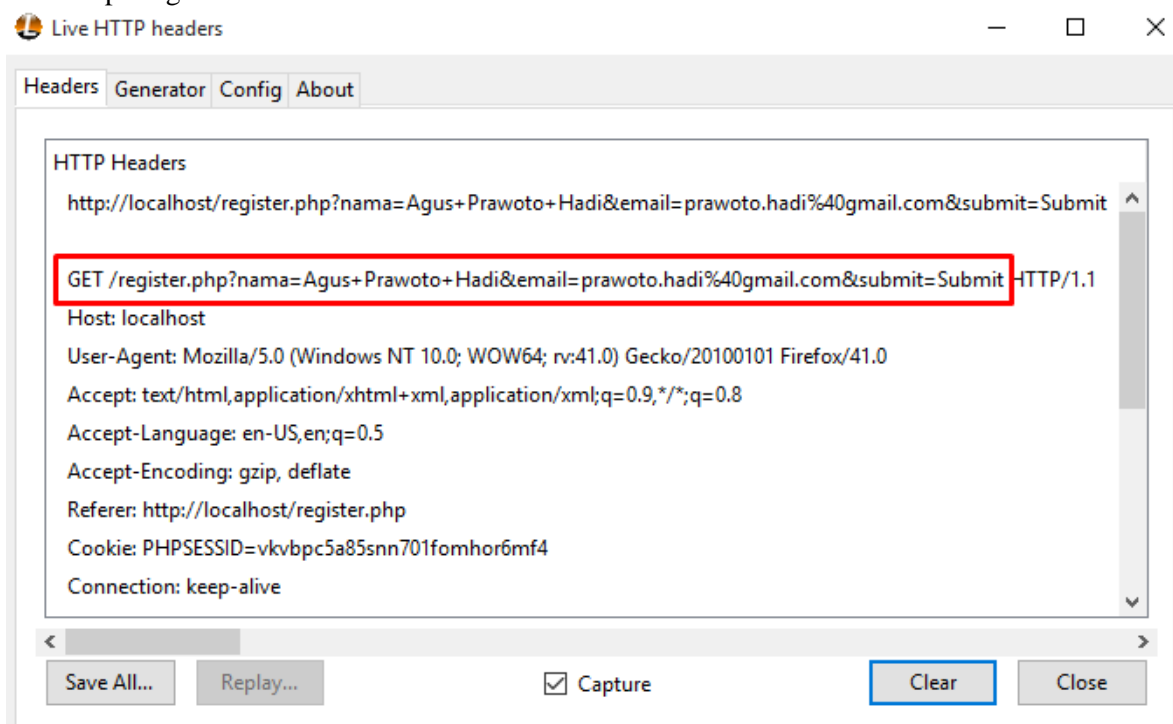
Nama: Alfian Luthfi
Umur: 19



Berikut ini merupakan contoh penggunaan metode GET pada form. Kita buat file registrasi.php dan tuliskan kode HTML berikut ini:

```
<html>
<body>
    <form method="GET" action="">
        <input type="text" name="nama"><br>
        <input type="text" name="email"><br>
        <input type="submit" name="submit" value="Submit">
    </form>
</body>
</html>
```

setelah itu simpan dan jalankan pada browser, pada field yang muncul, misal kita isikan nama: Ahmadi, email: bangmuslim@gmail.com, jika kita klik Submit, maka browser akan mengirim request dengan bentuk seperti gambar berikut ini:



dari gambar diatas terlihat bahwa metode yang kita gunakan adalah GET disertai query string nya, disamping itu url pada browser juga akan berubah menjadi:

```
https://jagowebdev.com/?nama=Agus+Prawoto+Hadi&email=prawoto%40gmail.com&submit=submit
```

dari data diatas terlihat bahwa <spasi> baik pada query string maupun pada url diencode menjadi tanda + dan @ menjadi %40. Ketika kita ambil data tersebut dengan PHP maka otomatis data pada url

akan didecode sehingga kita dapatkan bentuk asli dari data yang kita kirim, mari kita tambahkan kode pada file registrasi.php, sehingga menjadi:

```
<html>
<body>
    <form method="GET" action="">
        <input type="text" name="nama"><br>
        <input type="text" name="email"><br>
        <input type="submit" name="submit" value="Submit">
    </form>

    <?php
    if ($_GET)
    {
        echo 'Nama: ' . $_GET['nama'];
        echo '<br>';
        echo 'Email: ' . $_GET['email'];
    }
    ?>
</body>
</html>
```

ketika kita refresh browser maka akan kita dapatkan hasil:

```
Nama: Ahmadi
Email: bangmuslim@gmail.com
```

Variabel `$_GET` pada PHP berbentuk associative array. Variabel ini bentuknya sama seperti variabel pada umumnya, bedanya `$_GET` ini merupakan variabel global sehingga bisa diakses dimana saja.

Karena bentuknya sama dengan yang lain, variabel ini dapat kita manipulasi sebagaimana kita memanipulasi variabel array lainnya, misal dengan menambahkan nilainya:

```
$_GET['status'] = 'aktif' atau menghapusnya unset($_GET['nama'])
```

Kelebihan dan Kekurangan Method Get

Terdapat beberapa **kelebihan** penggunaan metode GET, diantaranya adalah:

1. Sempel, dan data mudah diedit, misal untuk menuju halaman 5 dari suatu website, kita tinggal mengganti urlnya.
2. Halaman dapat dibookmark dan disimpan pada history browser sehingga mudah untuk diakses kembali.
3. Dapat kembali ke halaman sebelumnya dengan mudah (dengan mengklik tombol Back pada browser).
4. Dapat direfresh dengan mudah.
5. Dapat di distribusikan/dishare.

Meskipun banyak kelebihannya, penggunaan metode ini memiliki beberapa kelemahan yaitu:

1. Panjang data terbatas hanya 2kb – 8kb (tergantung browsernya), jika melebihi batas tersebut akan muncul pesan error 414 Request-URI Too Long, sehingga tidak dapat digunakan untuk mengirim data dalam jumlah besar.
2. Hanya dapat mengirim data jenis teks, jenis lainnya seperti: gambar, file zip, dll tidak dapat dikirim.
3. Karena data dikirim via URL, data tersebut mudah terekspose.

Seperti itulah penggunaan Method GET pada PHP. Ada yang sering mengatakan Methode ini tidak aman karena Data yang kita inputkan terlihat di URL dan bisa diganti-ganti secara asal.

3. Method POST

Kita sering mendengar istilah post, yang biasanya terkait dengan POSTing ke sosial media, dimana pada kegiatan tersebut kita mengirim data berupa tulisan atau gambar untuk disimpan di server

sosial media tersebut. Begitu juga dengan istilah POST pada HTTP, POST digunakan untuk mengirim data yang biasanya di gunakan untuk menambah/merubah data pada server.

Pada protokol HTTP, metode POST dapat dikirim baik melalui query string maupun body, seperti pada GET, data yang dikirim melalui query string akan ditampilkan pada URL dan sedangkan yang dikirim melalui body tidak terlihat oleh user.

Pada PHP, data POST yang dikirim melalui query string disimpan pada variabel \$_GET (seperti metode GET) sedangkan yang dikirim melalui body disimpan pada variabel \$_POST. Sama seperti \$_GET, variabel \$_POST juga berbentuk associative array dan bersifat global yang artinya dapat diakses dimana saja, selain itu juga dapat dilakukan manipulasi sebagaimana variabel array lainnya.

Method POST adalah metode pengiriman data yang Datanya tidak disimpan pada URL. Data pada method POST ini tetap dikirimkan akan tetapi tidak ditampilkan pada URL seperti GET. Method POST ini biasanya digunakan saat registrasi yang membutuhkan input email dan password yang seharusnya tidak muncul di URL.

Method POST ini dirasa lebih aman daripada method GET, bahkan Method ini juga bisa mengirimkan File seperti gambar dan dokumen, tidak hanya Text saja. Bagaimana cara penggunaan dan Contohnya? Mari kita lihat.

```
<html lang="en">

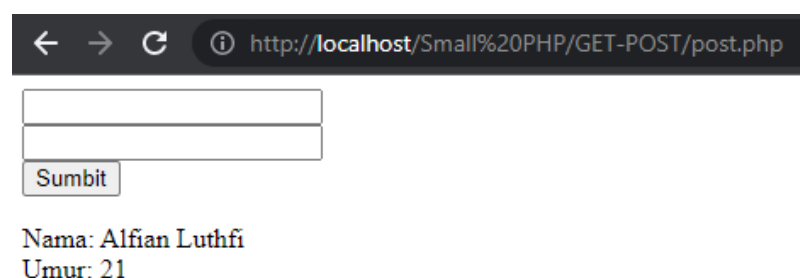
<head>
  <title>Method POST</title>
</head>

<body>
  <form action="" method="POST">
    <input type="text" name="nama"><br />
    <input type="number" name="umur"><br />
    <input type="submit" name="submit" value="Sumbit">
  </form>

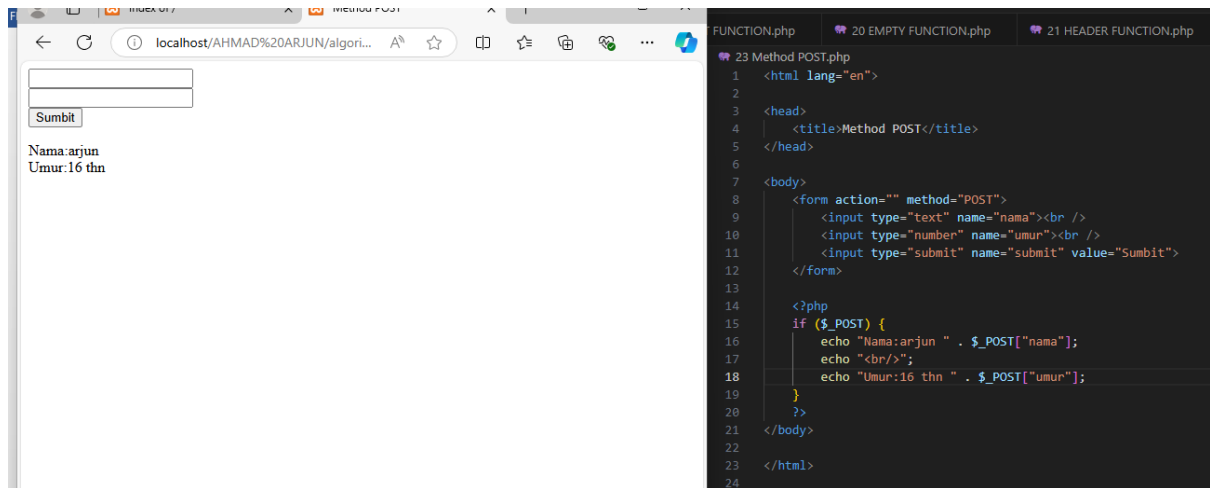
  <?php
  if ($_POST) {
    echo "Nama: " . $_POST["nama"];
    echo "<br/>";
    echo "Umur: " . $_POST["umur"];
  }
  ?>
</body>

</html>
```

