

Live Session 4

Menulis Kode dengan Prinsip Sesuai Guidelines dan Best Practices

Vocational School Graduate Academy



Skema Pelatihan :
Junior Web Developer

<i>Output</i> Pelatihan	<p>Setelah mengikuti pelatihan ini, diharapkan peserta kompeten:</p> <ol style="list-style-type: none"> 1. Mengimplementasikan user interface 2. Menerapkan perintah eksekusi bahasa pemrograman berbasis teks, grafik, dan multimedia 3. Menyusun fungsi, file atau sumber daya pemrograman yang lain dalam organisasi yang rapi 4. Menulis kode dengan prinsip sesuai guidelines dan best practices 5. Mengimplementasikan pemrograman terstruktur 6. Menggunakan library atau komponen pre-existing
Jam Pelatihan	67 JP / 9 hari
Jenis Pelatihan	Daring (<i>Online</i>)

Profil Pengajar

Enny Indasyah, S.ST, MT., M.Sc.

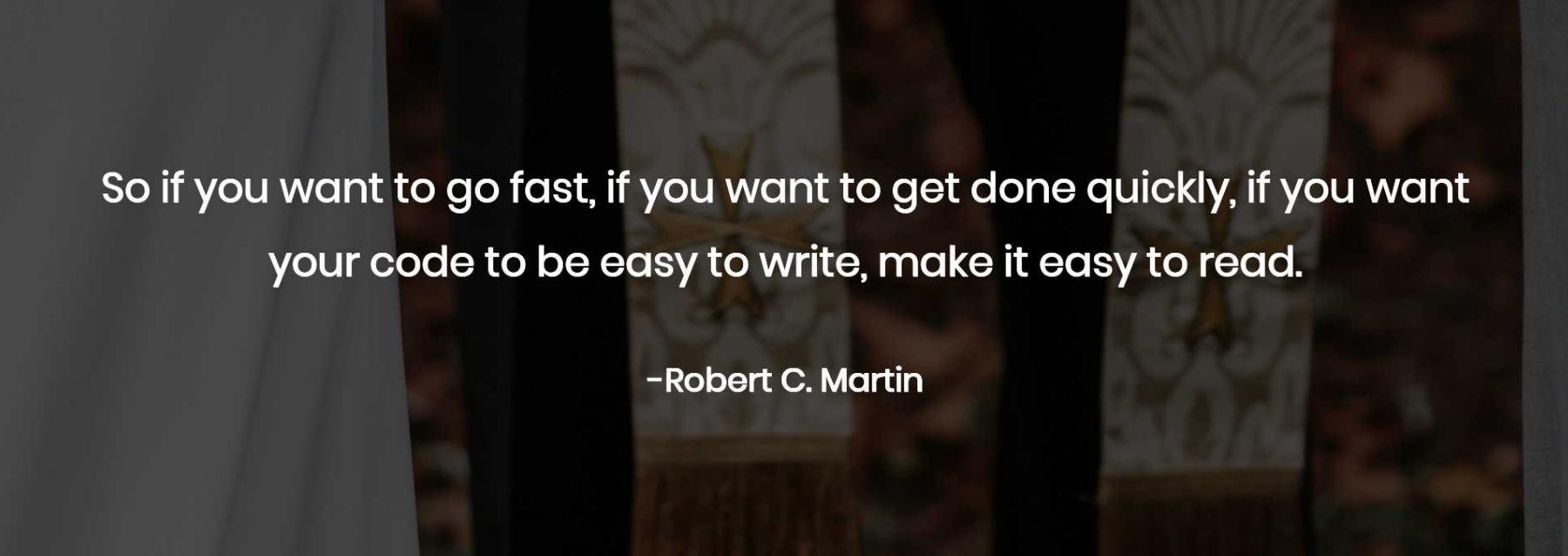
NIP : 199103302018072001

Email: enny_indasyah@its.ac.id

- **Jabatan Akademik :**
Dosen ITS
- **Riwayat Pendidikan :**
 - **Sarjana Terapan (S.ST) Teknik Telekomunikasi
Politeknik Elektronika Negeri Surabaya (PENS)**
 - **Master of Science (M.Sc) Department of Computer Science,
NTUST, Taiwan**
 - **Magister Teknik (M.T) Teknik Elektro – Jaringan Cerdas
Multimedia, ITS**



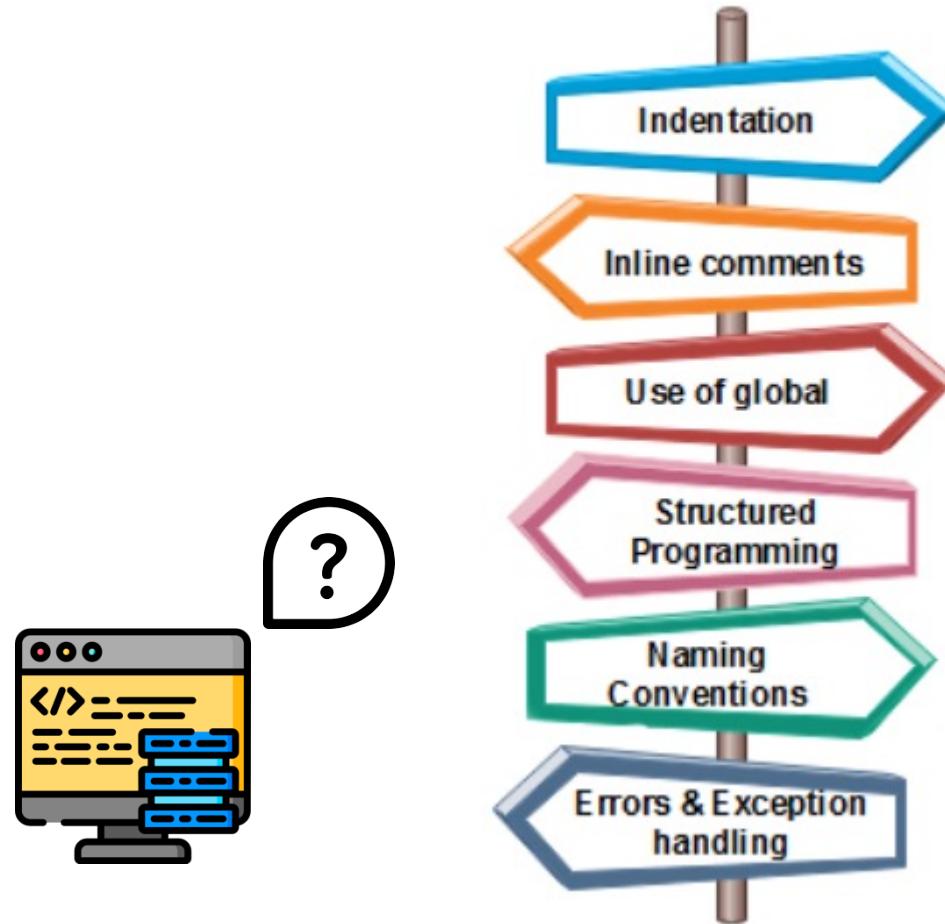
Motivasi



So if you want to go fast, if you want to get done quickly, if you want
your code to be easy to write, make it easy to read.

-Robert C. Martin

Apersepsi



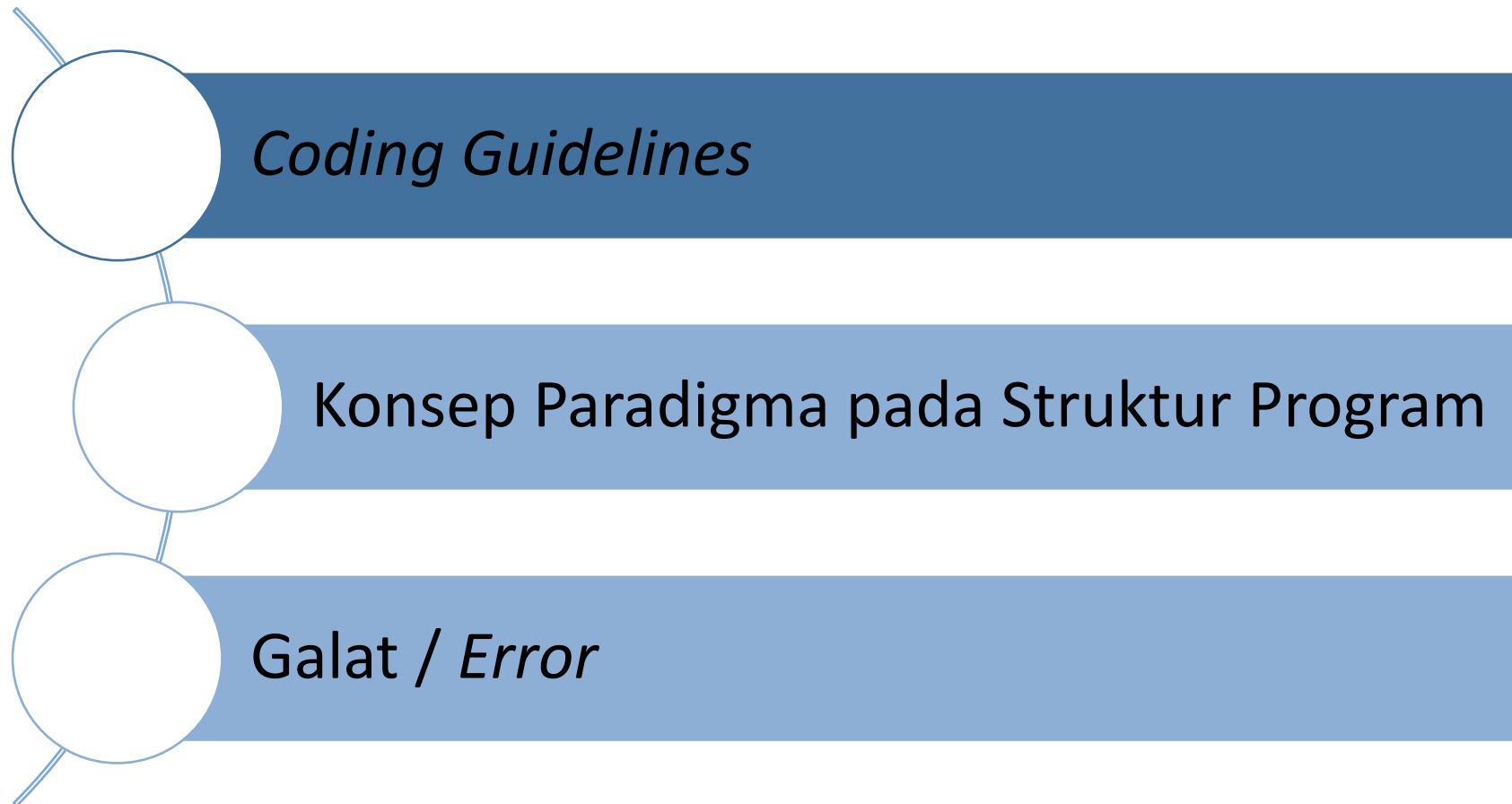
Sumber: <https://www.javatpoint.com/software-engineering-coding>

Menerapkan *Coding Guidelines* dan *Best Practices* dalam Penulisan Program (Kode Sumber)

Ringkasan Mata Pelatihan

- Unit Kompetensi Acuan: Menulis kode dengan prinsip sesuai *guidelines* dan *best practices*
- Kode Unit Kompetensi Acuan: J.620100.016.01
- Deskripsi singkat: Mata pelatihan ini menentukan kompetensi, pengetahuan dan sikap kerja yang diperlukan dalam menerapkan penulisan kode.
- Tujuan Pembelajaran: Peserta dapat menerapkan penulisan kode yang baik agar kode tersebut dapat dirawat (*Maintainability*).

Agenda



Coding Guidelines

Coding Guidelines

Coding Guidelines adalah acuan bagi *developer* untuk membuat kode program yang lebih mudah dibaca dan dipelihara.

Tujuan dari *coding guidelines*:

1. Menyeragamkan kode program yang dibuat oleh *developer* yang berbeda-beda
2. Mempermudah pemahaman isi kode program dan mengurangi kompleksitas program
3. Membantu kode program untuk bisa digunakan kembali (*reuse*)
4. Mempermudah mendeteksi kesalahan / *error*

Penamaan

Penamaan *local variable, global variable, constant, function* dan *method* sebaiknya memperhatikan:

- Mudah dipahami
- Menghindari penggunaan angka
- Mendeskripsikan isi dengan singkat dan jelas

Contoh Penamaan

- **Variable**

Variable menggunakan *camel case* dimana diawali dengan huruf kecil, sedangkan *global variable* diawali dengan kapital.

```
class FirstClass{
    //global variable
    InputProgram = 10;

    function testVariable()
    {
        //local variable
        inputTest = 1;
    }
}
```

Contoh Penamaan (2)

- ***Constant***

Constant menggunakan huruf kapital semua atau huruf kapital dengan garis bawah sebagai pengganti spasi

```
class FirstClass{  
    const INPUT = 100;  
    const TEST_VERSION = 2;  
}
```

Contoh Penamaan (3)

- ***Function***

Function menggunakan *camel case* dimana diawali dengan huruf kecil.

```
function luasSegitiga($tinggi, $alas)
{
    $luas = 0.5 * $alas * $tinggi;
    return $luas;
}
```

Indentation

Indentation sangat penting untuk mempermudah pembacaan kode program.
Beberapa bagian dari *indentation*:

- Menambahkan spasi setelah koma diantara dua *argument function*
- Setiap *nested block* harus dilakukan *indentation*
- *Indentation* dilakukan pada awal dan akhir setiap *block* program
- Setiap *block* program ditulis didalam tanda kurung kurawal {...}