Potongan syntax diatas akan menghasilkan Hasil yang sama seperti yang ada di Method GET. Tapi bedanya, saat kita isi kita tidak bisa melihat yang kita inputkan di URL. Beginilah hasilnya



The screenshot shows a web browser window with the address bar displaying `http://localhost/Small%20PHP/GET-POST/post.php`. Below the address bar, there is a form with two input fields: a text field and a number field. A button labeled "Sumbit" is positioned below the number field. The output of the form is displayed below the button, showing "Nama: Alfian Luthfi" and "Umur: 21".



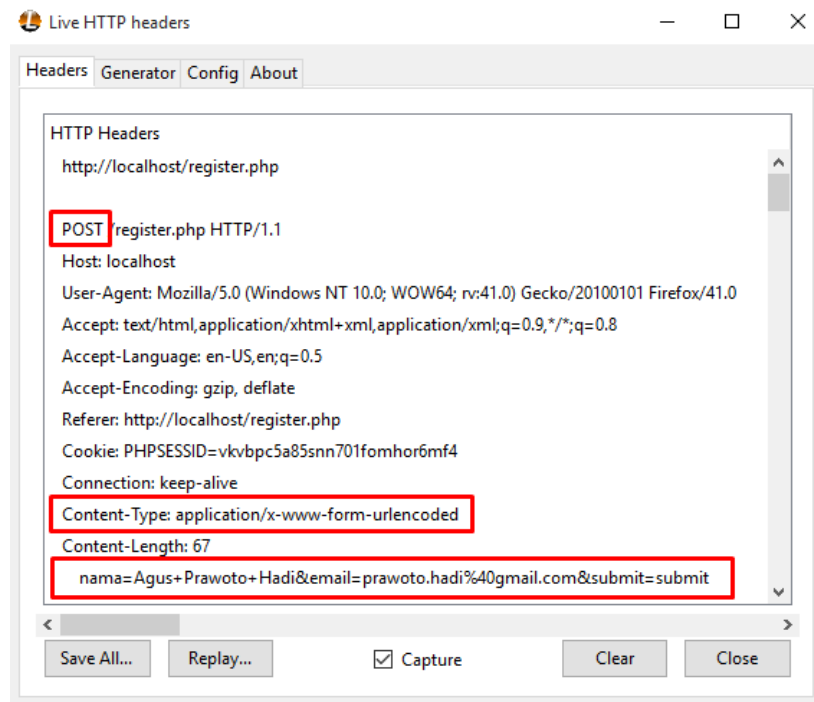
Bisa dilihat pada gambar diatas. Output yang keluar sama, akan tetapi di URL tidak muncul data seperti pada method GET. Pada akhirnya kita tidak bisa mengganti Data yang kita inputkan dan tidak bisa melihat data tersebut. Membuat penginputan data lebih *Secure*.

Penggunaan metode POST sering kita jumpai terutama pada saat pengiriman data menggunakan form html. Misal: meneruskan contoh sebelumnya, pada file registrasi.php kita ganti method pada bagian form dari `get` menjadi `post`

```
<html>
<body>
    <form method="POST" action="">
        <input type="text" name="nama"><br>
        <input type="text" name="email"><br>
        <input type="submit" name="submit" value="submit">
    </form>

    <?php
    if ($_POST)
    {
        echo 'Nama: ' . $_POST['nama'];
        echo '<br>';
        echo 'Email: ' . $_POST['email'];
    }
    ?>
</body>
</html>
```

misal field kita isi dengan nama: Ahmadi dan Email: bangmuslim@gmail.com, ketika kita klik submit, maka browser akan mengirim request ke server dengan bentuk seperti gambar dibawah ini:



dari gambar diatas terlihat data dikirim pada bagian body (kotak merah paling bawah) dan data yang dikirim diencode dengan sistem sama dengan GET (url encode) yang memang secara default cara pengiriman data pada POST adalah menggunakan `application/x-www-form-urlencoded`, disamping itu juga ada `multipart/form-data` yang digunakan untuk pengiriman data berupa file/binary.

POST yang dikirim via query string dan HTTP body

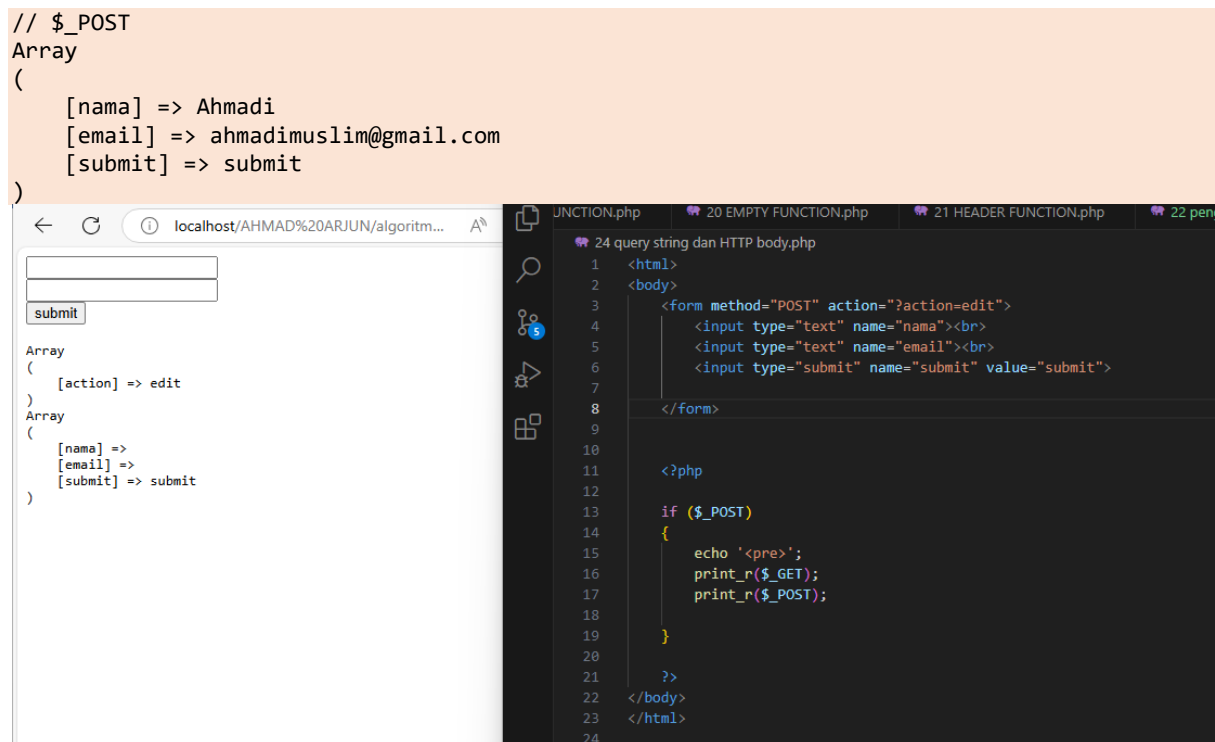
Sebelumnya telah kita singgung mengenai pengiriman POST melalui query string yang artinya akan ditampilkan pada URL. Sering kita fahami bahwa semua yang ada di URL adalah GET, karena kita sering berinteraksi dengan variabel `$_GET` dan dalam praktek hal tersebut tidak pernah menjadi masalah, tidak akan ada error yang muncul. Namun demikian sebenarnya yang terjadi tidaklah demikian, coba kita buka kembali file `register.php` dan kita ubah menjadi:

```
<html>
<body>
    <form method="POST" action="?action=edit">
        <input type="text" name="nama"><br>
        <input type="text" name="email"><br>
        <input type="submit" name="submit" value="submit">
    </form>

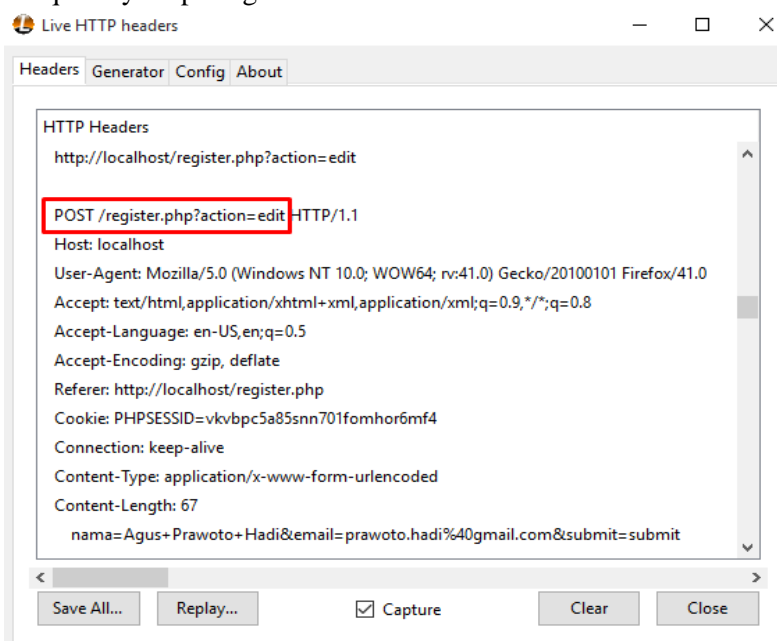
    <?php
    if ($_POST)
    {
        echo '<pre>';
        print_r($_GET);
        print_r($_POST);
    }
    ?>
</body>
</html>
```