Indentation (2)

```
<?php
    InputProgram = 1;

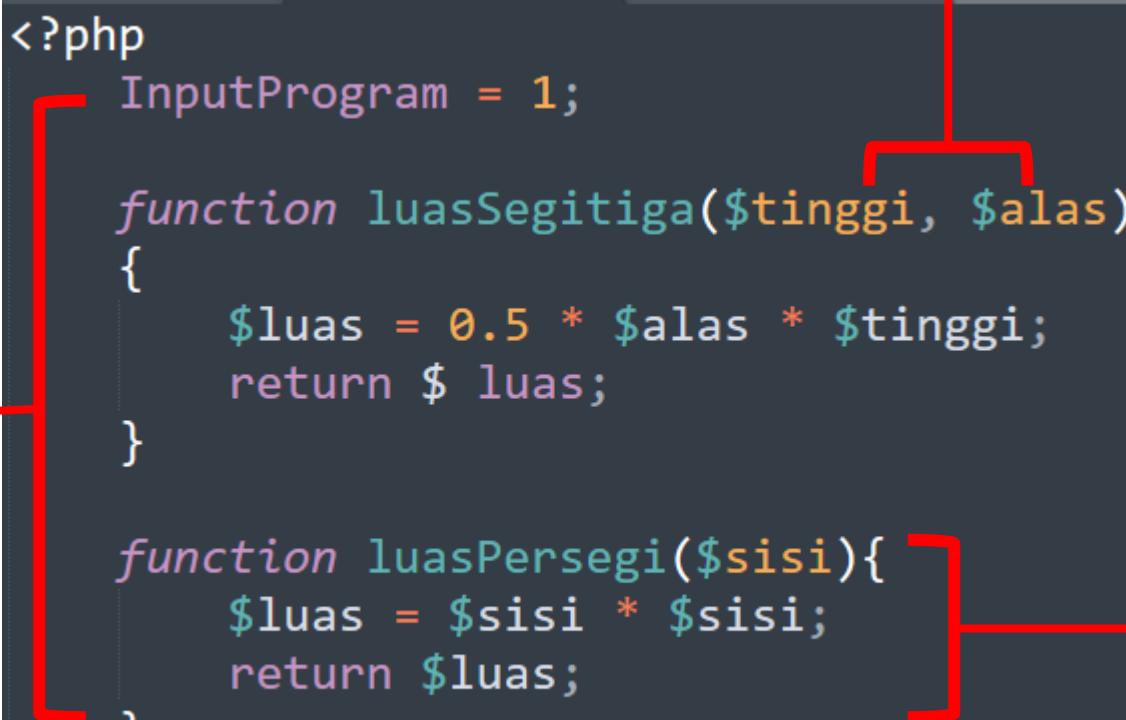
    function luasSegitiga($tinggi, $alas)
    {
        $luas = 0.5 * $alas * $tinggi;
        return $luas;
    }

    function luasPersegi($sisi){
        $luas = $sisi * $sisi;
        return $luas;
    }
?>
```

indentation ←

space ↑

block program dalam {...} →



Structured Programming

Structured atau *Modular Programming* harus digunakan dalam penulisan kode program. “**GO TO**” statements sebaiknya tidak digunakan untuk memudahkan pemahaman isi kode program.

Menghindari penggunaan *identifier* untuk kebutuhan yang berbeda

Setiap *variable* yang digunakan harus mendeskripsikan kegunaannya.

Penggunaan satu *variable* yang sama untuk beberapa tujuan yang berbeda harus dihindari untuk menghindari kesulitan di masa depan.

Kode Program Harus Didokumentasikan

Setiap kode program harus didokumentasikan dalam bentuk komen yang mudah dipahami.

Error Return Values

Setiap *function* dalam program ditangani dengan standard yang terorganisasi, misalnya setiap kesalahan (*error*) mengembalikan nilai 0 atau 1 untuk menyederhanakan *debugging*.

Coding Standard untuk PHP

Tag PHP

- Tag Pembuka HARUS berada pada barisnya sendiri dan HARUS diikuti oleh baris kosong.

```
< ? php print_welcome_message ();
```

Salah karena <?php tidak pada jalurnya sendiri.

```
<? php  
print_welcome_message ();
```

Salah karena <?php tidak diikuti oleh baris kosong

```
<? php  
print_welcome_message ();
```

BENAR

Tag Penutup

- Benar

Tidak menggunakan ?>

```
<? php  
print_welcome_message ();
```

- Salah

Salah karena ?> digunakan.

```
<? php  
print_welcome_message ();  
? >
```

- Tag Penutup TIDAK HARUS digunakan dalam file PHP, namun digunakan apabila ada kode PHP yang dituliskan dalam HTML (inline).

Tag Pembuka dan Penutup (inline PHP)

- Benar

Karena tag <?php dan ?> berada dalam satu baris

```
< div >
    < h1 > <? php print_welcome_message (); ?> </ h1 >
</ div >
```

- Salah

Salah karena tag <?php dan ?> tidak dalam satu baris

```
< div >
    < h1 > <? php
print_welcome_message ();
?> </ h1 >
</ div >
```

Tag Pembuka Pendek (short open tag)

- Tag pembuka pendek sebaiknya TIDAK digunakan.
- Benar
- Salah

```
<? php  
print_welcome_message ();
```

```
< ?  
print_welcome_message ();
```

Akhir dari File

- Bagian ini menjelaskan bagaimana setiap file PHP harus diakhiri.
- Cara penulisan komentar akhir file:
 - HARUS dimasukkan di akhir file
 - HARUS berada di jalurnya sendiri
 - HARUS didahului dan diakhiri oleh garis-garis kosong

```
<? php  
print_welcome_message ();  
// EOF
```

Namespace

- Deklarasi Namespace HARUS menjadi pernyataan pertama dan HARUS diikuti oleh baris kosong.
- Nama namespace HARUS dimulai dengan huruf kapital dan HARUS menjadi camelcase.
- Bila ada lebih dari satu namespace HARUS menggunakan sintaks kurung kurawal.
- Konstanta ajaib (magic constant) HARUS digunakan untuk referensi nama namespace.

Namespace (Contoh)

```
<? php

namespace MyCompany \ Model {
// ModuleOne body }

namespace MyCompany \ Lihat {
$ welcome_message = __NAMESPACE__ . ' \\ ' . get_welcome_message (); }

// EOF
```

Komentar

- Komentar satu baris HARUS menggunakan dua garis miring ke depan (//).
- Komentar multi-baris HARUS menggunakan format blokir (/*.....*/)
- Komentar header HARUS menggunakan format blokir.
- Komentar pembagi HARUS menggunakan format blok dengan 75 tanda bintang di antaranya.
- Komentar HARUS ada di jalur mereka sendiri.
- Blok kode HARUS dijelaskan atau dirangkum.
- Angka ambigu HARUS diklarifikasi.
- Variabel eksternal HARUS diklarifikasi.

Formatting

- Panjang baris tidak boleh melebihi 80 karakter, kecuali baris tersebut berupa text.
- Indentasi baris harus menggunakan tab.
- Baris kosong (dua kali enter) harus ditambahkan diantara dua kode logical block
- Meratakan teks harus menggunakan spasi
- Dua spasi kosong tidak boleh digunakan setelah pernyataan (statement) atau dua koma atau baris baru.
- Kata kunci (keyword) harus menggunakan huruf kecil (lowercase)

Formatting (2)

- Penamaan variabel harus menggunakan huruf kecil dan kata-kata harus dipisahkan oleh garis bawah.
- Variabel global harus dinyatakan satu variabel per baris dan harus diindentasi setelah yang pertama.
- Konstanta HARUS menggunakan huruf besar semua dan kata-kata HARUS dipisahkan oleh garis bawah.
- Pernyataan/ Statement HARUS ditempatkan pada baris sendiri dan HARUS diakhiri dengan tanda titik koma.

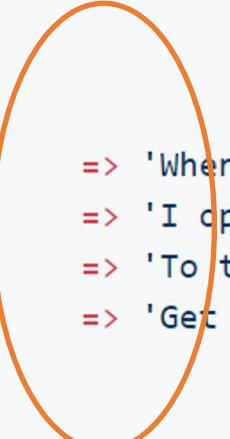
Formatting (3)

Contoh penulisan kode yang salah, karena perataan text harus menggunakan spasi, bukan tab

```
<?php

$movie_quotes = array(
    'slumdog_millionaire'
    'silver_linings_playbook'
    'the_lives_of_others'
    'the_shawshank_redemption'
);

// EOF
```



=> 'When somebody asks me a question, I tell them the answer.',
=> 'I opened up to you, and you judged me.',
=> 'To think that people like you ruled a country.',
=> 'Get busy living, or get busy dying.'

Formatting (4)

Contoh penulisan kode yang salah, karena indentasi text harus menggunakan tab, bukan spasi

```
<?php  
  
$movie_quotes = array(  
    'slumdog_millionaire'      => 'When somebody asks me a question, I tell them the answer.',  
    'silver_linings_playbook'   => 'I opened up to you, and you judged me.',  
    'the_lives_of_others'       => 'To think that people like you ruled a country.',  
    'the_shawshank_redemption' => 'Get busy living, or get busy dying.'  
);  
  
// EOF
```

Formatting (5)

Contoh penulisan kode yang benar, karena indentasi text menggunakan tab dan perataan text menggunakan spasi.

```
<?php

$movie_quotes = array(
    'slumdog_millionaire' => 'When somebody asks me a question, I tell them the answer.',
    'silver_linings_playbook' => 'I opened up to you, and you judged me.',
    'the_lives_of_others' => 'To think that people like you ruled a country.',
    'the_shawshank_redemption' => 'Get busy living, or get busy dying.'
);

// EOF
```

Fungsi

- Nama fungsi harus menggunakan huruf kecil dan kata-kata harus dipisahkan oleh garis bawah
Contoh: function welcome_message() {
- Awalan fungsi HARUS dimulai dengan kata kerja
Contoh: get_, add_, update_, delete_, convert_
- Pemanggilan fungsi tidak boleh menggunakan spasi antara nama fungsi dan tanda kurung terbuka
Contoh: func();

Fungsi (2)

Argumen didalam fungsi harus dituliskan sbb:

- Tidak boleh menggunakan spasi sebelum tanda koma.
- Harus menggunakan spasi setelah tanda koma.
- Boleh menggunakan baris baru untuk argument yang Panjang.
- Setiap argument harus dituliskan di baris yang berbeda
- Harus menggunakan satu kali indentasi untuk argument dibawahnya
- Harus diurutkan dari yang paling dibutuhkan sampai argument pilihan.
- Harus diurutkan berdasarkan argumen yang paling penting ke argument yang tidak begitu penting.

Function Return

Sebisa mungkin segera ditampilkan.

Variable yang menghasilkan return value Harus diinisialisasikan sebelumnya. HARUS diawali dengan baris baru, kecuali apabila function return berada didalam control statement.

```
<?php  
  
function get_movies() {  
    $movies = array();  
    // ...  
  
    return $movies;  
}  
  
// EOF
```

Struktur Kontrol

- Kata kunci HARUS diikuti oleh spasi misalnya if (, switch (, do {,for (
- Setelah tanda kurung buka tidak boleh diikuti oleh spasi misalnya (\$expr,(\$i
- Tanda kurung tutup tidak boleh diawali dengan spasi misalnya \$expr), \$i++),\$value)
- Tanda Kurung Kurawal buka HARUS diawali dengan spasi dan HARUS diikuti oleh baris baru misalnya \$expr) { ,\$i++) {

Struktur Kontrol (2)

- Struktur tubuh HARUS di-indentasi sekali dan HARUS ditutup dengan kurung kurawal (tidak ada tulisan cepat)
misalnya if (\$expr) { ↪ → ... ↪ }
- Penutupan brace HARUS dimulai pada baris berikutnya
yaitu ... ↪ }
- Struktur control yang bersarang tidak boleh melebihi tiga level
misalnya if (\$expr1) { if (\$expr2) { if (\$expr3) { if (\$expr4) { ... }}}}}

If, Elseif, Else

- Biasakan menggunakan elseif sebagai pengganti else if
- elseif dan else HARUS dituliskan diantara } dan { pada satu baris

```
if ( $ expr1 ) {  
// if body } elseif ( $ expr2 ) { // elseif body } else { // else body }
```

Switch, Case

- Case statement HARUS di indentasi sekali
- Badan case HARUS di indentasi dua kali
- Break kata kunci HARUS di indentasi dua kali
- Case logic HARUS dipisahkan oleh satu baris kosong

```
<?php

switch ($expr) {
    case 0:
        echo 'First case, with a break';
        break;

    case 1:
        echo 'Second case, which falls through';
        // no break
    case 2:
    case 3:
    case 4:
        echo 'Third case, return instead of break';
        return;

    default:
        echo 'Default case';
        break;
}

// EOF
```

For, foreach

```
<?php

    for ($i = 0; $i < 10; $i++) {
        // for body
    }

    foreach ($iterable as $key => $value) {
        // foreach body
    }

    // EOF
```

Try, catch

```
<?php

try {
    // try body
}
catch (FirstExceptionType $e) {
    // catch body
}
catch (OtherExceptionType $e) {
    // catch body
}

// EOF
```

Class

- Pemberian nama file untuk class hanya boleh berisi satu definisi dan HARUS diawali dengan kata kunci class-
Contoh: class User → class- user.php,
class Office → class-office.php
- Class namespace HARUS didefinisikan dan HARUS menyertakan nama vendor
Contohnya: namespace MyCompany\Model;
namespace MyCompany\View; ,namespace MyCompany\Controller;

```
<?php  
  
namespace MyCompany\Model;  
  
class User  
{  
    // ...  
}  
  
// EOF
```

Class (2)

- Nama class HARUS dimulai dengan huruf kapital dan apabila terdiri dari 2 kata HARUS menjadi camelcase
Contoh: MyCompany
- Harus ada dokumentasi class dan HARUS menggunakan model tag phpDocumentor
Contoh: @author, @global,@package
- Definisi class HARUS menempatkan kurung kurawal pada baris tersendiri
Contoh: class User { ... }

Properti dari Class

- Penamaan variable harus mengikuti standard yang ada.
- HARUS menetapkan visibilitas dari variable.
- Nama variable tidak boleh diawali dengan garis bawah jika tipe variable private atau protected.

```
<?php  
  
namespace MyCompany\Model;  
  
class User  
{  
    public $var1;  
    protected $var2;  
    private $var3;  
}  
  
// EOF
```

Properti dari Method

- Penamaan method harus mengikuti standard penulisan fungsi.
- HARUS menetapkan visibilitas dari method.
- Nama method tidak boleh diawali dengan garis bawah jika tipe method private atau protected.

```
<?php

namespace MyCompany\Model;

class User
{
    // ...

    public function get_var1() {
        return $this->var1;
    }

    protected function get_var2() {
        return $this->var2;
    }

    private function get_var3() {
        return $this->var3;
    }
}

// EOF
```

Instansiasi Sebuah Class

- HARUS mengikuti penamaan variable yang standard
- HARUS menyertakan tanda kurung

```
<?php  
  
$office_program = new OfficeProgram();  
  
// EOF
```

Konsep Paradigma pada Struktur Program

Paradigma Pemrograman

Paradigma pemrograman adalah cara untuk mengklasifikasikan kode program berdasarkan fitur program yang dibuat.

Jenis-jenis paradigma pemrograman:

1. *Procedural Programming*
2. *Logical Programming*
3. *Functional Programming*
4. *Object-Oriented Programming*

Procedural Programming

Procedural programming merupakan pradigma pemrograman berdasarkan konsep bagaimana prosedur dipanggil, dimana kode program disusun dalam bentuk *list* instruksi yang menjelaskan tahap-tahapan yang harus dikerjakan.

Kelebihan *procedural programming*:

- Kode program *portable*.
- Program dapat digunakan kembali pada program lain tanpa perlu menyalinnya (*copy*).
- Alur program dapat ditelusuri dengan metode *top-down approach*.

Logical Programming

Logical programming mempunyai dasar pada logika matematika dimana *program statements* mengekspresikan fakta dan aturan tentang pemecahan masalah.

Contohnya saat kita memesan kopi, *imperative approach* yang akan dilakukan adalah:

1. Masuk ke dalam toko kopi
2. Antri untuk memilih menu yang akan dibeli
3. Memesan menu yang dipilih
4. Memilih apakah kopi akan dinikmati di tempat atau dibungkus (*take away*)
5. Membayar
6. Mengambil pesanan dan meninggalkan tempat pemesanan

Functional Programming

Functional programming mempunyai ciri dengan membangun struktur dan elemen program. *Functional programming* terdiri dari ***pure function*** dimana *function* berisi argument yang menerima inputan dan akan mengembalikan nilai.

Beberapa contoh *functional programming* adalah:

- *Pure function*
- *Recursion*

Functional Programming (2)

- ***Pure function***

Output fungsi tergantung pada inputan yang diberikan.

```
function addNum($firNum, $secNum){  
    $sumNum = $firNum + $secNum;  
    return $sumNum;  
}
```

Functional Programming (3)

- ***Recursion***

Fungsi yang akan memanggil dirinya sendiri saat proses eksekusi.

```
function countInt ($input){  
    if($input <= 0){  
        return "Input a positive integer  
    ";  
    }  
    else if($input > 10){  
        return "Counting complete";  
    }  
    else{  
        return countInt($input + 1);  
    }  
}
```

Object-Oriented Programming

Object-oriented programming merepresentasikan objek dalam bentuk ***class***. Setiap *class* akan menyimpan seluruh data dan fungsi serta dapat berinteraksi dengan *class* lain.

Features pada OOP:

- *Encapsulation*
Fitur mendasar berfungsi untuk menyembunyikan detail yang tidak ingin ditampilkan. Konsep ini membungkus data dan *internal method* pada objek tersebut.
- *Inheritance*
Mekanisme untuk mengambil sifat dari kelas lain dan digunakan dalam pembentukan hirarki yang saling berbagi atribut dan *method*.

Object-Oriented Programming (2)

- *Data abstraction*
Reduksi untuk menyederhanakan representasi keseluruhan data. *Data abstraction* biasanya adalah tahap pertama dari *database design*.
- *Polymorphism*
Konsep OOP yang mengambil kemampuan dari *variable*, fungsi atau objek untuk membentuk berbagai bentuk (*forms*).

Best-Practice untuk PHP

Inisialisasi variable

Inisialisasi variable harus dilakukan pertama kali sebelum penggunaan variable tersebut

- Boleh saja:

```
<?php  
  
if ($expr) {  
    // ....  
}  
  
$movies = array();  
$movies = get_movies();  
  
// EOF
```

- Namun sebaiknya gunakan ini:

```
<?php  
  
$movies = array();  
  
if ($expr) {  
    // ....  
}  
  
$movies = get_movies();  
  
// EOF
```

Inisialisasi/ Urutan deklarasi variable, method dan properties

- HARUS diawali dengan global, ikuti dengan konstanta, dan diakhiri dengan variabel local
- HARUS diawali dengan properti kemudian diikuti dengan method dalam class
- HARUS diawali dengan public, diikuti protected, dan diakhiri dengan private method dalam class
- HARUS sesuai dengan abjad dalam kelompoknya

Inisialisasi/ Urutan deklarasi variable, method dan properties

```
<?php

global $app_config,
       $cache,
       $db_connection;

define('ENVIRONMENT', 'PRODUCTION');
define('MODE', 1);

$id = 0;
$firstname = '';
$lastname = '';

// EOF
```

```
<?php

namespace MyCompany\Model;

class Office
{
    private $id;
    private $name;
    private $status;

    public function get_id() {
        // ...
    }

    protected function get_status() {
        // ...
    }

    private function get_name() {
        // ...
    }
}

// EOF
```

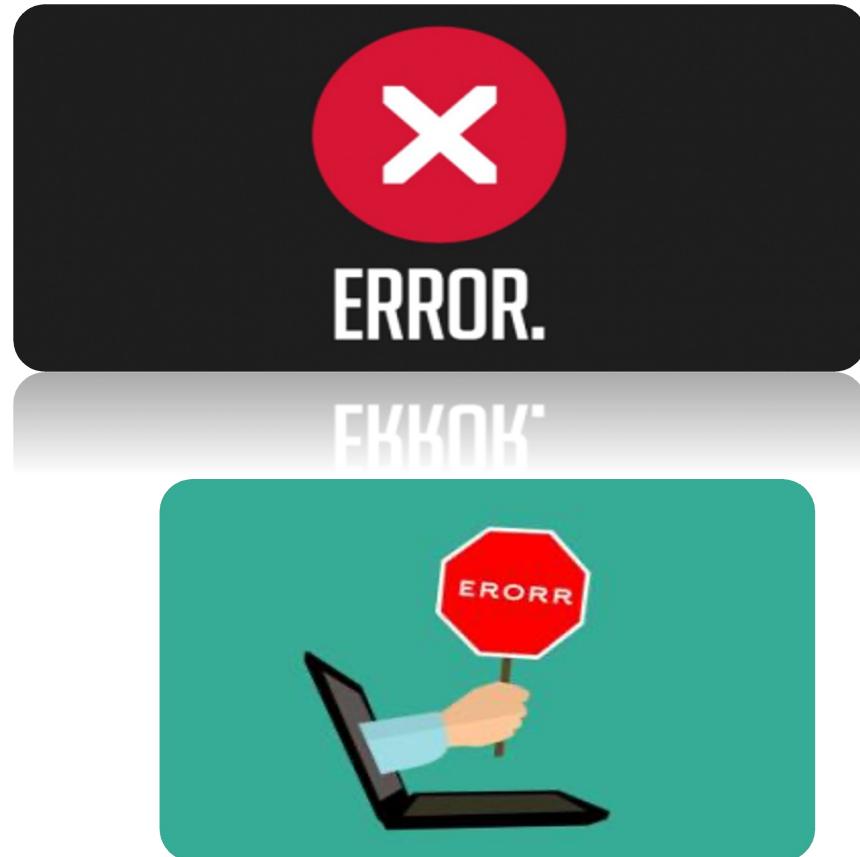


Galat / Error

Eksekusi Script Web Sesuai Skenario



Identifikasi Kesalahan / Error Saat Eksekusi Web



Jenis Error pada Software

- ❖ Internal Error : error pada kode program
- ❖ Eksternal Error : error pada interaksi dengan hal lain diluar kode program
- ❖ Internal error dapat dicegah dengan pemrograman yang lebih hati-hati
- ❖ Eksternal error berkaitan dengan gagal membuka file/database, tidak terkoneksi dengan jaringan, tidak dapat membuka module/file php, dll).

Identifikasi Kesalahan Eksekusi Web

Jenis Kesalahan pada saat eksekusi Web :

- ❖ *Error* : Kesalahan tersebut dapat berasal dari kesalahan kode pada script
- ❖ *Http Error* : Kesalahan komunikasi script dengan server php

Kesalahan pada Kode PHP (*Error*)

Error merupakan salah satu jenis kesalahan yang terjadi ketika melakukan eksekusi pada suatu halaman web, yang menyebabkan halaman web tidak menampilkan hasil sesuai dengan yang diinginkan. Error dapat diartikan sebagai penyimpangan dari kondisi yang benar.

Pada PHP terdapat empat jenis *kesalahan/error*, yaitu :

1. *Parse Error / Syntax Error*
2. *Fatal Error*
3. *Warning Error*
4. *Notice*

Parse Errors

- ❖ Merupakan error yang terjadi jika ada kesalahan sintaks dalam script dan akan memunculkan pesan kesalahan sebagai output pada halaman browser.
- ❖ Parse error akan menghentikan proses eksekusi dari sebuah script, sehingga tidak dapat menampilkan hasil apapun kecuali pesan kesalahan.

Penyebab :

- ❖ Kutipan yang tidak ditutup, tanda petik tidak sesuai, petik satu atau dua
- ❖ Kelebihan atau kekurangan tanda kurung (" () ")
- ❖ Kurung kurawal yang tidak ditutup ({})
- ❖ Kurang tanda titik koma (;)
- ❖ Kurang tanda titik sebagai penggabungan beberapa string (.)
- ❖ Kesalahan dalam penulisan nama variable, aturan penulisan nama variabel, yaitu : diawali tanda \$, dan diikuti oleh huruf atau underscore (_), tidak boleh diawali spasi, angka dan karakter khusus.

- ***Parse Errors (Syntax Error)***

- Kesalahan *syntax* (kutip, kurung, titik koma, dll) dalam *script* yang biasanya diakibatkan oleh kesalahan pengetikan.
- Pesan kesalahan akan muncul pada *outputnya* ketika dijalankan.
- *Parse error* akan menghentikan proses eksekusi *script*.

```
15 echo "Junior Web Developer";  
16 echo "BPPTIK"  
17 echo "Kementerian Kominfo";
```

```
15 echo "Junior Web Developer";  
16 echo "BPPTIK";  
17 echo "Kementerian kominfo";
```

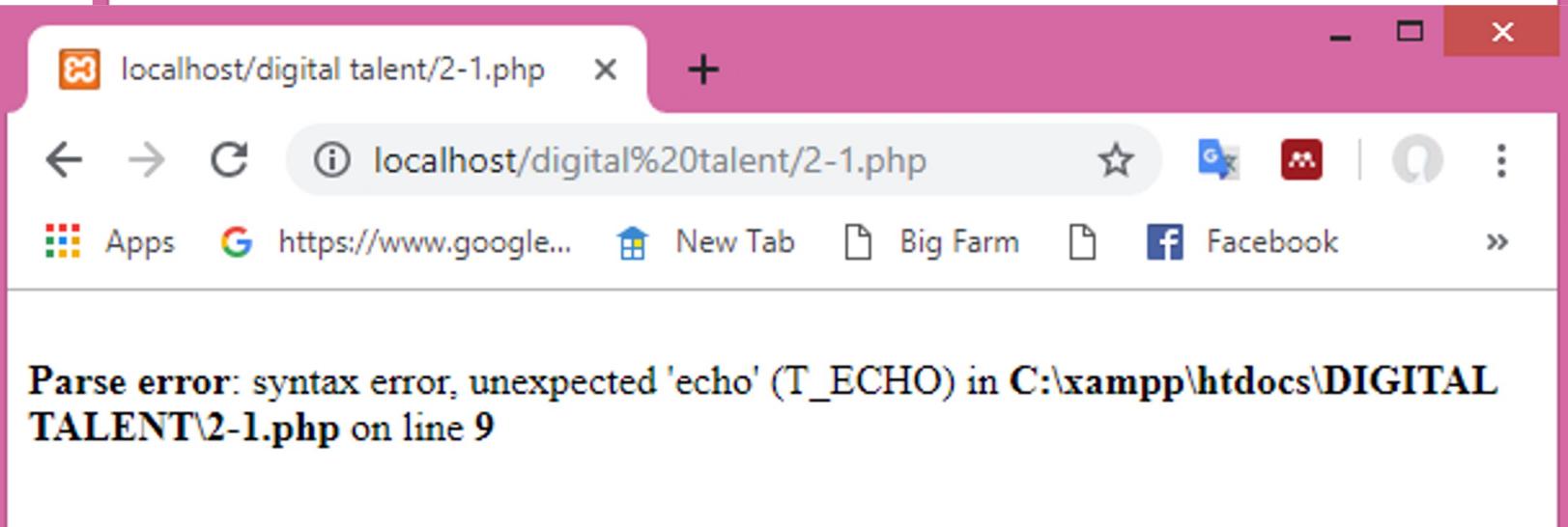
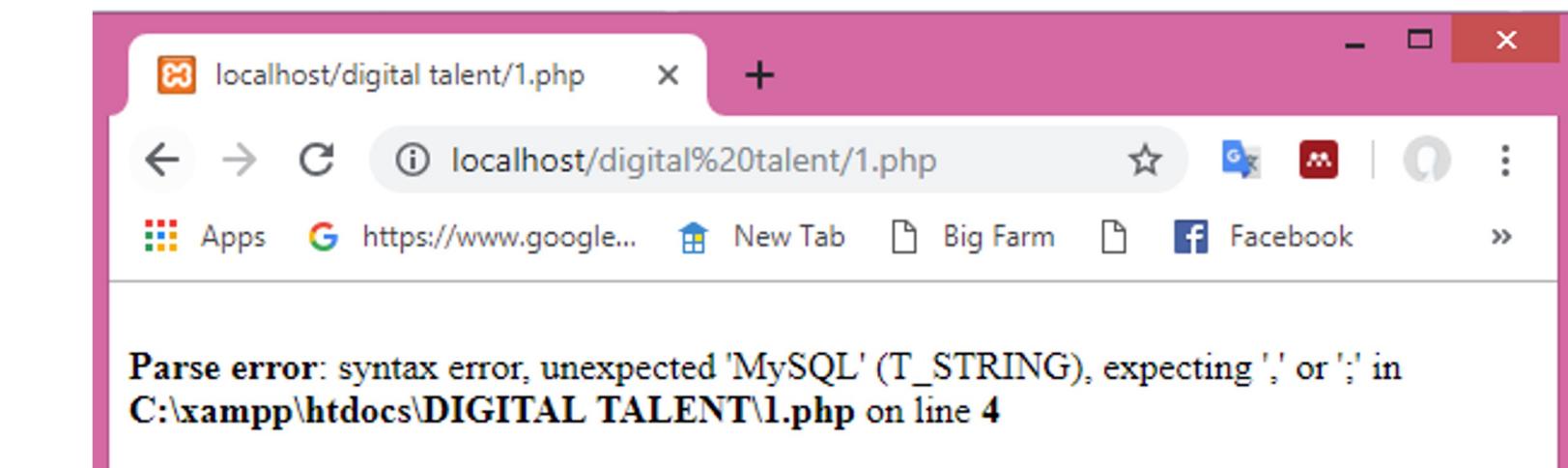
Parse error: syntax error, unexpected 'echo' (T_ECHO), expecting ';' or ',' in C:\xampp\htdocs\test.php on line 17

Kesalahan terjadi karena kurangnya tanda titik koma (;) di baris ke 16

Contoh Parse Error

```
1 <?php  
2     echo "PHP";  
3     echo "HTML';  
4     echo "MySQL;  
5 ?>
```

```
1 <?php  
2 function test_error1()  
3 {  
4     echo "ini fungsi  
test_error1";  
5 }  
6  
7 test_error2()  
8 {  
9     echo "fatal error!";  
10 }  
11  
12 ?>
```



Fatal Errors

- Merupakan *error* yang terjadi ketika PHP mengerti kode yang ditulis pada script, namun apa yang diminta pada kode script tidak dapat dieksekusi.
- Fatal error juga dapat terjadi saat mencoba mengakses fungsi yang belum didefinisikan.