misal kita isi dengan data yang sama dengan contoh sebelumnya, maka ketika kita klik submit, hasil yang kita peroleh adalah:

```
// $_GET
Array
(
    [action] => edit
)
```



dan bentuk HTTP Requestnya seperti gambar dibawah ini:



dari gambar diatas terlihat bahwa metode yang kita gunakan adalah POST dan data dikirim dengan dua cara baik melalui query string maupun HTTP body. query string disimpan pada variabel `$_GET` dan HTTP body disimpan pada variabel `$_POST`

Kelebihan dan kekurangan Method Post

Pengiriman data menggunakan metode POST memiliki beberapa kelebihan diantaranya:

1. Lebih aman dari pada metode GET karena data yang dikirim tidak terlihat, serta parameter yang dikirim tidak disimpan pada history browser/log browser.
2. Dapat mengirim data dalam jumlah besar.
3. Dapat mengirim berbagai jenis data seperti gambar, file, dll, tidak harus teks.

Meskipun terdapat kelebihan, penggunaan metode ini juga memiliki beberapa kelemahan, walaupun sebenarnya bukan kelemahan melainkan memang menjadi karakteristik dari metode ini:

1. Data tidak disimpan pada history browser.
2. Data tidak dapat dibookmark.

Karena dianggap sebagai data sensitif, maka ketika kita meresh browser, akan muncul konfirmasi pengiriman ulang data, demikian juga ketika kita tekan tombol back.

4. Kapan Menggunakan Method GET dan POST pada PHP

Untuk menentukan apakah kita akan menggunakan Method GET atau POST pada PHP, kita harus memikirkan terlebih dahulu: Apakah terjadi perubahan pada data di Server kita?. Apabila terjadi perubahan, jelas kita harus menggunakan Method POST, akan tetapi kalau tidak terjadi perubahan pada Server, kita bisa menggunakan Method GET.

Contohnya, misal kita membuat pagination pada PHP, lebih baik jelas menggunakan method GET.

POST digunakan saat ada Data Sensitif yang dikirimkan ke Database. Misal kan ID dari suatu hal, atau email dan password, tentunya kita harus menggunakan POST. Akan cukup bahaya apabila kita menggunakan GET, karena nanti semua orang bisa melihat Data Sensitif yang kita kirimkan.

Untuk memutuskan apakah kita akan menggunakan metode GET atau POST, kita harus selalu mengajukan pertanyaan: **apakah akan terjadi perubahan pada server?** jika ya, maka kita gunakan POST, jika tidak maka kita gunakan GET.

Contohnya adalah ketika membuat pagination, kita cukup menggunakan GET tidak perlu POST, bisa dibayangkan ketika kita sudah sampai halaman 5 suatu artikel, dan kita mau membookmarknya untuk dibaca di lain waktu, dan ternyata tidak bisa karena halaman tersebut menggunakan POST, dengan terpaksa kita harus membuka halaman 1 lagi!, Begitu juga dengan sistem pencarian seperti pada google, bisa dibayangkan jika metode yang digunakan adalah POST.

Contoh lainnya adalah penggunaan ID untuk menghapus data, perhatikan url berikut: <http://www.contoh.com/user.php?action=delete&id=1>, url ini akan berbahaya jika hal ini benar-benar terjadi. Kenapa? Karena data user akan terhapus dengan begitu mudahnya, lain halnya jika ID tersebut digunakan hanya untuk menampilkan data seperti <http://www.contoh.com/user.php?action=view&id=1>.

singkatnya GET untuk READ data, POST untuk CREATE, UPDATE, DELETE data. Pertanyaan kedua adalah **apakah ada data sensitif yang dikirim?** jika ya maka kita gunakan POST.

Contohnya adalah pengiriman username dan password ketika login, atau data keuangan seperti kartu kredit. Jika data tersebut dikirim via url maka jelas data tersebut akan terekspose kemana-mana terutama jika ter index oleh google. (Kecuali pada penggunaan AJAX). GET dan POST merupakan metode yang digunakan protokol HTTP untuk pertukaran data.

Pada PHP, data yang dikirim menggunakan metode GET akan disimpan dalam variabel \$_GET, sedangkan POST akan disimpan pada variabel \$_POST (untuk data yang dikirim via url disimpan pada variabel \$_GET). Masing masing metode memiliki kelebihan dan kekurangan dan memang sebenarnya kedua metode tersebut ditujukan untuk keperluan berbeda, sehingga kita harus tahu kapan menggunakan GET dan kapan menggunakan POST.

PHP juga menyediakan variabel \$_REQUEST yang merupakan gabungan dari variabel \$_GET, \$_POST dan \$_COOKIE. Pada variabel ini tidak terlihat dari mana datangnya data apakah dari GET atau POST sehingga sebaiknya lebih berhati hati ketika menggunakan variabel ini.

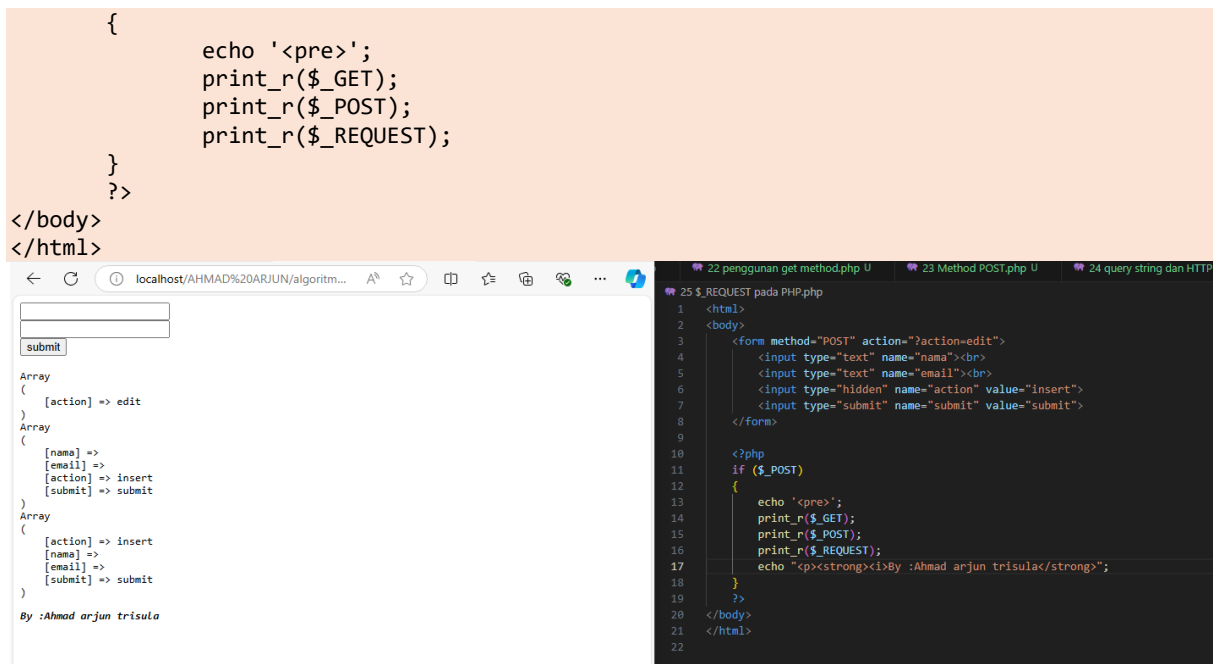
5. \$_REQUEST pada PHP

PHP menyediakan variabel global bernama \$_REQUEST, variabel ini merupakan gabungan dari tiga variabel yaitu: \$_GET, \$_POST dan \$_COOKIE. Karakteristik variabel ini sama dengan \$_GET dan \$_POST yaitu berbentuk associative array yang bersifat global dan dapat kita manipulasi seperti variabel array lainnya.