Penyebab :

- Apa yang diminta pada kode script tidak dapat dieksekusi.
- Menghentikan proses eksekusi dari sebuah script, sehingga outputnya hanya berupa pesan kesalahan.

- **Fatal Errors**
 - PHP mengerti kode program yang ditulis, namun apa yang diminta tidak dapat dilakukan.
 - Misalnya program memanggil fungsi yang tidak terdefinisi.
 - *Fatal error* akan menghentikan eksekusi *script*.

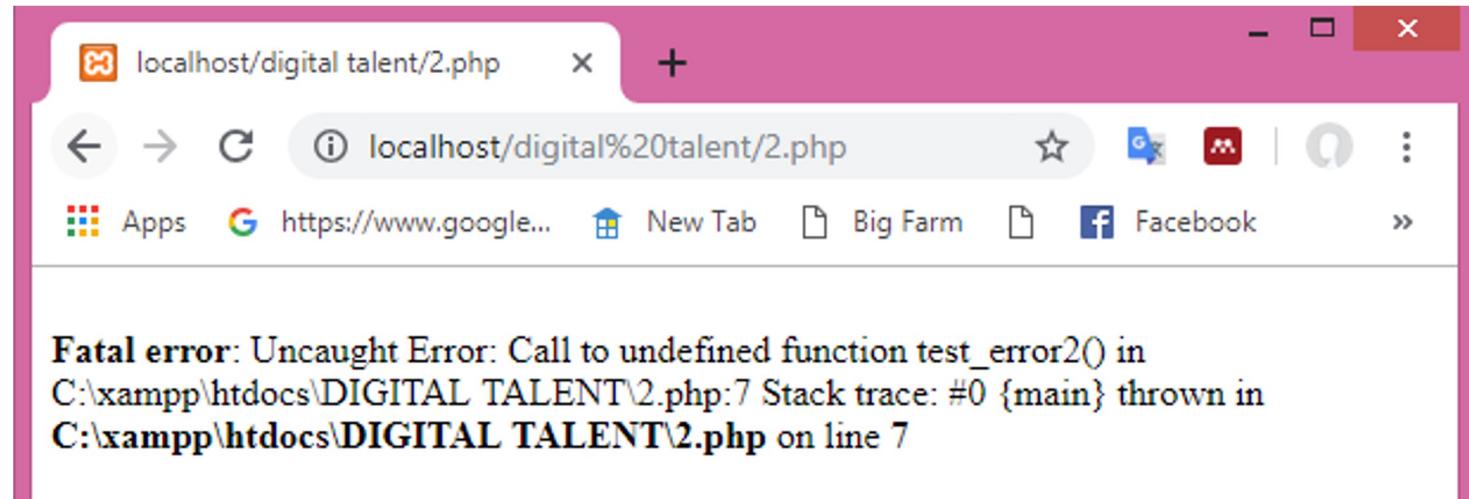
```
10     function luasPersegi($sisi){  
11         $luas = $sisi * $sisi;  
12         return $luas;  
13     }  
14  
15     luasSegitiga();
```

Fungsi luasSegitiga() yang dipanggil pada baris 15 harus didefinisikan terlebih dahulu

Fatal error: Uncaught Error: Call to undefined function luasSegitiga() in C:\xampp\htdocs\test.php:15 Stack trace: #0 {main} thrown in C:\xampp\htdocs\test.php on line 15

Fatal Errors

```
1 <?php
2 function test_error1()
3 {
4     echo "ini fungsi
5         test_error1";
6 }
7
8 test_error2();
9 echo "fatal error!";
10
11 ?>
```



Pada contoh script, terdapat kesalahan yaitu kurangnya penulisan **function** pada **test_error2()** sehingga muncul fatal error.

Warning Errors

- ❖ Fungsi **include** digunakan untuk memanggil file lain, namun karena file yang dipanggil tidak ditemukan, maka terjadi warning error.
- ❖ Warning error juga dapat terjadi jika pada fungsi aritmatika dimana terdapat pembagian dengan angka 0.
- ❖ *Warning error* tidak akan menghentikan proses eksekusi dari sebuah script, namun, akan menampilkan pesan berupa *warning* pada browser.

Penyebab :

- Menggunakan fungsi include pada sebuah *missing file*
- Kesalahan parameter yang terdapat pada sebuah fungsi.
- *Warning error* tidak akan menghentikan proses eksekusi dari sebuah script.

- **Warning Errors**

- File yang tidak ada atau jumlah parameter yang tidak pas saat pemanggilan suatu fungsi.
- *Warning error* tidak akan menghentikan eksekusi dari *script*.
- Kesalahan yang terjadi pada contoh di bawah yaitu file connect.php tidak ada sehingga Ketika file.php dijalankan dan memanggil file connect.php dengan fungsi include, maka akan mengeluarkan pesan warning eror
- Fungsi **include** digunakan untuk memanggil file lain, namun karena file yang dipanggil tidak ditemukan, maka terjadi warning error.

```
15
```

```
include ("connect.php");
```

File connect.php tidak ditemukan

Warning: include(connect.php): failed to open stream: No such file or directory in
C:\xampp\htdocs\test.php on line 15

Warning: include(): Failed opening 'connect.php' for inclusion (include_path='C:\xampp
\php\PEAR') in C:\xampp\htdocs\test.php on line 15

Contoh Warning Errors

```
1 <?php  
2     include ("Welcome.php");  
3     echo "Warning Error";  
4  
5 ?>
```

A screenshot of a Microsoft Edge browser window titled "localhost/digital talent/3.php". The address bar shows "localhost/digital%20talent/3.php". The page content displays two warning messages:
Warning: include(Welcome.php): failed to open stream: No such file or directory in C:\xampp\htdocs\DIGITAL TALENT\3.php on line 2
Warning: include(): Failed opening 'Welcome.php' for inclusion (include_path='C:\xampp\php\PEAR') in C:\xampp\htdocs\DIGITAL TALENT\3.php on line 2
Warning Error

```
1 <?php  
2 $a = 10;  
3 $b = 0;  
4  
5 echo $a / $b;  
6 ?>
```

A screenshot of a Microsoft Edge browser window titled "localhost/digital talent/3-1.php". The address bar shows "localhost/digital%20talent/3-1.php". The page content displays one warning message:
Warning: Division by zero in C:\xampp\htdocs\DIGITAL TALENT\3-1.php on line 5
INF

Notice Errors

Notice error hampir sama dengan *warning error*, tidak menghentikan proses eksekusi script.

- Eksekusi tetap dijalankan, tapi pada bagian yang error akan ditampilkan *notice*.

Penyebab :

- *Undefined* variabel (variabel yang dipanggil atau dieksekusi, tidak atau belum didefinisikan)
- *Notice error* tidak menghentikan proses eksekusi script.

- ***Notice Errors***

- Variable yang diakses belum didefinisikan.
- *Notice error* tidak mengentikan *script*.

```
15     $lokasi = "BPPTIK Kementerian Kominfo";  
16     echo $waktu;
```

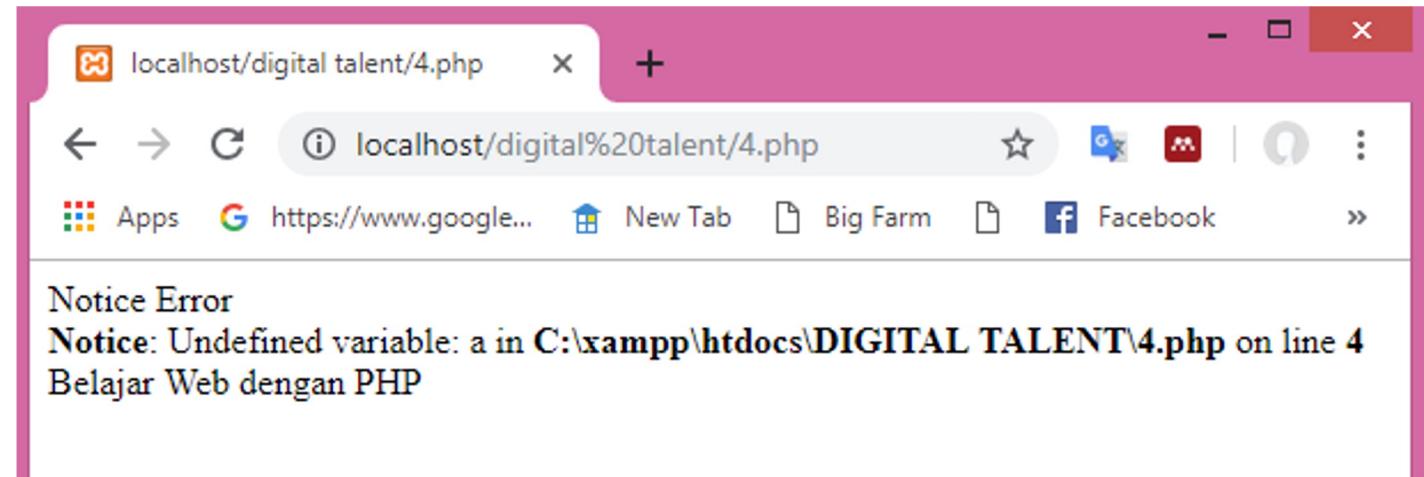
Variable \$waktu belum didefinisikan

Notice: Undefined variable: waktu in C:\xampp\htdocs\test.php on line 16

Notice Errors

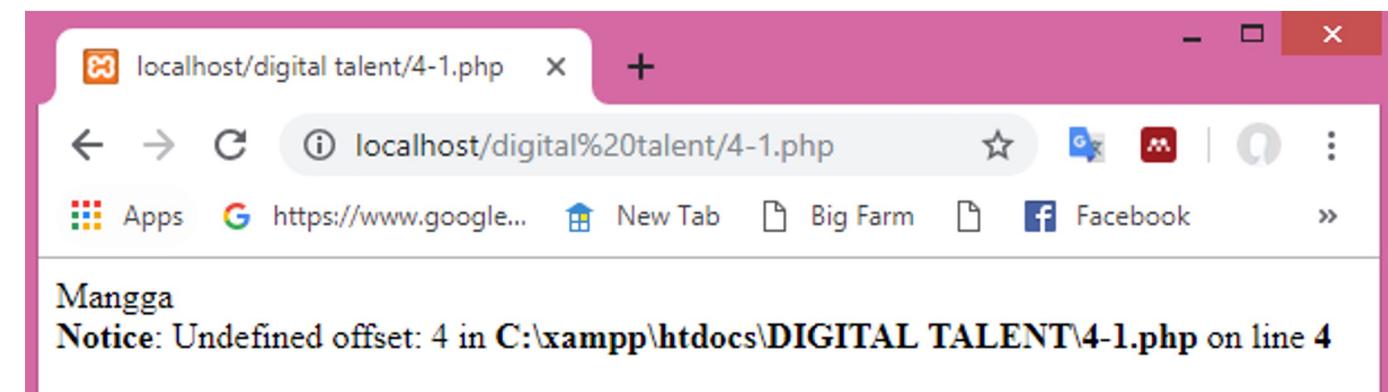
Pada contoh pertama, yang dipanggil oleh echo adalah variabel **a**, sedangkan variabel **a** tidak atau belum didefinisikan, sehingga menyebabkan notice error.

```
1 <?php
2     $x = "Belajar Web
3         dengan PHP";
4     echo "Notice Error";
5     echo $a;
6     echo $x;
7 ?>
```



Pada contoh kedua terjadi notice error karena pengaksesan elemen array yang tidak atau belum ada nilainya.

```
1 <?php
2 $arrBuah = array ("Mangga", "Apel",
3 "Pisang", "Jeruk");
4 echo $arrBuah[0]; //Mangga
5 echo $arrBuah[4]; //Jeruk
6 ?>
```



Kesalahan Berupa HTTP Error

Daftar kode status HTTP terdiri dari:

1xx : **Informasi**, yang menunjukkan bahwa Permintaan dipahami, melanjutkan proses. Kode ini hanya digunakan untuk status saja.

2xx : **Sukses**, Pada kode ini, server memberikan status suksesnya diterima, dipahami, disetujui, dan diproses.

3xx : **Pengalihan**

4xx : **Kesalahan Klien**, Pada kode ini, klien memberikan status kesalahan dalam memproses permintaan.