Dalam pembentukan variabel \$_REQUEST, variabel yang menjadi prioritas adalah \$_POST, sehingga jika antara \$_GET dan \$_POST terdapat `key` yang sama, maka yang digunakan adalah data pada variabel \$_POST, contoh mari kita ubah file register.php dengan menambahkan satu hidden field bernama action:

```
<html>
<body>
    <form method="POST" action="?action=edit">
        <input type="text" name="nama"><br>
        <input type="text" name="email"><br>
        <input type="hidden" name="action" value="insert">
        <input type="submit" name="submit" value="submit">
    </form>

    <?php
    if ($_POST)
```



Seperti pada contoh sebelumnya, kita isikan nama: Ahmadi dan email: bangmuslim@gmail.com, ketika kita submit maka kita dapatkan hasil:

```

// $_GET
Array
(
    [action] => edit
)

// $_POST
Array
(
    [nama] => Ahmadi
    [email] => ahmadimuslim@gmail.com
    [action] => insert
    [submit] => submit
)

// $_REQUEST
Array
(
    [action] => insert
    [nama] => Ahmadi
    [email] => prawoto.hadi@gmail.com
    [submit] => submit
)

```

Dari contoh diatas terlihat bahwa nilai action yang digunakan adalah insert yang terdapat pada variabel \$_POST. Kita pribadi jarang dan mungkin tidak pernah menggunakan variabel \$_REQUEST karena akan menyulitkan dan membuat ambigu, tidak jelas dari mana datangnya data apakah dari inputan user atau dari url, terlebih jika program yang kita buat kompleks, dengan menggunakan \$_GET atau \$_POST masalah tersebut tidak akan terjadi.

PENGAYAAN

Buatlah sebuah program hitung yang menggunakan method POST dan GET, dengan menerapkan penggunaan fungsi, operator, perulangan atau percabangan.!

Contoh:

```
<!DOCTYPE html>
```

```

<html>

<head>
    <title>WAKTU TEMPUH</title>
</head>

<body>

    <form action="" method="post">
        <h1>Program Hitung Waktu Tempuh</h1>
        Jarak Tempuh (km) : <br>
        <input type="text" name="jarak"><br><br>
        Kecepatan (km/jam): <br>
        <input type="text" name="cepat"><br><br>
        <input type="submit" value="Hitung Waktu">
    </form>

</body>

</html>

<?php
if (isset($_POST['jarak']) && isset($_POST['cepat'])) {

    $jarak = $_POST['jarak'];
    $cepat = $_POST['cepat'];

    $waktu = $jarak / $cepat;

    echo

    "
    <h1>Program Hitung Waktu Tempuh</h1>
    <p>Jarak : <b>$jarak</b> km</p>
    <p>Kecepatan : <b>$cepat</b> km/jam</p>
    <h3>Waktu Tempuh : $waktu jam</h3>
    ";
}
?>

```


Output:

Program Hitung Waktu Tempuh

Jarak Tempuh (km) :

Kecepatan (km/jam):

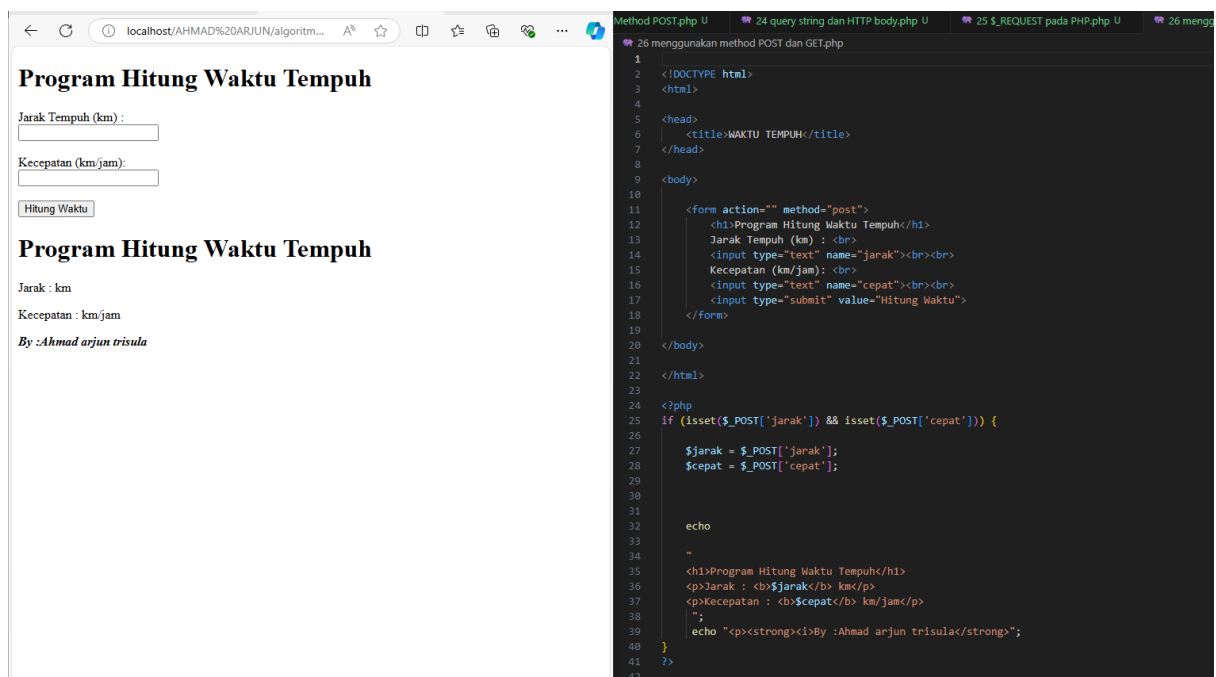
Hitung Waktu

Program Hitung Waktu Tempuh

Jarak : 100 km

Kecepatan : 50 km/jam

Waktu Tempuh : 2 jam



GET FORM

Pelajari cara mudah untuk mengirim data formulir menggunakan metode GET di PHP melalui tutorial langkah demi langkah ini.

Membuat dan mengirim data formulir adalah aspek penting dalam pengembangan web. Di PHP, metode GET adalah salah satu cara untuk mengirimkan data formulir dari browser ke server. Berikut adalah panduan singkat untuk mengimplementasikan formulir yang menggunakan metode GET di PHP.

Apa Itu Metode GET?

Metode GET dalam HTTP digunakan untuk meminta data dari sumber yang ditentukan. Dalam konteks formulir, ketika kamu menggunakan metode GET, data formulir dikirimkan melalui URL. Ini berguna untuk operasi pencarian atau ketika data yang dikirimkan tidak sensitif.

Kelebihan:

Data bisa langsung dilihat di URL.

Mudah digunakan untuk debugging.

Kekurangan:

Tidak aman untuk data sensitif karena nilai terlihat di URL.

Terbatas pada panjang URL yang bisa dihandle oleh browser.

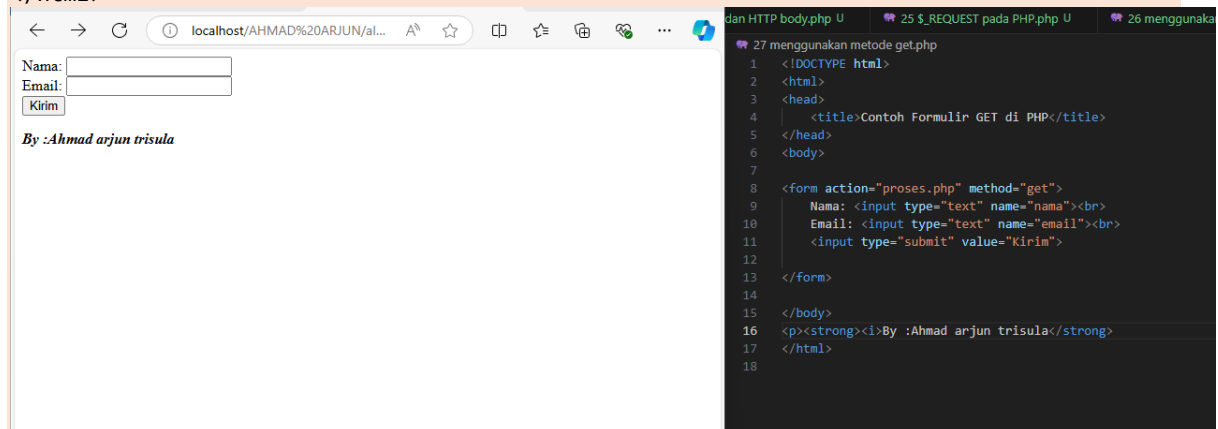
Membuat Formulir dengan Metode GET

Mari buat formulir HTML sederhana yang mengirimkan data menggunakan metode GET.

```
<!DOCTYPE html>
<html>
<head>
    <title>Contoh Formulir GET di PHP</title>
</head>
<body>
```

```
<form action="proses.php" method="get">
    Nama: <input type="text" name="nama"><br>
    Email: <input type="text" name="email"><br>
    <input type="submit" value="Kirim">
</form>

</body>
</html>
```



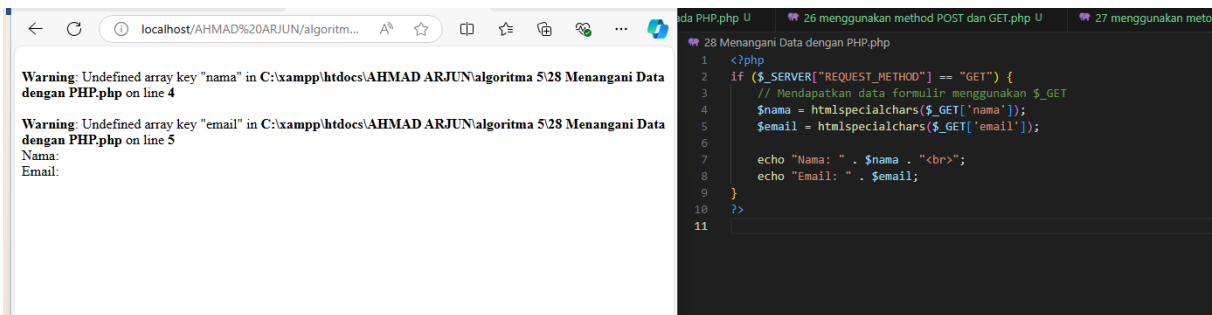
Pada contoh di atas, atribut method="get" pada elemen <form> menunjukkan bahwa metode pengiriman data yang digunakan adalah GET.