5xx : **Kesalahan Server**

Kesalahan pada Client

- ❖ 400 Permintaan Tak Layak
- ❖ 401 Unauthorized
- ❖ 402 Payment Required
- ❖ 403 Terlarang
- ❖ 404 Tidak Ditemukan
- ❖ 405 Method Not Allowed
- ❖ 406 Not Acceptable
- ❖ 407 Proxy Authentication Required
- ❖ 408 Request Timeout
- ❖ 409 Conflict
- ❖ 410 Tidak tersedia

Kesalahan pada Client

- ❖ 411 Length Required
- ❖ 412 Precondition Failed
- ❖ 413 Request Entity Too Large
- ❖ 414 Request-URI Too Long
- ❖ 415 Unsupported Media Type
- ❖ 416 Requested Range Not Satisfiable
- ❖ 417 Expectation Failed
- ❖ 419 Authentication Timeout
- ❖ 420 Method Failure

Kesalahan pada Server

- ❖ 500 Internal Server Error
- ❖ 501 Not Implemented
- ❖ 502 Bad Gateway
- ❖ 503 Service Unavailable
- ❖ 504 Gateway Timeout
- ❖ 505 HTTP Version Not Supported
- ❖ 506 Variant Also Negotiates
- ❖ 507 Insufficient Storage
- ❖ 508 Loop Detected
- ❖ 509 Bandwidth Limit Exceeded

Kesalahan pada Server

- ❖ 510 Not Extended
- ❖ 511 Network Authentication Required
- ❖ 520 Origin Error
- ❖ 521 Web server is down
- ❖ 522 Connection timed out
- ❖ 523 Proxy Declined Request
- ❖ 524 A timeout occurred
- ❖ 598 Network read timeout error
- ❖ 599 Network connect timeout error

Jenis Kesalahan Umum

Beberapa kesalahan yang umum dan sering terjadi antara lain :

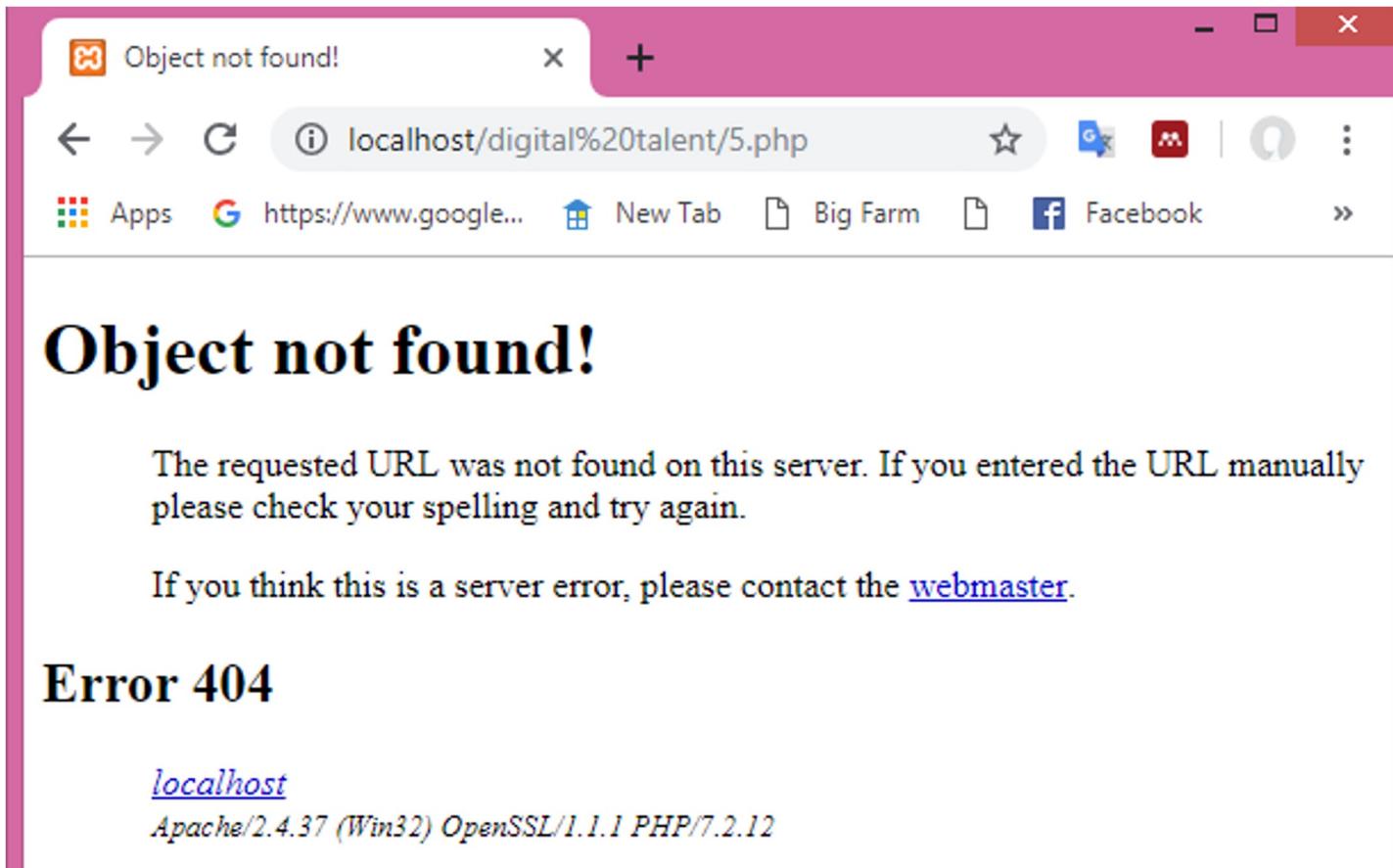
1. Error 404 (Not Found)
2. Error 403 (Forbidden)
3. Error 500 (Internal Server Error)
4. Error 503 (Service Unavailable)
5. Error 504 (Gateway Time-out)

1. Error 404 (Not Found)/ Broken Link

Penyebab :

- ❖ Kesalahan penulisan URL
- ❖ URL halaman telah diubah oleh si pengunggah
- ❖ Halaman yang diakses sudah tidak tersedia
- ❖ Halaman telah dihapus

Contoh Error 404 (Not Found)/ Broken Link



- Error ini merupakan peringatan yang muncul ketika browser Anda tidak menemukan halaman atau file yang diakses.
- Dapat terjadi karena konten yang diakses hilang, telah diubah atau mengalami kerusakan.

2. Error 403 (Forbidden)

Penyebab :

- ❖ Permasalahan pada *permission/hak* akses suatu halaman atau script.

Contoh Error 403 (Forbidden)

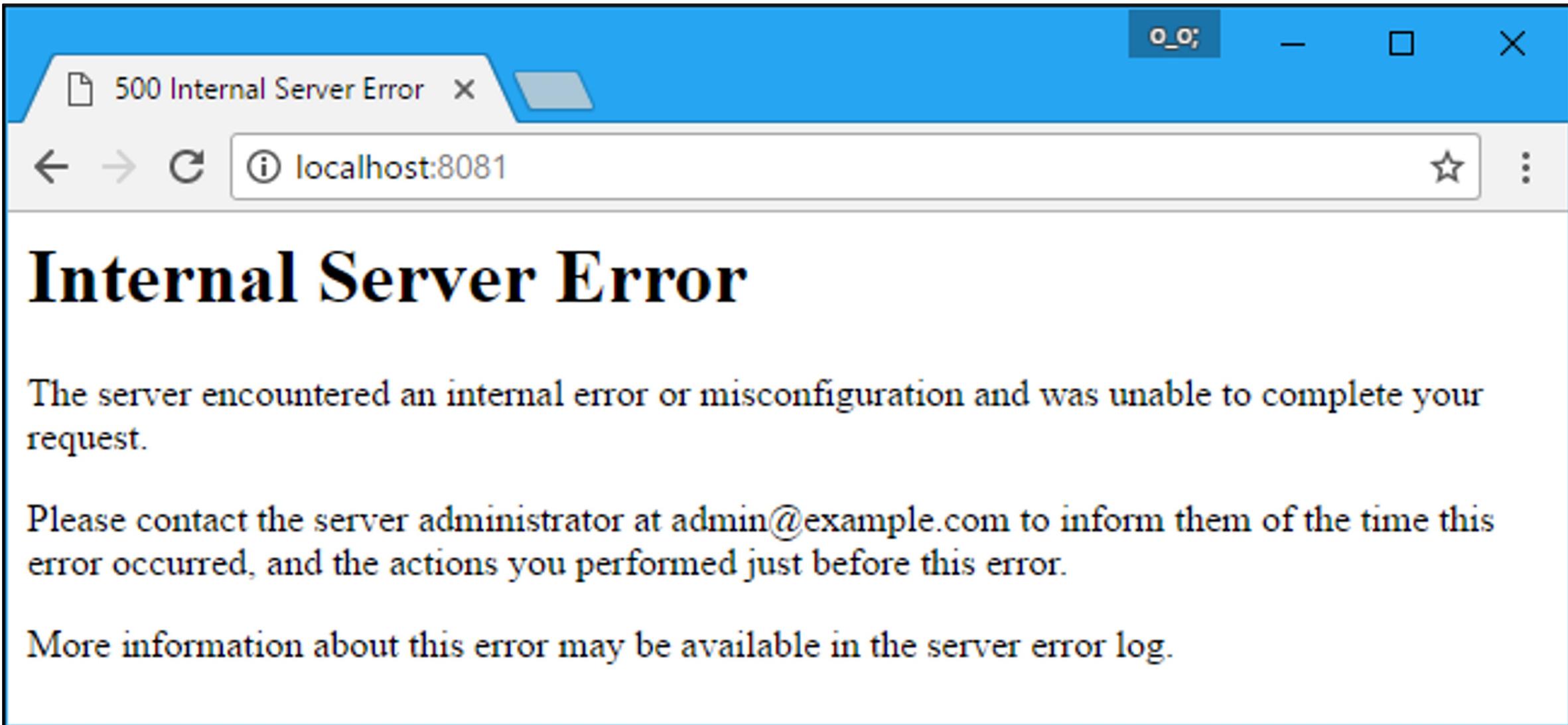


3. Error 500 (Internal Server Error)

Penyebab :

- ❖ Adanya permasalahan komunikasi antara server dan script website.
- ❖ Perubahan konfigurasi file .htaccess
- ❖ Perubahan yang tidak disengaja pada file .htaccess
- ❖ Instalasi plugin/extension yang melakukan file .htaccess
- ❖ Terhapusnya file .htaccess

Contoh Error 500 (Internal Server Error)

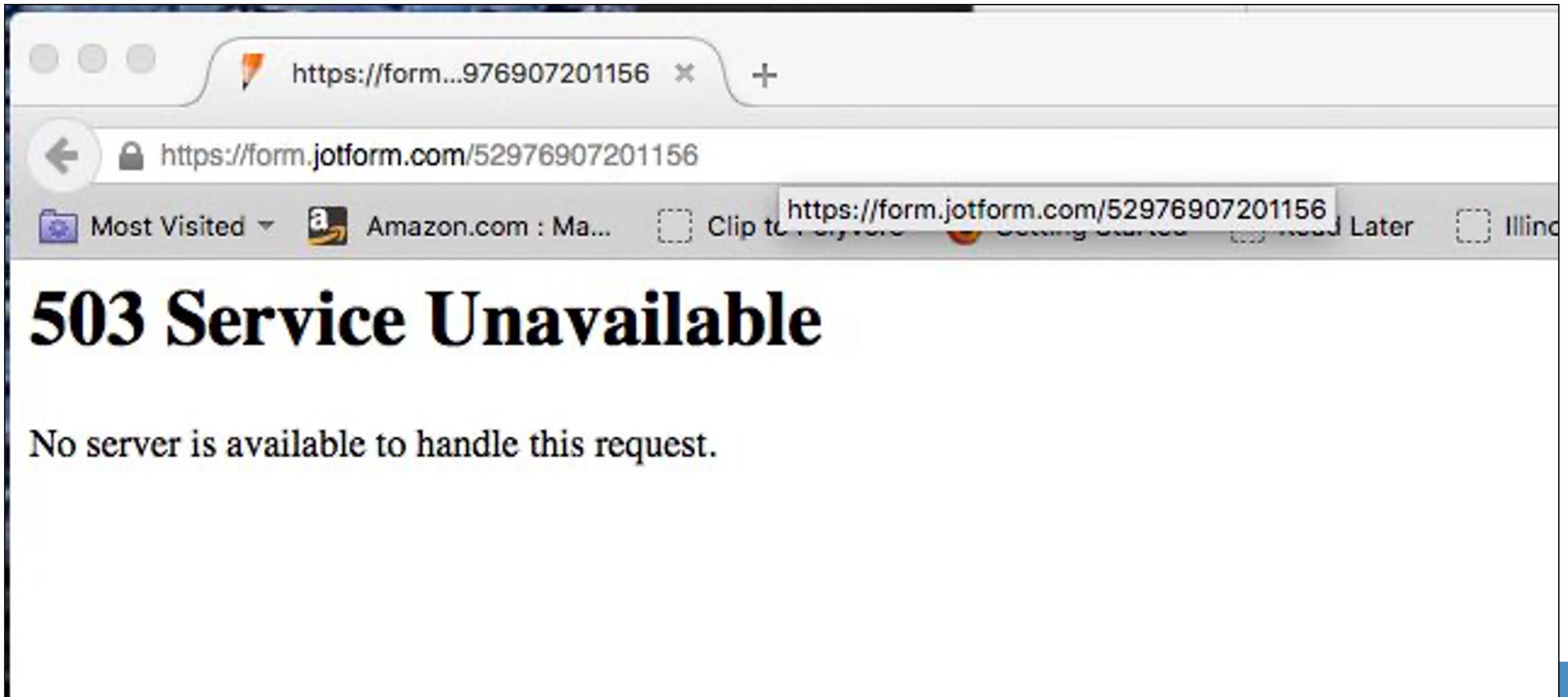


4. Error 503 (*Service Unavailable*)

Penyebab :

- ❖ Permasalahan server
- ❖ Server *down*,
- ❖ Server dalam kondisi *maintenance*,
- ❖ Penggunaan *resource server* yang cukup tinggi

Contoh Error 503 (*Service Unavailable*)