Menangani Data dengan PHP

Setelah pengguna mengisi formulir dan menekan tombol kirim, data akan dikirimkan ke file proses.php. Berikut contoh cara menangani data tersebut di PHP.

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    // Mendapatkan data formulir menggunakan $_GET
    $nama = htmlspecialchars($_GET['nama']);
    $email = htmlspecialchars($_GET['email']);

    echo "Nama: " . $nama . "<br>";
    echo "Email: " . $email;
}
?>
```



Dalam contoh kode di atas, kami menggunakan `$_GET` untuk mengakses data yang dikirim oleh formulir. Selain itu, kami menggunakan fungsi `htmlspecialchars` untuk menghindari serangan Cross-site Scripting (XSS).

Tips Keamanan

Meskipun metode GET mudah dan cepat, ingatlah bahwa semua informasi dikirim melalui URL. Itu berarti informasi tersebut bisa disimpan dalam history browser, log server, atau bahkan terlihat oleh orang lain yang mempunyai akses ke komputer pengguna. Gunakan metode POST untuk data yang lebih sensitif.

Dengan mengikuti langkah-langkah di atas, sekarang kamu sudah paham cara menggunakan formulir dengan metode GET di PHP. Penggunaan metode ini sangat cocok untuk kasus di mana keamanan data bukanlah prioritas utama dan kamu memerlukan kemudahan dalam melakukan debugging atau operasi pencarian sederhana.

POST FORM

Pelajari cara mengirimkan data formulir menggunakan metode POST di PHP dengan langkah-langkah mudah dan contoh kode praktis.

Mengirimkan data dari formulir web ke server merupakan salah satu kegiatan utama yang dilakukan dalam pembangunan aplikasi web. PHP, sebagai bahasa pemrograman server-side yang populer, menyediakan cara yang mudah untuk menangani data formulir yang dikirimkan oleh pengguna. Metode POST adalah salah satu metode yang sering digunakan untuk mengirim data formulir dengan cara yang aman.

Membuat Formulir Sederhana di HTML

Sebelum kita mengirim data menggunakan PHP, pertama-tama kamu perlu membuat sebuah formulir HTML. Berikut adalah contoh kode formulir yang bisa kamu gunakan:

```
<form action="handle-post.php" method="post">
  <label for="name">Nama:</label>
  <input type="text" id="name" name="name">

  <label for="email">Email:</label>
  <input type="email" id="email" name="email">

  <input type="submit" value="Kirim">
</form>
```

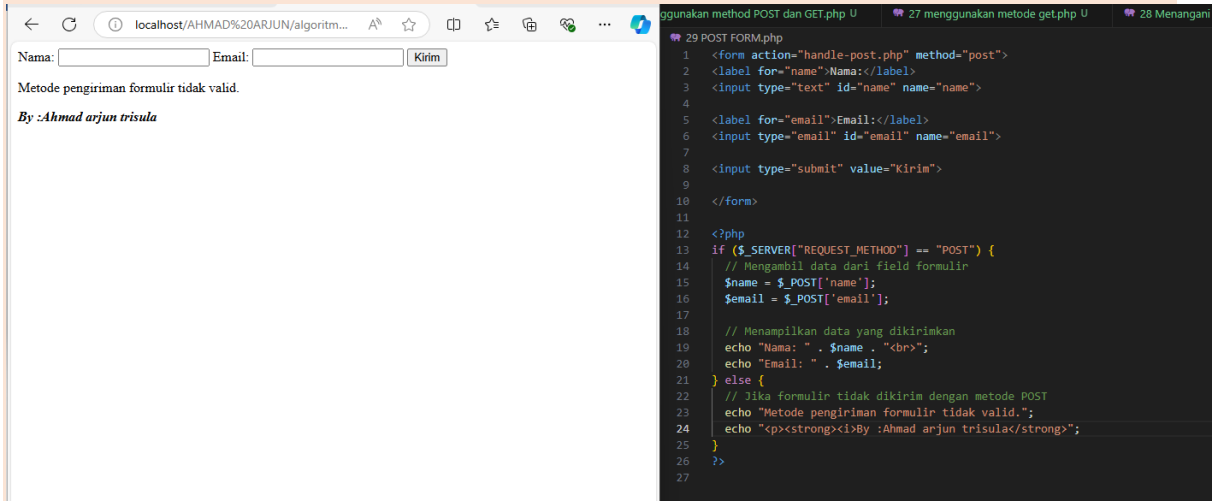
Perhatikan bahwa atribut `action` mengarahkan data formulir ke file `handle-post.php` yang akan kita buat nanti. Atribut `method` diatur ke `"post"`, ini berarti data formulir akan dikirim menggunakan metode POST.

Mengolah Data Formulir di PHP

Setelah formulir dikirimkan, kamu akan memerlukan skrip PHP untuk menangani dan memproses data tersebut. Buat file baru dengan nama `handle-post.php` dan masukkan kode berikut:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Mengambil data dari field formulir
    $name = $_POST['name'];
    $email = $_POST['email'];

    // Menampilkan data yang dikirimkan
    echo "Nama: " . $name . "<br>";
    echo "Email: " . $email;
} else {
    // Jika formulir tidak dikirim dengan metode POST
    echo "Metode pengiriman formulir tidak valid.";
}
?>
```



Memeriksa Metode Pengiriman Formulir

Kode di atas pertama-tama memeriksa apakah formulir telah dikirimkan menggunakan metode POST dengan mengecek isi dari `$_SERVER["REQUEST_METHOD"]`. Ini penting untuk menghindari akses langsung ke skrip PHP tanpa melalui formulir.

Mengakses Data Formulir

Setelah kita yakin bahwa data dikirim dengan metode POST, kita bisa mengakses data yang dikirimkan menggunakan superglobal `$_POST` di PHP. Untuk setiap input formulir, gunakan atribut `name` untuk mengambil datanya. Misalnya, untuk mengambil data dari input dengan `name="name"`, gunakan `$_POST['name']`.

Menampilkan Data yang Dikirim

Setelah data diambil, contoh di atas menampilkan data tersebut dengan menggunakan `echo`. Ini akan menunjukkan kepada pengguna data yang mereka kirimkan ke server.

Dengan mengikuti langkah-langkah di atas dan menggabungkan kode HTML dan PHP, kamu telah berhasil membuat formulir yang bisa mengirimkan data menggunakan metode POST. Kamu bisa mengembangkan skrip PHP lebih lanjut untuk memvalidasi, menyimpan, atau memproses data formulir sesuai dengan kebutuhan aplikasi web yang kamu buat.

REQUIRED FORM

Pelajari cara membuat form wajib di PHP beserta validasi dan pengambilan data dari form dengan mudah melalui panduan langkah demi langkah.

Dalam pembuatan aplikasi web, formulir adalah komponen penting yang sering digunakan untuk mengumpulkan informasi dari pengguna. PHP menyediakan cara yang efisien untuk mengelola data formulir, termasuk validasi untuk memastikan bahwa data yang diperlukan telah terisi. Berikut ini adalah panduan membuat form wajib (required form) menggunakan PHP.

Membuat Form HTML

Sebelum menggunakan PHP, kita perlu membuat form HTML terlebih dahulu. Berikut adalah contoh sederhana formulir dengan beberapa elemen input yang harus diisi pengguna.

```
<form method="post" action="proses_form.php">
  Nama: <input type="text" name="nama" required>
  Email: <input type="email" name="email" required>
  <input type="submit" value="Kirim">
</form>
```

Atribut required dalam elemen input menandakan bahwa field tersebut wajib diisi sebelum form dapat dikirim.

Mengelola Data dengan PHP

Setelah formulir diisi dan dikirim, kita perlu mengambil dan mengelola data tersebut dengan PHP di proses_form.php.

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Cek jika field nama dan email telah terisi
    if (empty($_POST["nama"])) {
        echo "Nama harus diisi.";
    } else {
        $nama = $_POST["nama"];
        // Lanjutkan dengan proses $nama
    }

    if (empty($_POST["email"])) {
        echo "Email harus diisi.";
    } else {
        $email = $_POST["email"];
        // Lanjutkan dengan proses $email
    }
}
?>
```

Validasi Data Form

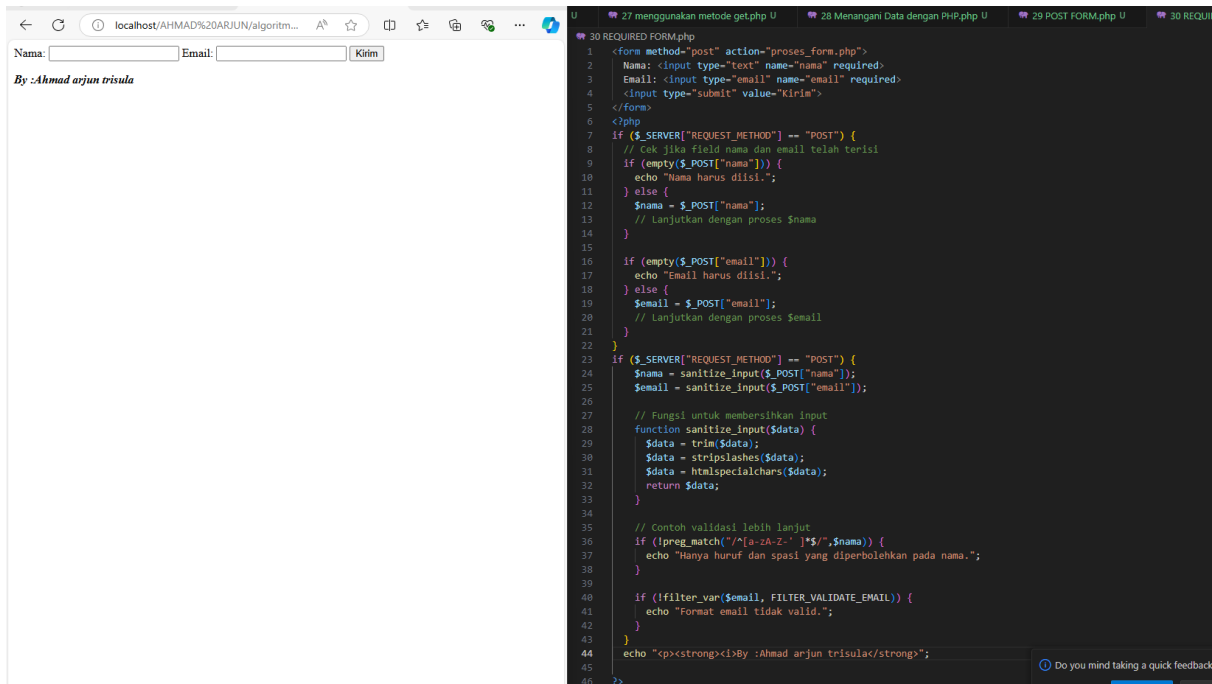
Kita juga perlu menambahkan validasi untuk meningkatkan keamanan dan integritas data.

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $nama = sanitize_input($_POST["nama"]);
    $email = sanitize_input($_POST["email"]);

    // Fungsi untuk membersihkan input
    function sanitize_input($data) {
        $data = trim($data);
        $data = stripslashes($data);
        $data = htmlspecialchars($data);
        return $data;
    }

    // Contoh validasi lebih lanjut
    if (!preg_match("/^[a-zA-Z-' ]*$/", $nama)) {
        echo "Hanya huruf dan spasi yang diperbolehkan pada nama.";
    }

    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Format email tidak valid.";
    }
}
?>
```



Dengan mengikuti langkah-langkah ini, kamu sudah bisa membuat form yang wajib diisi dengan validasi sederhana menggunakan PHP. Selamat mencoba dan teruslah belajar untuk meningkatkan keamanan dan fitur pada aplikasi webmu.

FORM URL

Pelajari cara membentuk URL dalam PHP untuk pengiriman data formulir dengan contoh kode sederhana.

Mengirimkan data dari formulir web ke server adalah salah satu fungsi dasar yang sering dibutuhkan dalam pembuatan website. PHP menyediakan cara sederhana untuk menangani proses ini. Dalam artikel ini, kamu akan belajar bagaimana membentuk URL untuk mengirimkan data formulir menggunakan metode GET dan POST di PHP.

Memahami Metode Pengiriman Formulir

Sebelum kita membahas pembentukan URL, penting untuk memahami dua metode pengiriman formulir utama: GET dan POST.

Metode GET

Metode GET mengirimkan data formulir melalui URL. Data yang dikirimkan dapat dilihat pada address bar browser. Hal ini berguna untuk operasi pencarian atau setiap request yang tidak memerlukan keamanan data yang tinggi.

Metode POST

Berbeda dengan GET, metode POST mengirimkan data melalui HTTP message body, tidak melalui URL. Ini menjadikan POST pilihan yang lebih aman untuk mengirimkan data sensitif, seperti password atau informasi pribadi.

Membentuk URL dengan Metode GET

Ketika menggunakan metode GET, kamu dapat membentuk URL secara langsung dalam atribut action pada tag <form>. Misalnya, jika kamu memiliki formulir pencarian sederhana:

```
<form action="search.php" method="get">
  Cari: <input type="text" name="query">
  <input type="submit" value="Search">
</form>
```

Saat formulir tersebut disubmit, PHP akan menyertakan input field query sebagai parameter URL. Sebagai contoh, jika kamu mencari kata “buku”, URL yang terbentuk akan seperti:

search.php?query=buku

Mengolah Data Formulir dengan PHP

Kamu perlu script PHP untuk menyaring dan mengolah data yang dikirimkan. Gunakan variabel superglobal \$_GET atau \$_POST untuk mengakses data ini.

Untuk Metode GET

```
if(isset($_GET['query'])) {  
    $searchTerm = $_GET['query'];  
    // Proses searchTerm untuk menampilkan hasil pencarian  
}
```

Untuk Metode POST

```
if(isset($_POST['username']) && isset($_POST['password'])) {  
    $username = $_POST['username'];  
    $password = $_POST['password'];  
    // Proses username dan password untuk autentikasi  
}
```

Penanganan Karakter Khusus

Karakter khusus dalam URL harus di-encode untuk mencegah ambiguitas atau masalah parsing. PHP memiliki fungsi urlencode() untuk mengatasi hal ini.

Contoh penggunaan:

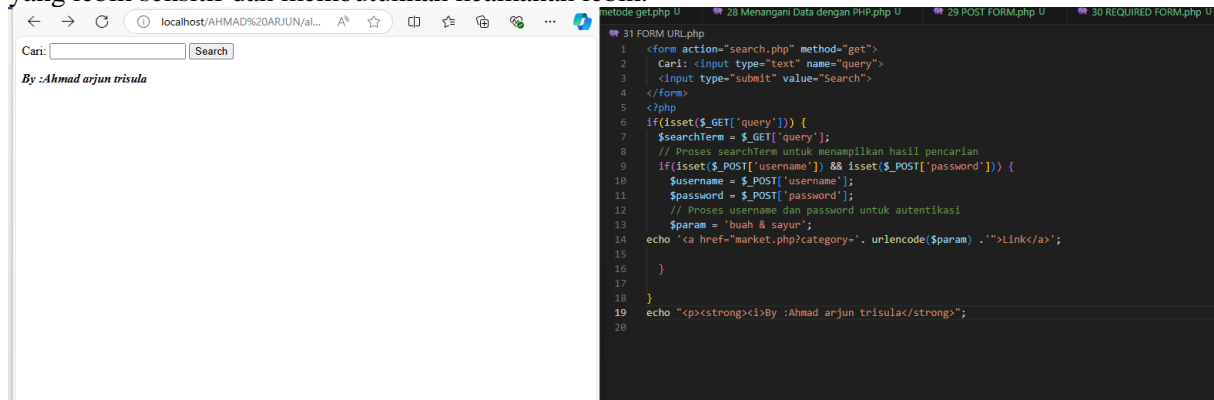
```
$param = 'buah & sayur';  
echo '<a href="market.php?category='. urlencode($param) .'">Link</a>';
```

Link yang akan dihasilkan akan meng-encode & menjadi %26, sehingga tidak akan rusak saat dikirim:

market.php?category=buah+%26+sayur

Kesimpulan

Dengan memahami cara membentuk URL dan mengirimkan data formulir di PHP, kamu bisa membuat interaksi dengan pengguna di website kamu menjadi lebih interaktif dan dinamis. Ingat, gunakan metode yang sesuai dengan kebutuhan: GET untuk operasi sederhana dan POST untuk data yang lebih sensitif dan membutuhkan keamanan lebih.



FORM EMAIL

Pelajari cara mengirim email menggunakan form di PHP dengan panduan langkah demi langkah yang mudah dan praktis untuk pemula.

Mengirim email melalui form adalah cara yang efektif untuk berkomunikasi dengan pengguna website kamu. PHP menyediakan fitur yang memungkinkan untuk mengirim email langsung dari website. Dalam tutorial ini, kamu akan belajar bagaimana membuat form dan mengirim email menggunakan script PHP.

Memulai dengan HTML Form

Untuk mengirim email, pertama-tama kita perlu membuat form HTML di mana pengguna dapat memasukkan datanya. Berikut adalah contoh sederhana dari form HTML:

```
<form method="post" action="sendmail.php">
  <label for="name">Nama:</label>
  <input type="text" id="name" name="name">

  <label for="email">Email:</label>
  <input type="email" id="email" name="email">

  <label for="message">Pesan:</label>
  <textarea id="message" name="message"></textarea>

  <input type="submit" value="Kirim">
</form>
```

Menangani Form dengan PHP

Setelah pengguna mengisi form dan menekan tombol “Kirim”, data form akan dikirim ke sendmail.php. Di sini kita akan menulis script PHP untuk mengolah data tersebut.

Memastikan Metode Request adalah POST

Sebelum memproses form, pastikan bahwa form tersebut dikirim menggunakan metode POST.

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Lanjutkan dengan mengolah data form
}
```

Mengambil Data dari Form

Kamu bisa mengambil data dari form menggunakan variabel \$_POST.

```
$name = $_POST['name'];
$email = $_POST['email'];
$message = $_POST['message'];
```

Validasi Data

Sebelum mengirim email, lakukan validasi sederhana untuk memastikan bahwa data yang dimasukkan valid.