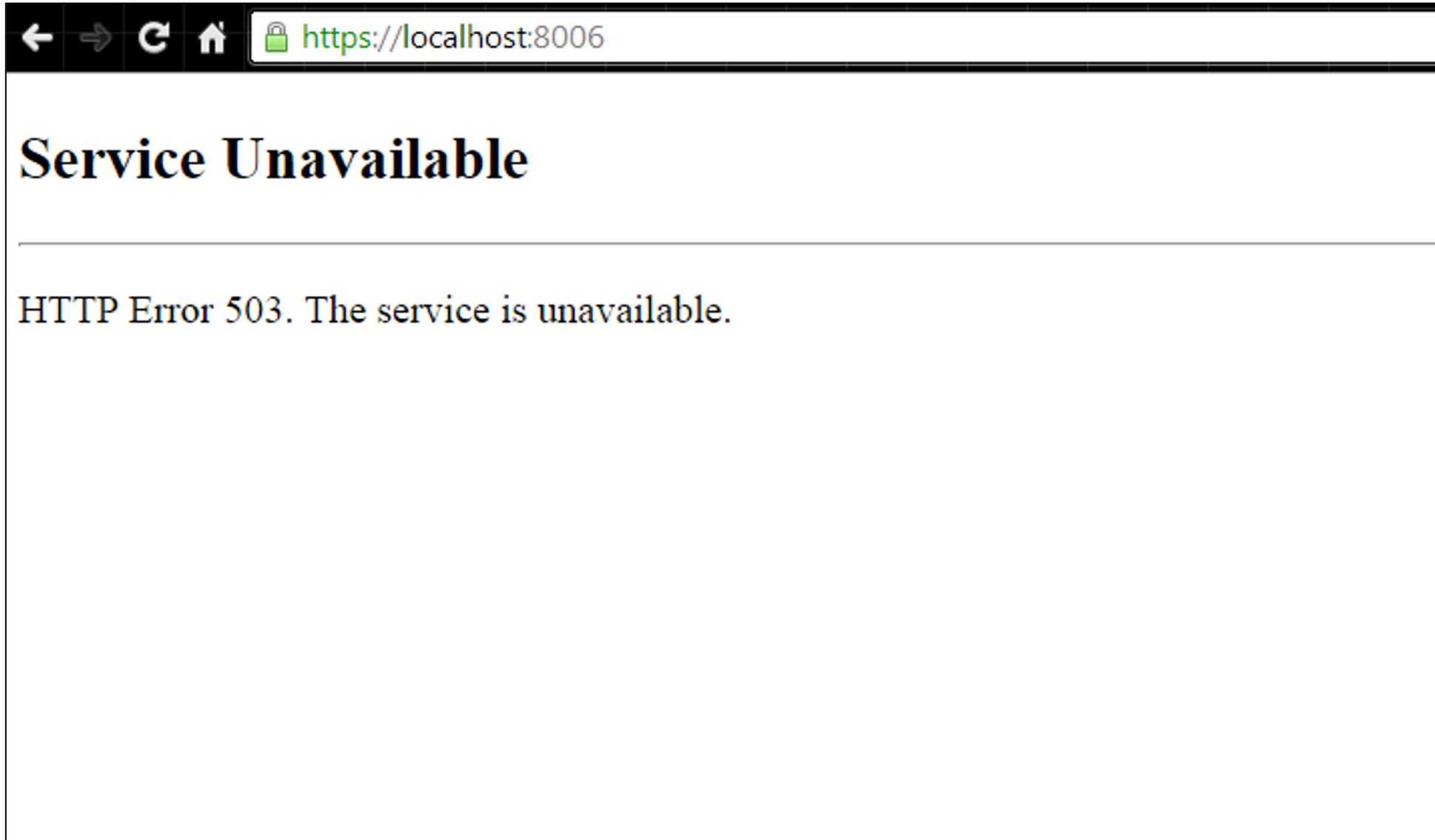
5. Error 504 (*Gateway Time-out*)

Penyebab :

- *Request* (permintaan) data yang terlalu lama ke server
- Dipengaruhi oleh gangguan pada penyedia layanan internet
- Pembagian serta penggunaan data akses yang tidak seimbang
- Kesalahan jaringan server di Internet maupun koneksi internet pada komputer yang digunakan

Contoh Error 504 (*Gateway*

Time-out)



Web Tidak Dapat diakses Setelah Hosting

Kondisi yang harus diperhatikan :

1. Setting DNS
2. File index
3. Nama domain
4. Koneksi Internet
5. IP terblokir
6. Komputer terserang ARP Spoofing/trojan/virus
7. Upload file
8. Pengaturan file php.ini
9. Pengaturan database

1. Kesalahan Setting DNS

Yang perlu diperhatikan :

- Terjadi perubahan pada DNS
- Setting DNS salah
- **DNS** (Domain Name Server) adalah sebuah standar teknologi yang mengatur penamaan publik dari sebuah situs website, atau juga bisa disebut dengan sebuah sistem yang menyimpan informasi tentang nama host atau nama domain dalam bentuk distributed database didalam jaringan komputer. Dengan adanya DNS ini maka semua orang dapat menulis domain di web browser anda dan komputer anda akan menemukan domain tersebut di internet.
- Yang perlu diperhatikan :
Website tidak dapat diakses dalam beberapa jam atau beberapa hari setelah perubahan DNS. Meskipun setting DNS salah, website masih bisa diakses dalam beberapa jam, namun tiba-tiba tidak bisa diakses selang beberapa waktu kemudian

2. File Index

Yang perlu diperhatikan :

- File index tidak ada di server web
- File index kosong

File index tidak ditemukan :

- Ketika sebuah halaman web diakses, maka server web akan mencoba membuka salah satu file berikut: index.html, index.htm, index.php (server web akan mencari keberadaan file-file tersebut dalam urutan seperti itu).
- Jika salah satu dari file tersebut tidak ada server web akan menampilkan error seperti *error document*, *No Index file*, atau server web akan menampilkan daftar semua file dan folder yang ada didalam account web tersebut.

File index kosong

- Jika file index kosong (tidak ada isinya), program browser internet akan menampilkan halaman kosong.
- Jika halaman index anda berupa index.php, maka munculnya halaman kosong dapat juga disebabkan karena adanya error dalam file index.php dan server web telah di setting untuk menampilkan halaman kosong jika terjadi kesalahan dalam pemrosesan file php tersebut.

3. Nama Domain

Yang perlu diperhatikan :

- File index tidak ada di server web
- File index kosong
- Waktu kadaluarsa atau expired dari nama domain

4. Koneksi Internet

Yang perlu diperhatikan :

- Tidak dapat mengakses situs tersebut sedangkan situs-situs lain bisa diakses
- Hal yang pertama kali dilakukan adalah memeriksa koneksi dari komputer ke server web dimana situs tersebut berada menggunakan perintah seperti "ping" yang tersedia di komputer.
- Buka Command Prompt, ketikkan *ping www.domain.com*
- Jika hasilnya menunjukkan beberapa baris "reply", berarti sambungan ke server web tidak mengalami masalah.
- Waktu yang tertera dalam hasil perintah ping menunjukkan seberapa cepat sambungan dari komputer ke server web.

5. IP terblokir

Yang perlu diperhatikan :

- Tidak dapat mengakses semua halaman web
- Terblokir oleh Firewall

Beberapa penyebab :

Berkali-kali salah saat login
cpanel/webmail/ftp/ssh/pop3/imap

Terlalu sering *me-refresh* browser

Melakukan spam

Hacking

DOS attack

Port scan

6. Komputer Terserang Virus

Yang perlu diperhatikan :

- Kondisi komputer terkena virus atau tidak
- Scan menggunakan Antivirus

7. Upload File yang tidak Sempurna

Yang perlu diperhatikan :

- Ukuran dan jumlah file yang di-upload ke hosting

8. Pengaturan File php.ini

Yang perlu diperhatikan :

- Pengaturan register_global
- Pada beberapa hosting, register global OFF sedangkan di localhost biasanya ON, setelah web di upload kemudian diakses, web dapat muncul tapi jika di klik halaman lain pada web, tidak berganti halamannya atau isi dari variabelnya tidak terkirim.

9. Pengaturan Database

Yang perlu diperhatikan :

- Koneksi database
- Jika terdapat kesalahan mengenai database, perhatikan konfigurasi dari koneksi database, dan pastikan settingannya (hostnya, usernamenya, passwordnya, dan lain-lain) sudah benar.
- Selain itu cek versi rdbms local / rdbms untuk membuat web pastikan compatible dengan rdbms hosting.

Kesimpulan

1. Pengujian Source Code bertujuan untuk memastikan software dapat berjalan 100% sesuai dengan yang di harapkan baik secara Logika dan secara fungsional
2. Pengujian di lakukan dengan metode Black Box dan White Box
3. Error atau kesalahan eksekusi web dapat terjadi pada sisi internal (error source code) dan eksternal (http error)
4. Pada PHP terdapat empat jenis *kesalahan/error* , yaitu *Parse Error/ Syntax Error, Fatal Error, Warning Error, Notice*
5. Beberapa kesalahan yang umum dan sering terjadi antara lain : Error 404 (Not Found), Error 403 (Forbidden), Error 500 (Internal Server Error), Error 503 (Service Unavailable), Error 504 (Gateway Time-out)
6. Selain itu terdapat beberapa kesalahan yang harus diperhatikan setelah web dihosting ke server

Penanganan Galat / Error

Penanganan Galat / Error

- Penanganan error adalah proses menangkap kesalahan pada program untuk diambil tindakan yang sesuai.
- Metode Penanganan Error
 - Menggunakan Function "die()
 - Menulis function Penanganan Error sendiri
 - Error reporting

Function "die()"

Misalkan terdapat kode sbb:

```
<?php  
$file=fopen("hello.txt", "r");  
?>
```

Jika file tersebut tidak ditemukan, maka akan keluar pesan berikut:

```
Warning: fopen(hello.txt) [function.fopen]: failed to open stream:  
No such file or directory in /var/www/webfolder/test.php on line 2
```

Untuk menghindari pesan error seperti di atas, kita dapat memeriksa keberadaan file terlebih dahulu.

Ketika menulis program PHP, kita harus memeriksa semua kondisi error yang mungkin terjadi sebelum mengambil langkah berikutnya.

Penggunaan Function die();

Memeriksa keberadaan file terlebih dahulu

```
<?php  
if(!file_exists("hello.txt")) {  
    die("File tidak ditemukan");  
} else {  
    $file=fopen("hello.txt", "r");  
}  
?>
```

Jika file tidak ditemukan, maka akan muncul pesan:

```
File tidak ditemukan
```

Program akan terhenti ketika memanggil function die(); dan menampilkan pesan yang dapat dipahami oleh pengguna

Metode Menulis Function Penanganan Error Sendiri

Kita dapat menulis fungsi sendiri untuk menangani kesalahan apa pun

```
<?php
//function penanganan error
function customError($errno, $errstr) {
    echo "<b>Error:</b> [$errno] $errstr";
}

//set error handler
set_error_handler("customError");

//picu error
echo ($test);
?>
```

Error: [8] Undefined
variable: test

Metode Menulis Function Penanganan Error Sendiri

- Fungsi yang dibuat harus mampu menangani minimal dua parameter (tingkat kesalahan dan pesan kesalahan) tetapi dapat menerima hingga lima parameter (opsional: file, nomor baris, dan konteks kesalahan):
- Syntax:

```
error_function(error_level,error_message,error_file,error_line, error_context)
```

Metode Menulis Function Penanganan Error Sendiri

- Parameter Kesalahan:

Parameter	Deskripsi
error_level	Wajib. Menentukan tingkat laporan kesalahan untuk kesalahan yang ditentukan pengguna. Harus berupa angka nilai. Lihat tabel di slide berikutnya untuk daftar tingkat laporan kesalahan.
error_message	Wajib. Pesan error yang disampaikan ke pengguna.
error_file	Tidak wajib. Menampilkan nama file dimana error terjadi
error_line	Tidak wajib. Menampilkan nomor baris dimana error terjadi.
error_context	Tidak Wajib. Menampilkan sebuah array yang mengandung setiap variabel, dan nilainya yang digunakan ketika terjadi error.

Metode Menulis Function Penanganan Error Sendiri

- Level Kesalahan:

Nilai	Konstanta	Deskripsi
2	E_WARNING	Kesalahan yang tidak fatal. Eksekusi program tidak terhenti.
8	E_NOTICE	Pemberitahuan Run-time. Dalam kode mungkin terdapat sebuah kesalahan, tetapi dapat juga terjadi ketika kode berjalan normal
256	E_USER_ERROR	Kesalahan fatal yang dibuat oleh programmer. Ini seperti sebuah E_ERROR yang di set oleh programmer menggunakan function trigger_error()
512	E_USER_WARNING	Peringatan yang tidak fatal yang dibuat oleh programmer. Ini seperti sebuah E_WARNING yang di set oleh programmer menggunakan function trigger_error()
1024	E_USER_NOTICE	Pemberitahuan yang dibuat oleh programmer. Ini seperti sebuah E_NOTICE yang di set oleh programmer menggunakan function trigger_error()
4096	E_RECOVERABLE_ERROR	Kesalahan fatal yg dapat ditangkap. Ini seperti sebuah E_ERROR Tapi yang ditangkap oleh programmer dg mendefinisikan handle (set_error_handler())
8191	E_ALL	Semua kesalahan dan peringatan

Penanganan Pengecualian (Exception)

- Pengecualian digunakan untuk mengubah alur program yg normal jika terjadi kesalahan (error) tertentu. Kondisi tersebut disebut pengecualian.
- Pada dasarnya yang terjadi saat exception dipicu adalah:
 1. Keadaan kode yang paling baru (sebelum terjadi kesalahan) disimpan
 2. Eksekusi kode akan beralih ke fungsi handler exception (custom) yang telah ditentukan
 3. Bergantung pada situasinya, handler kemudian dapat melanjutkan eksekusi dari status kode yang disimpan, mengakhiri eksekusi program atau melanjutkan program dari lokasi yang berbeda dalam kode.

Penggunaan Exception dasar

Penulisan Exception yang standard harus mengandung:

- try - Fungsi yang menggunakan exception harus berada di dalam blok "try". Jika exception tidak terpicu, kode akan dilanjutkan seperti biasa. Namun jika exception terpicu, exception akan di-"throw".
- throw - Ini adalah bagaimana Anda memicu exception. Setiap "throw" harus memiliki setidaknya satu "catch".
- catch - Blok "catch" mengambil exception dan membuat objek yang berisi informasi dari execption.

Contoh Penggunaan Exception

```
<?php
//create function with an exception
function checkNum($number) {
    if($number>1) {
        throw new Exception("Value must be 1 or below");
    }
    return true;
}

//trigger exception in a "try" block
try {
    checkNum(2);
    //If the exception is thrown, this text will not be shown
    echo 'If you see this, the number is 1 or below';
}

//catch exception
catch(Exception $e) {
    echo 'Message: ' . $e->getMessage();
}
?>
```

- Kode di samping melempar exception dan menangkapnya:
- Fungsi checkNum () dibuat. Ia memeriksa apakah angka lebih besar dari 1. Jika ya, pengecualian dilemparkan
- Fungsi checkNum () dipanggil dalam blok "coba"
- Pengecualian dalam fungsi checkNum () dilemparkan
- Blok "catch" mengambil pengecualian dan membuat objek (\$ e) yang berisi informasi pengecualian
- Pesan kesalahan dari pengecualian digaungkan dengan memanggil \$ e->getMessage () dari objek pengecualian

Contoh Penggunaan Exception

```
<?php
//create function with an exception
function checkNum($number) {
    if($number>1) {
        throw new Exception("Value must be 1 or below");
    }
    return true;
}

//trigger exception in a "try" block
try {
    checkNum(2);
    //If the exception is thrown, this text will not be shown
    echo 'If you see this, the number is 1 or below';
}

//catch exception
catch(Exception $e) {
    echo 'Message: ' . $e->getMessage();
}
?>
```

Apabila dijalankan akan menghasilkan



Message: Value must be 1 or below

Error Reporting

- Fungsi `error_reporting(0)` menentukan kesalahan mana yang dilaporkan.
- PHP memiliki banyak level kesalahan, dan menggunakan fungsi ini menetapkan level kesalahan tersebut untuk kode program yang sedang dijalankan.
- Syntax:

```
error_reporting(Level);
```

Error Reporting (Contoh)

```
<?php  
    // Turn off error reporting  
    error_reporting(0);  
  
    // Report runtime errors  
    error_reporting(E_ERROR | E_WARNING | E_PARSE);  
  
    // Report all errors  
    error_reporting(E_ALL);  
  
    // Same as error_reporting(E_ALL);  
    ini_set("error_reporting", E_ALL);  
  
    // Report all errors except E_NOTICE  
    error_reporting(E_ALL & ~E_NOTICE);  
?>
```

Kinerja Situs Web

- **Ketika berbicara** tentang kinerja aplikasi web, kita berbicara tentang kinerja web atau kinerja runtime.
- **Kinerja web** sebagai ukuran waktu mulai dari saat pengguna akhir meminta konten hingga kapan konten itu tersedia di perangakat pengguna.
- **Kinerja runtime** sebagai indikasi seberapa responsif aplikasi terhadap input pengguna saat runtime.

Menggunakan Ukuran Performansi dalam Menuliskan kode sumber

Mengukur Kinerja Program (kecepatan)

```
<?php  
$awal = microtime(true);  
  
// Berhenti untuk sesaat atau berisi blok kode program yg diukur  
usleep(1000);  
  
$akhir = microtime(true);  
$waktu = $akhir - $awal;  
  
echo "Program berjalan selama $waktu detik\n";  
?>
```

- **Waktu eksekusi** program dihitung dengan menghitung selisih waktu awal program dimulai dengan waktu program selesai.
- `Microtime()` – menghasilkan waktu saat ini
- Ref <https://www.php.net/manual/en/function.microtime.php>

Menggunakan Ukuran Performansi dalam Menuliskan kode sumber

Mengukur Kinerja Program (penggunaan memory)

```
<?php  
/*  
Blok kode program yang akan diukur penggunaan memorinya  
*/  
  
$penggunaan_memory = memory_get_usage();  
echo "Program menggunakan memory sebesar  
$penggunaan_memory bytes\n";  
  
?>
```

- Fungsi `memory_get_usage()` akan menghasilkan output jumlah memory yang digunakan oleh program dalam bytes
- Ref <https://www.php.net/manual/en/function.memory-get-usage.php>

Menggunakan Ukuran Performansi dalam Menuliskan kode sumber

Menghitung efisiensi program

$$Efisiensi = \left(\frac{\text{Kinerja setelah efisiensi}}{\text{(Kinerja sebelum efisiensi)}} \right) \times 100\%$$

- Kinerja program dapat berupa kecepatan atau penggunaan memory

Menulis kode PHP dengan Efisien

- **Menggunakan Fungsi bawaan PHP**

Menggunakan fungsi bawaan PHP lebih cepat dibanding menulis fungsi sendiri.

Referensi fungsi bawaan PHP dapat dilihat di manual PHP
<https://www.php.net/manual/en/funcref.php>

- **Membuat Kelas hanya jika diperlukan**

Kelas dan method dibuat jika memang benar-benar diperlukan (digunakan kembali di banyak kode).

- **Menutup Koneksi**

Menutup koneksi basisdata setelah tidak dibutuhkan akan menghemat memory.

Menulis kode PHP dengan Efisien

- **Menggunakan petik tunggal**

Ketika menggunakan string, penggunaan petik tunggal (' ') lebih cepat dari pada petik ganda (" "). Petik ganda akan memeriksa keberadaan variabel di dalamnya, sehingga butuh waktu lebih lama.

Lebih lambat

```
$awal = microtime(true);  
$i = 0;  
while($i < 1000) {  
    $tmp[] = "";  
    ++$i;  
  
}  
  
$akhir = microtime(true);  
$waktu = $akhir - $awal;  
echo "Membutuhkan $waktu detik\n";
```

Lebih cepat

```
$awal = microtime(true);  
$i = 0;  
while($i < 1000) {  
    $tmp[] = '';  
    ++$i;  
  
}  
  
$akhir = microtime(true);  
$waktu = $akhir - $awal;  
echo "Membutuhkan $waktu detik\n";
```

Menulis kode PHP dengan Efisien

- Mengurangi perhitungan yang tidak perlu**

Menghitung dan memberikan nilai ke variabel di awal, lebih cepat dari pada menghitungnya beberapa kali dimana perhitungan tersebut diperlukan.

Lebih lambat

```
$awal = microtime(true);  
$arrA = array(1,2,3,4,5,6,7,8,9);  
for( $i=0; i< count($arrA) ; $i++) {  
    echo count($arrA);  
}  
  
$akhir = microtime(true);  
$waktu = $akhir - $awal;  
echo "Membutuhkan $waktu detik\n";
```

Lebih cepat

```
$awal = microtime(true);  
$arrA = array(1,2,3,4,5,6,7,8,9);  
$len = count($arrA);  
for( $i=0; i< $len; $i++) {  
    echo $len;  
}  
  
$akhir = microtime(true);  
$waktu = $akhir - $awal;  
echo "Membutuhkan $waktu detik\n";
```

Menulis kode PHP dengan Efisien

- **Menggunakan function isset()**

Gunakan isset() jika memungkinkan untuk memeriksa lebih besar dari 0.
Hindari menggunakan count(), strlen(), sizeof().

Lebih lambat

```
if(count($returnValue) > 0) {  
    // lakukan sesuatu  
}
```

Lebih cepat

```
if(isset($returnValue)) {  
    // lakukan sesuatu  
}
```

Menulis kode PHP dengan Efisien

- **Menggunakan Swith Case dibanding If**

Menggunakan pernyataan case lebih efisien dibandingkan struktur if/else untuk mengerjakan pekerjaan yang sama.

Menulis kode PHP dengan Efisien

- Menggunakan Swith Case

Lebih lambat

```
if($a == 0){  
    // lakukan tugas a  
}else if($a == 1){  
    // lakukan tugas b  
}else{  
    // lakukan tugas c  
}
```

Lebih cepat

```
Switch ($a){  
    case 0:  
        // lakukan tugas a  
    case 1:  
        // lakukan tugas b  
    default:  
        // lakukan tugas c  
}
```

Mengimplementasikan Kemudahan Interaksi

Desain User Interface (UI) fokus untuk mengantisipasi apa yang mungkin dilakukan pengguna dan memastikan bahwa antarmuka memiliki elemen-elemen yang mudah diakses, difahami dan digunakan untuk memfasilitasi aktivitas pengguna.

Dalam membangun aplikasi yang memiliki kinerja tinggi, kita perlu memperhatikan juga kemudahan aplikasi ketika digunakan oleh pengguna.

Aplikasi yang dibuat harus dipastikan sesuai dengan panduan desain Antarmuka Pengguna (User Interface).

10 Panduan desain antarmuka pengguna Nielsen dan Molich's

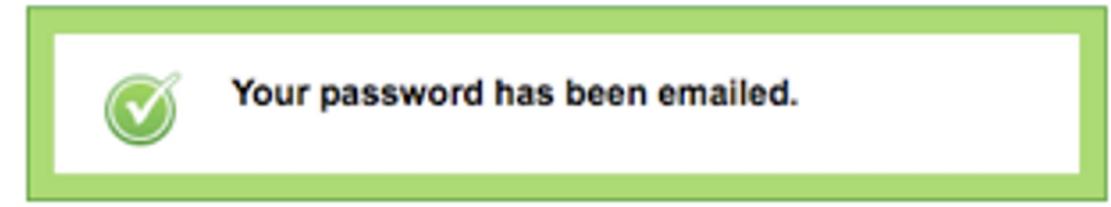
Mengimplementasikan Kemudahan Interaksi

1) Visibilitas status sistem.

Pengguna harus selalu diberitahu tentang status proses yang terjadi. Pemberitahuan harus mudah dimengerti, mudah terlihat di layar, dan ditampilkan dalam waktu yang wajar.



Indikator menunjukkan proses yang terjadi



Theresa Neil sign in

Pesan umpan balik ditampilkan ketika aksi telah dilakukan

Mengimplementasikan Kemudahan Interaksi

1) Visibilitas status sistem.

Type new password:

Six-characters minimum; case sensitive

Password strength: Strong 

Kekuatan sandi ditunjukkan sesuai password yang dimasukkan

Mengimplementasikan Kemudahan Interaksi

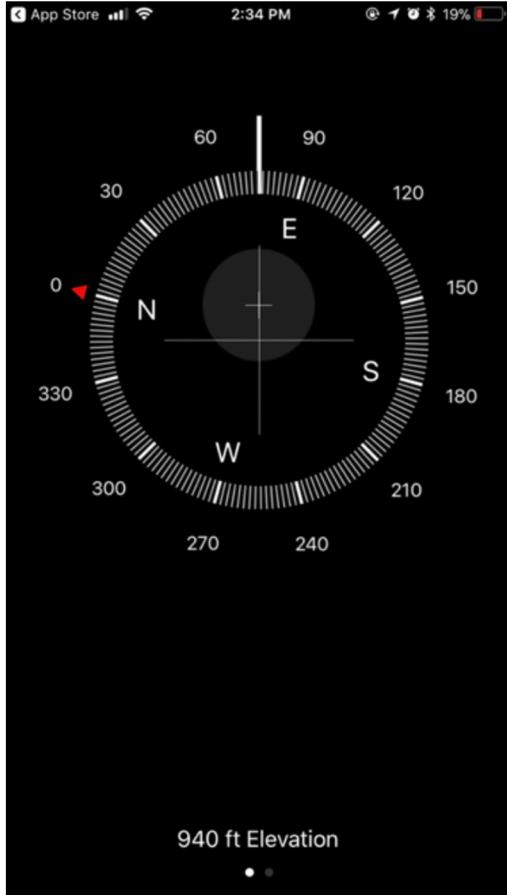
2) Aplikasi sesuai dengan dunia-nyata

Sistem harus menggunakan bahasa, kata-kata, frasa dan konsep yang familiar dengan pengguna.

Menyajikan informasi dalam urutan logis dan mendukung ekspektasi pengguna yang berasal dari pengalaman mereka di dunia nyata akan membuat sistem lebih mudah digunakan.

Mengimplementasikan Kemudahan Interaksi

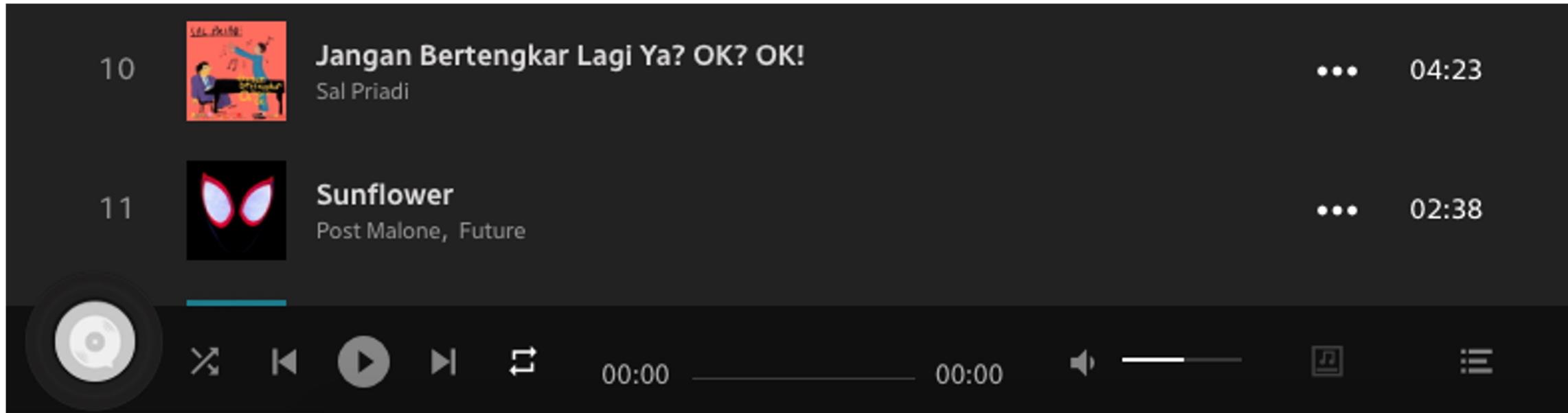
2) Aplikasi sesuai dengan dunia-nyata



Elemen antarmuka pengguna di aplikasi kompas, serupa dengan kompas sebenarnya, akan mempermudah pengguna untuk menggunakan

Mengimplementasikan Kemudahan Interaksi

2) Aplikasi sesuai dengan dunia-nyata

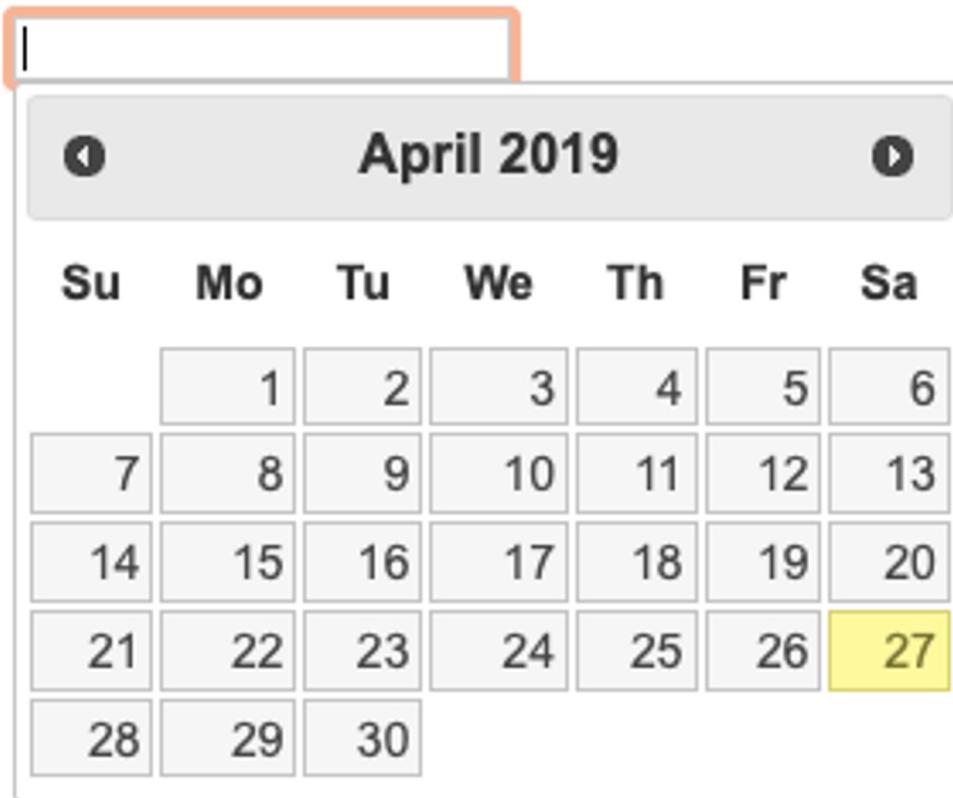


Penggunaan elemen ikon pemutar musik yang standar di perangkat musik akan mempermudah pengguna