```
if (empty($name) || empty($email) || empty($message)) {
    echo "Semua bidang harus diisi.";
    exit;
}

if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Email tidak valid.";
    exit;
}
```

Mengirim Email dengan PHP

PHP memiliki fungsi mail() yang digunakan untuk mengirim email. Berikut adalah contoh penggunaan fungsi tersebut:

```
$to = 'tujuan@example.com'; // Ganti dengan alamat email tujuan
$subject = 'Pesan dari Website';
$headers = "From: " . $email . "\r\n";
$headers .= "Reply-To: " . $email . "\r\n";
$headers .= "X-Mailer: PHP/" . phpversion();

if (mail($to, $subject, $message, $headers)) {
    echo "Pesan terkirim dengan sukses.";
} else {
    echo "Pesan gagal dikirim.";
}
```

Memformat Pesan Email

Untuk memformat pesan email, kamu dapat menambahkan headers tambahan atau menggunakan HTML dalam pesan. Pastikan untuk menetapkan Content-Type di headers jika menggunakan HTML.

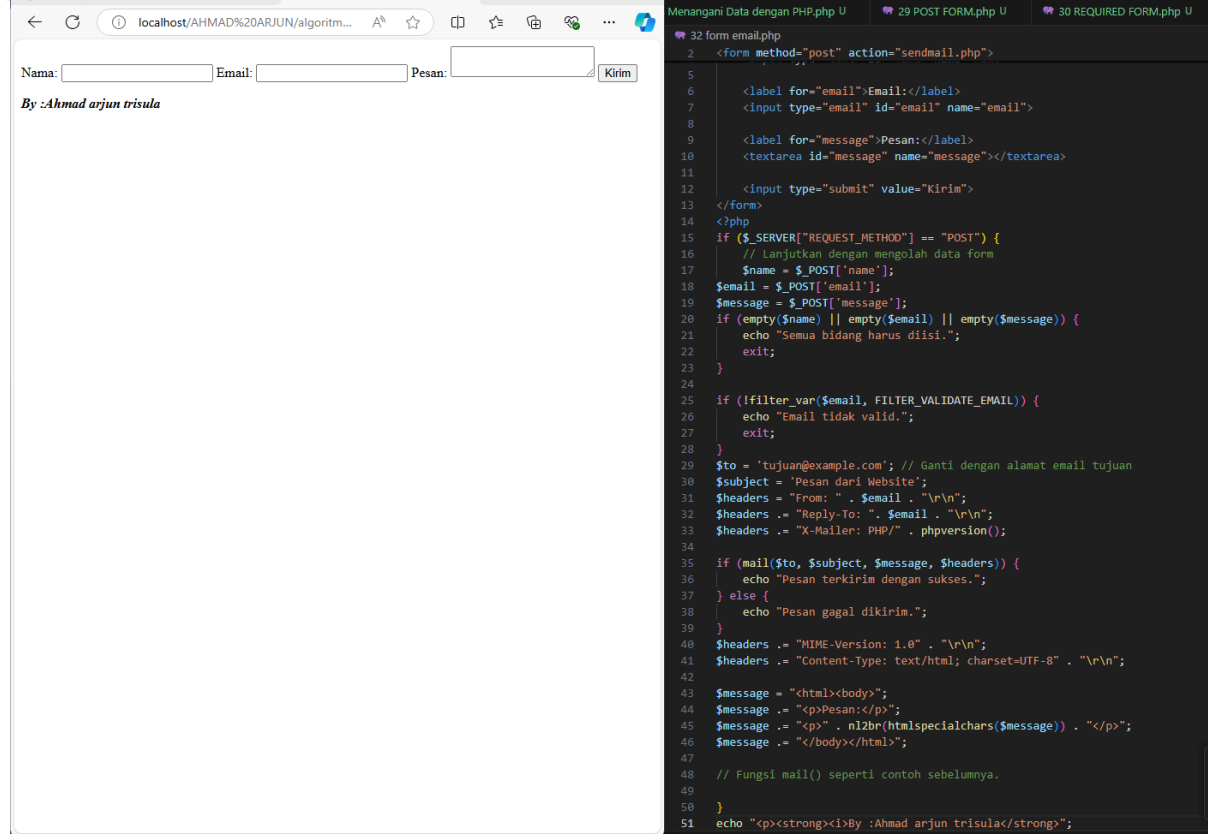

```

$headers .= "MIME-Version: 1.0" . "\r\n";
$headers .= "Content-Type: text/html; charset=UTF-8" . "\r\n";

$message = "<html><body>";
$message .= "<p>Pesan:</p>";
$message .= "<p>" . nl2br(htmlspecialchars($message)) . "</p>";
$message .= "</body></html>";

```

// Fungsi mail() seperti contoh sebelumnya.



The image shows a web browser on the left and a code editor on the right. The browser displays a simple email form with fields for 'Nama:', 'Email:', and 'Pesan:', followed by a 'Kirim' button. Below the form, it says 'By :Ahmad arjun trisula'. The code editor shows the PHP script for 'form_email.php'. The script handles the form submission, validates the email address, and sends the message using the mail() function. The email body is constructed using the form data and includes a signature.

```

32 form_email.php
2 <form method="post" action="sendmail.php">
5
6 <label for="email">Email:</label>
7 <input type="email" id="email" name="email">
8
9 <label for="message">Pesan:</label>
10 <textarea id="message" name="message"></textarea>
11
12 <input type="submit" value="Kirim">
13 </form>
14 <?php
15 if ($_SERVER["REQUEST_METHOD"] == "POST") {
16     // Lanjutkan dengan mengolah data form
17     $name = $_POST['name'];
18     $email = $_POST['email'];
19     $message = $_POST['message'];
20     if (empty($name) || empty($email) || empty($message)) {
21         echo "Semua bidang harus diisi.";
22         exit;
23     }
24
25     if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
26         echo "Email tidak valid.";
27         exit;
28     }
29
30     $to = 'tujuan@example.com'; // Ganti dengan alamat email tujuan
31     $subject = 'Pesan dari Website';
32     $headers = "From: " . $email . "\r\n";
33     $headers .= "Reply-To: " . $email . "\r\n";
34     $headers .= "X-Mailer: PHP/" . phpversion();
35
36     if (mail($to, $subject, $message, $headers)) {
37         echo "Pesan terkirim dengan sukses.";
38     } else {
39         echo "Pesan gagal dikirim.";
40     }
41
42     $headers .= "MIME-Version: 1.0" . "\r\n";
43     $headers .= "Content-Type: text/html; charset=UTF-8" . "\r\n";
44
45     $message = "<html><body>";
46     $message .= "<p>Pesan:</p>";
47     $message .= "<p>" . nl2br(htmlspecialchars($message)) . "</p>";
48     $message .= "</body></html>";
49
50     // Fungsi mail() seperti contoh sebelumnya.
51 }
52 echo "<p><strong><i>By :Ahmad arjun trisula</i></strong>";

```

Dengan mengikuti langkah-langkah dalam tutorial ini, kamu sekarang dapat membuat form di website yang akan mengirim email melalui PHP. Ingatlah untuk selalu memvalidasi dan membersihkan data yang dikirim pengguna untuk menghindari risiko keamanan, seperti serangan injection. Selamat mencoba!

COOKIES PHP

Pelajari cara menggunakan cookies dalam PHP untuk menyimpan data pada browser pengguna, meliputi cara membuat, mengambil, dan menghapus cookies dengan kode PHP yang sederhana.

Cookies merupakan cara sederhana untuk menyimpan data di sisi client dan bisa digunakan di dalam PHP. Cookies sangat berguna untuk berbagai hal, seperti menyimpan preferensi pengguna atau data login sehingga pengguna tidak perlu memasukkannya setiap kali mengunjungi situs web.

Memahami Cookies

Cookies adalah file kecil yang disimpan di komputer pengguna oleh situs web yang dikunjungi. Cookies berisi informasi yang dapat dibaca oleh web server di masa mendatang. Informasi ini bisa berupa pengaturan, token sesi, atau data lain yang ingin diingat oleh situs web tentang pengguna.

Membuat Cookies di PHP

Untuk membuat cookie dalam PHP, kamu bisa menggunakan fungsi `setcookie()`. Cookie harus dibuat sebelum tag HTML dikirim ke browser, yaitu sebelum ada output apapun.

`setcookie(name, value, expire, path, domain, secure, httponly);`

name: Nama cookie

value: Nilai dari cookie

expire: Waktu kedaluwarsa cookie

path: Path di server di mana cookie akan tersedia

domain: Domain dimana cookie berlaku

secure: Apabila diset true, cookie hanya akan di-set melalui koneksi HTTPS

httponly: Jika diset true, maka cookie hanya akan diakses melalui protokol HTTP

Contoh Membuat Cookie

```
// Membuat cookie yang berlaku selama satu jam
setcookie("user", "nama_pengguna", time() + (3600), "/");
```

Mengambil Nilai Cookie

Setelah cookie dibuat, nilai yang disimpan dalam cookie dapat diakses melalui array superglobal `$_COOKIE`.

Contoh Mengambil Nilai Cookie

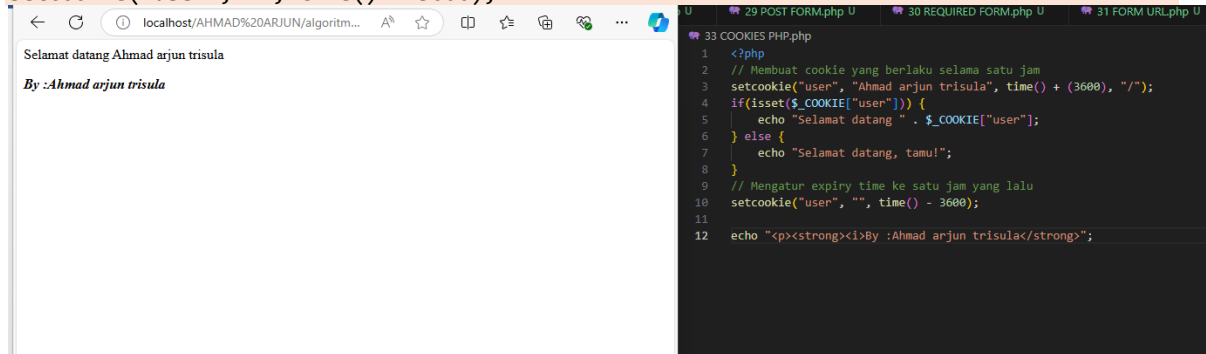
```
if(isset($_COOKIE["user"])) {
    echo "Selamat datang " . $_COOKIE["user"];
} else {
    echo "Selamat datang, tamu!";
}
```

Menghapus Cookies

Kamu bisa menghapus cookie dengan menyetel waktu kedaluwarsa ke waktu yang sudah lewat.

Contoh Menghapus Cookie

```
// Mengatur expiry time ke satu jam yang lalu
setcookie("user", "", time() - 3600);
```



Dengan menggunakan contoh kode di atas, kamu sudah bisa membuat, mengambil, dan menghapus cookies di PHP. Ingatlah bahwa mengelola cookies dengan benar sangat penting untuk privasi pengguna dan keamanan data.

SESSIONS PHP

Panduan penggunaan sesi dalam PHP untuk pemeliharaan data antarsesi, termasuk cara memulai, menyimpan, dan menghapus data sesi.

Sesi adalah cara untuk menyimpan informasi (dalam variabel) yang bisa digunakan di berbagai halaman. Tanpa sesi, aplikasi web tidak akan dapat menyimpan informasi tentang pengguna dari satu halaman ke halaman lainnya. PHP membuat penggunaan sesi menjadi sangat sederhana.

Memulai Sesi PHP

Sebelum kamu bisa menyimpan informasi ke dalam sesi, kamu harus terlebih dahulu memulai sesi. PHP menyediakan fungsi `session_start()` untuk ini. Fungsi ini harus diletakkan di awal skrip, sebelum ada output apapun ke browser.

```
<?php
// Memulai sesi
session_start();
?>
```

Pastikan bahwa kamu memanggil `session_start()` sebelum tag HTML atau echo apa pun.

Menyimpan Data ke Sesi

Setelah sesi dimulai, kamu bisa menyimpan data ke dalam variabel `$_SESSION`. Ini adalah array superglobal, yang berarti bisa diakses dari mana saja di skrip PHP.

```
<?php
session_start();
// Menyimpan data ke sesi
$_SESSION["namaPengguna"] = "Budi";
?>
```

Setelah kamu menyimpan data ke dalam sesi, data tersebut bisa diakses dari halaman lain yang juga memulai sesi.

Mengakses Data Sesi

Untuk mengakses data yang telah kamu simpan dalam sesi, gunakan kembali array `$_SESSION` setelah memulai sesi.

```
<?php
session_start();
// Mengakses data dari sesi
echo 'Selamat Datang, ' . $_SESSION["namaPengguna"] . '!';
?>
```

Menghapus Variabel Sesi

Jika kamu ingin menghapus variabel tertentu dari sesi, kamu bisa menggunakan `unset()`. Untuk menghapus semua variabel sesi, kamu bisa menggunakan `session_unset()`.

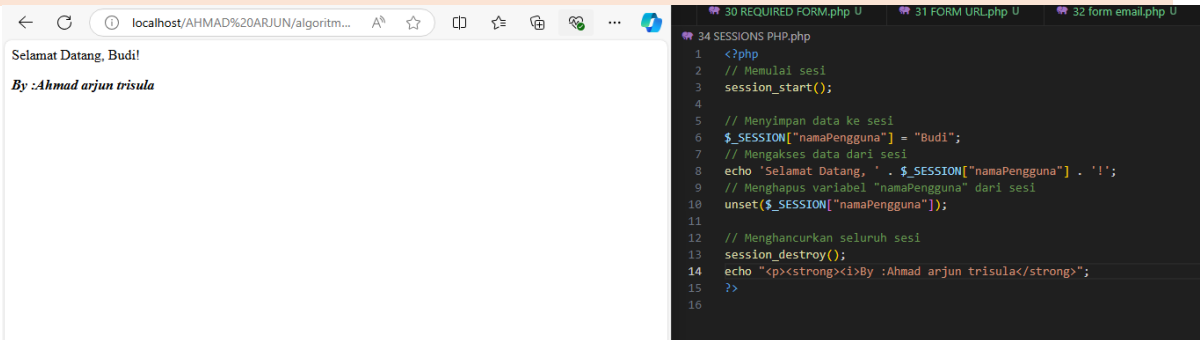
```
<?php
session_start();
// Menghapus variabel "namaPengguna" dari sesi
unset($_SESSION["namaPengguna"]);
?>
```

Menghancurkan Sesi

Untuk sepenuhnya menghancurkan sesi, termasuk data dan ID sesi, gunakan `session_destroy()`. Ini akan menghapus seluruh sesi yang aktif.

```
<?php
session_start();
```

```
// Menghancurkan seluruh sesi
session_destroy();
?>
```



Setelah sesi dihancurkan, ID sesi yang lama tidak akan valid. Jika kamu memulai sesi baru, ID sesi baru akan dibuat.

Kesimpulan

Menggunakan sesi dalam PHP sangatlah penting untuk membuat aplikasi yang dinamis dan bisa menyimpan data pengguna. Dengan fungsi-fungsi yang telah disediakan PHP, kamu bisa dengan mudah memulai, menyimpan, mengakses, dan menghancurkan data sesi. Ingatlah untuk selalu memulai skrip kamu dengan `session_start()` sebelum melakukan operasi apapun pada sesi.

COOKIE VS SESSION PHP

Perbedaan utama antara cookie dan sesi di PHP, bagaimana mereka bekerja, dan kapan harus menggunakan masing-masing.

Mengelola data pengguna seperti preferensi dan status login adalah aspek penting dalam pengembangan web. Di PHP, ada dua cara utama untuk melakukan ini: menggunakan Cookie dan Sesi. Meskipun keduanya sering digunakan untuk tujuan yang serupa, mereka memiliki perbedaan penting dalam cara kerja dan keamanannya.

Pengertian Cookie

Cookie adalah file kecil yang disimpan di komputer pengguna oleh browser web. Cookie digunakan untuk menyimpan informasi seperti preferensi pengguna atau data login sehingga saat pengguna kembali ke situs web, situs tersebut dapat mengenali mereka dan menyesuaikan apa yang ditampilkan sesuai dengan data yang tersimpan.

Cara Kerja Cookie

Situs web mengirimkan cookie ke browser pengguna.

Browser menyimpan cookie sebagai file kecil di komputer pengguna.

Setiap kali pengguna kembali ke situs web, browser mengirimkan cookie kembali ke server.

Keuntungan Cookie

Persisten: Cookie dapat disimpan untuk jangka waktu yang lama, bahkan setelah browser ditutup.

Mudah diimplementasikan: Penggunaan cookie relatif sederhana di PHP.

Kelemahan Cookie

Kurang aman: Karena disimpan di sisi klien, cookie lebih rentan terhadap manipulasi dan pencurian data oleh pihak ketiga.

Ukuran terbatas: Cookie memiliki batasan ukuran, umumnya hingga 4KB per cookie.

Pengertian Sesi

Sesi adalah cara untuk menyimpan informasi di server daripada di komputer pengguna. Informasi sesi disimpan secara sementara di server dan dihapus begitu sesi pengguna berakhir atau setelah jangka waktu tertentu.

Cara Kerja Sesi

Ketika sesi dimulai, PHP akan membuat ID sesi unik.

ID sesi ini dikirim ke browser pengguna, biasanya dalam bentuk cookie.

Data sesi disimpan di server dengan ID sesi yang terkait.

Keuntungan Sesi

Lebih aman: Karena data disimpan di server, hal itu membuat sesi lebih terlindung dari akses tidak sah.

Ukuran besar: Sesi dapat menyimpan lebih banyak data dibandingkan dengan cookie.

Kelemahan Sesi

Kehilangan data: Jika browser ditutup atau sesi kadaluwarsa, data akan hilang kecuali disimpan ke database atau penyimpanan lain.

Menggunakan sumber daya server: Sesi membutuhkan penyimpanan di server, yang bisa menjadi masalah jika banyak pengguna yang aktif.

Perbandingan Cookie dan Sesi

Durasi Penyimpanan

Cookie dapat disimpan untuk waktu yang lama, sedangkan sesi hanya ada selama browser terbuka atau sampai durasi sesi berakhir.

Keamanan

Sesi dianggap lebih aman karena data disimpan di server dan tidak dapat diakses melalui browser pengguna secara langsung.

Penggunaan Sumber Daya

Cookie tidak menggunakan sumber daya server, sementara sesi membutuhkan ruang penyimpanan di server.

Penanganan Data

Cookie lebih baik untuk data yang tidak sensitif dan tidak berubah-ubah, seperti preferensi tampilan. Sesi lebih cocok untuk data sensitif seperti informasi login atau data transaksi.

Kapan Menggunakan Cookie

Gunakan cookie saat kamu ingin menyimpan informasi tidak sensitif di komputer pengguna untuk jangka waktu yang lama, seperti pengaturan personalisasi.

Kapan Menggunakan Sesi

Gunakan sesi untuk menangani data sensitif atau saat kamu perlu informasi yang harus hilang saat pengguna keluar dari situs atau sesi berakhir.

Dalam pengembangan web, mengetahui kapan menggunakan cookie atau sesi dapat membuat aplikasi kamu lebih efektif dan aman. Sesuaikan penggunaan keduanya berdasarkan kebutuhan fungsi dan keamanan dari situs yang kamu kembangkan.