Mengimplementasikan Kemudahan Interaksi

2) Aplikasi sesuai dengan dunia-nyata

Date:

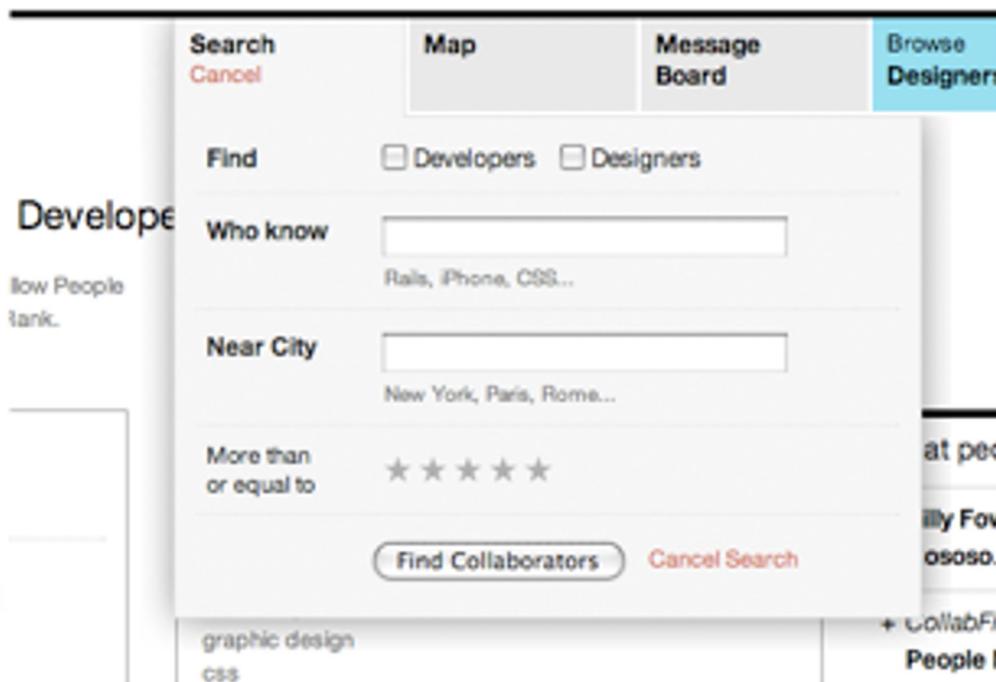


Penggunaan widget kalender dengan tampilan yang sama dengan kalender cetak

Mengimplementasikan Kemudahan Interaksi

3) Kontrol dan Kebebasan Pengguna

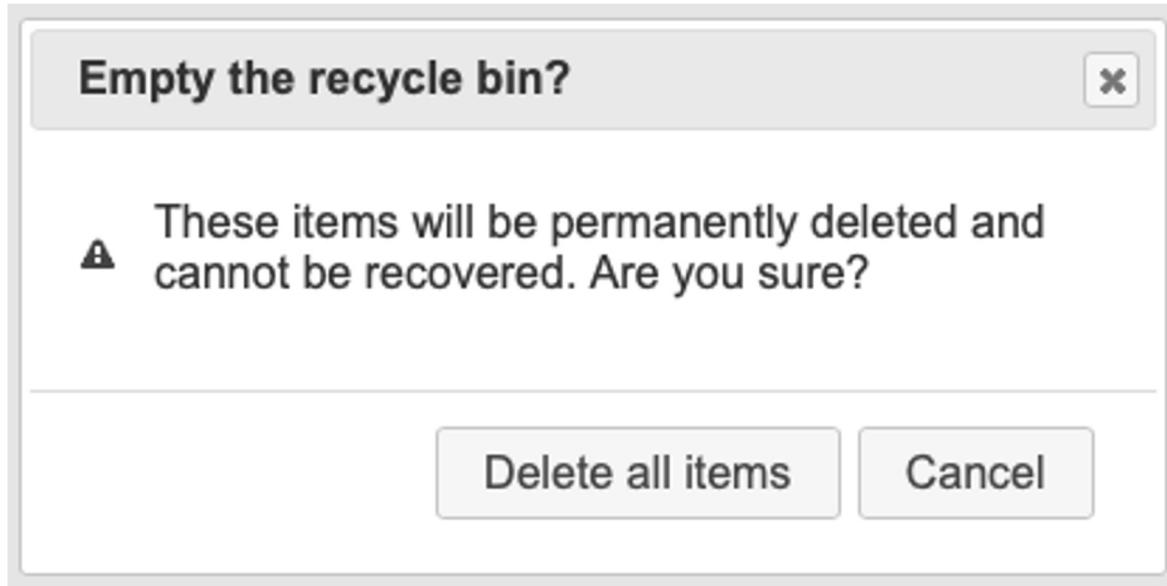
Tawarkan ruang digital kepada pengguna agar memungkinkan langkah mundur, termasuk membatalkan dan mengulangi tindakan sebelumnya.



Fasilitas pencarian mudah dibuka, memasukkan kata kunci & filter, eksekusi atau membatalkan.

Mengimplementasikan Kemudahan Interaksi

3) Kontrol dan Kebebasan Pengguna

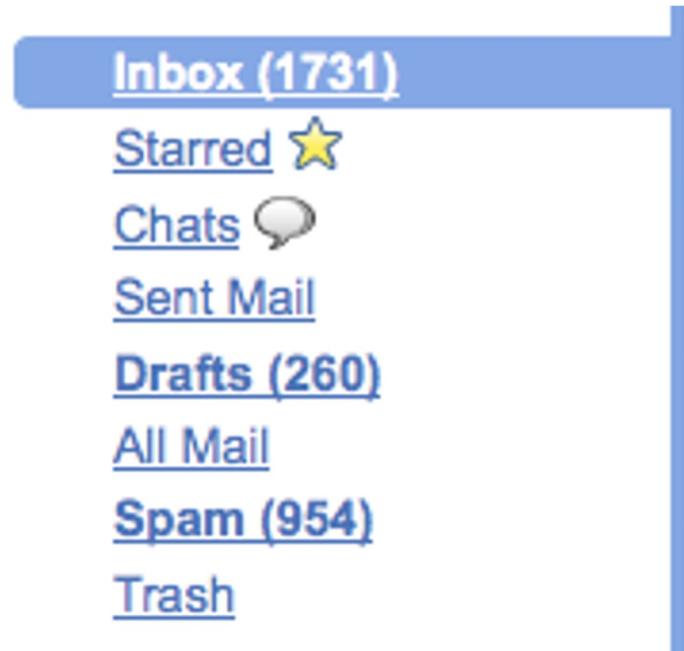


Memberi peringatan kepada pengguna untuk aktifitas yang beresiko. Apakah mau dilanjutkan

Mengimplementasikan Kemudahan Interaksi

4) Konsistensi dan Standar

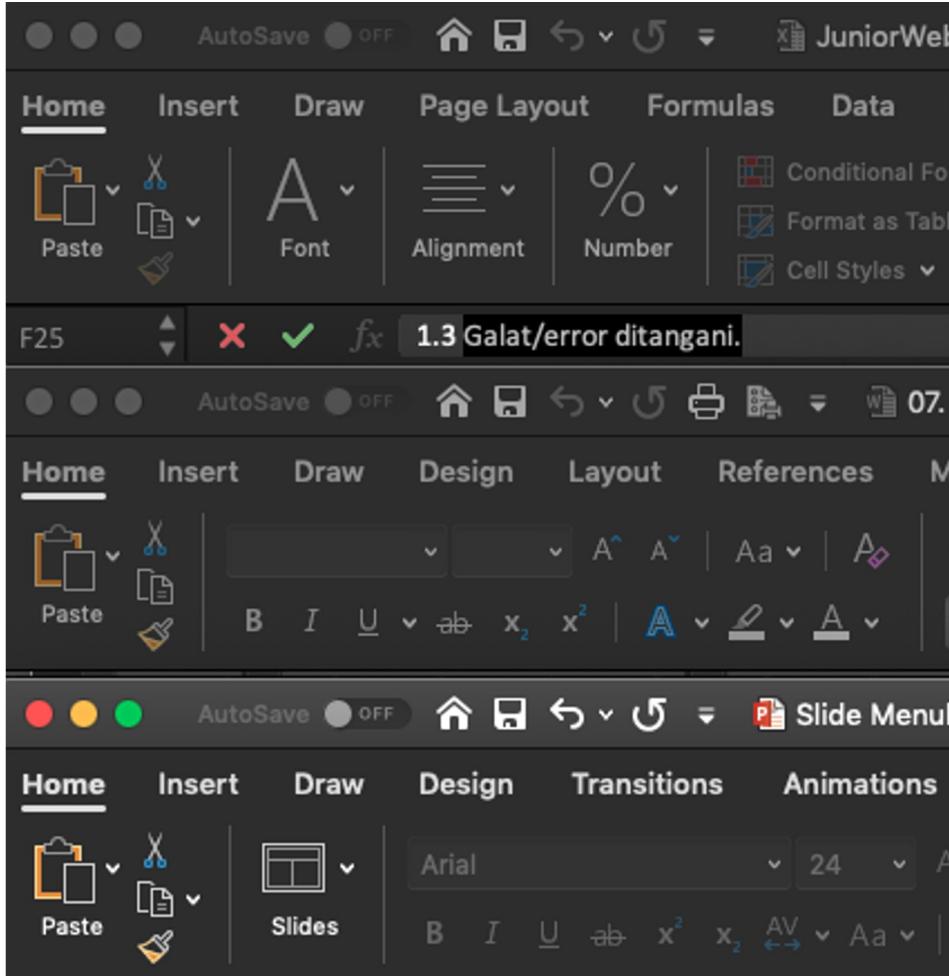
Elemen grafis dan terminologi dipertahankan di seluruh platform yang sama. Misalnya, ikon yang mewakili satu kategori atau konsep tidak boleh mewakili konsep yang berbeda ketika digunakan pada layar yang berbeda.



Gmail menyusun organisasi folder yang sama dengan yang digunakan dalam aplikasi email klien (mis OutLook): Inbox, Draft, Sent Mail

Mengimplementasikan Kemudahan Interaksi

4) Konsistensi dan Standar

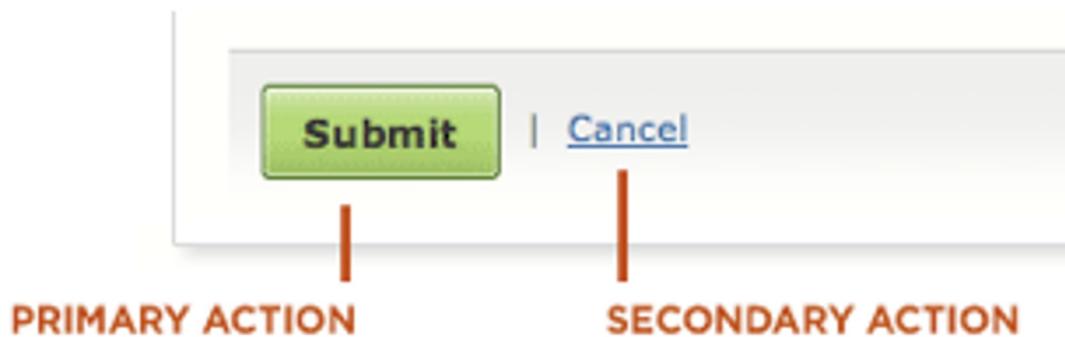


MicrosoftOffice memiliki susunan menu yang serupa untuk aplikasi Word, Excell dan PowerPoint

Mengimplementasikan Kemudahan Interaksi

5) Pencegahan Kesalahan

- Sistem dirancang agar potensi kesalahan dijaga seminimal mungkin.
- Pengguna tidak suka dipanggil untuk mendeteksi dan memperbaiki masalah, yang kadang-kadang berada di luar tingkat keahlian mereka.
- Menghilangkan atau menandai tindakan yang dapat menyebabkan kesalahan.



Buat tindakan utama menjadi menonjol dengan area klik yang lebih besar. Batalkan dan tindakan sekunder hanya ditampilkan sebagai tautan

Mengimplementasikan Kemudahan Interaksi

5) Pencegahan Kesalahan



user inter

user interaction standard

user interface

user interface design

user interface design adalah

user interview

user intervention

user interface perpustakaan online

user interface dan user experience

user interface dalam bahasa pemrograman

user interface android

Rekomendasi otomatis untuk mencegah kesalahan masukan

Mengimplementasikan Kemudahan Interaksi

5) Pencegahan Kesalahan



Fokus otomatis pada input mencegah sumber frustrasi

Mengimplementasikan Kemudahan Interaksi

6) Mengenali daripada mengingat

Minimalkan beban memori pengguna. Buat objek, tindakan, dan opsi terlihat. Pengguna tidak harus mengingat informasi dari satu bagian dialog ke bagian lainnya. Petunjuk penggunaan sistem harus terlihat atau mudah diambil kapan pun diperlukan.

Fira Sans AaBbCcDdEeFfGgHhIiJjKkL

Z Y M m > Fira Sans > Mozilla > 32 Styles > DOWNLOAD OTF

Corbert AaBbCcDdEeFfGgHhIiJj

Z Y M m > Corbert > The Northern Block > 2 Styles > DOWNLOAD OTF (OFFSITE)

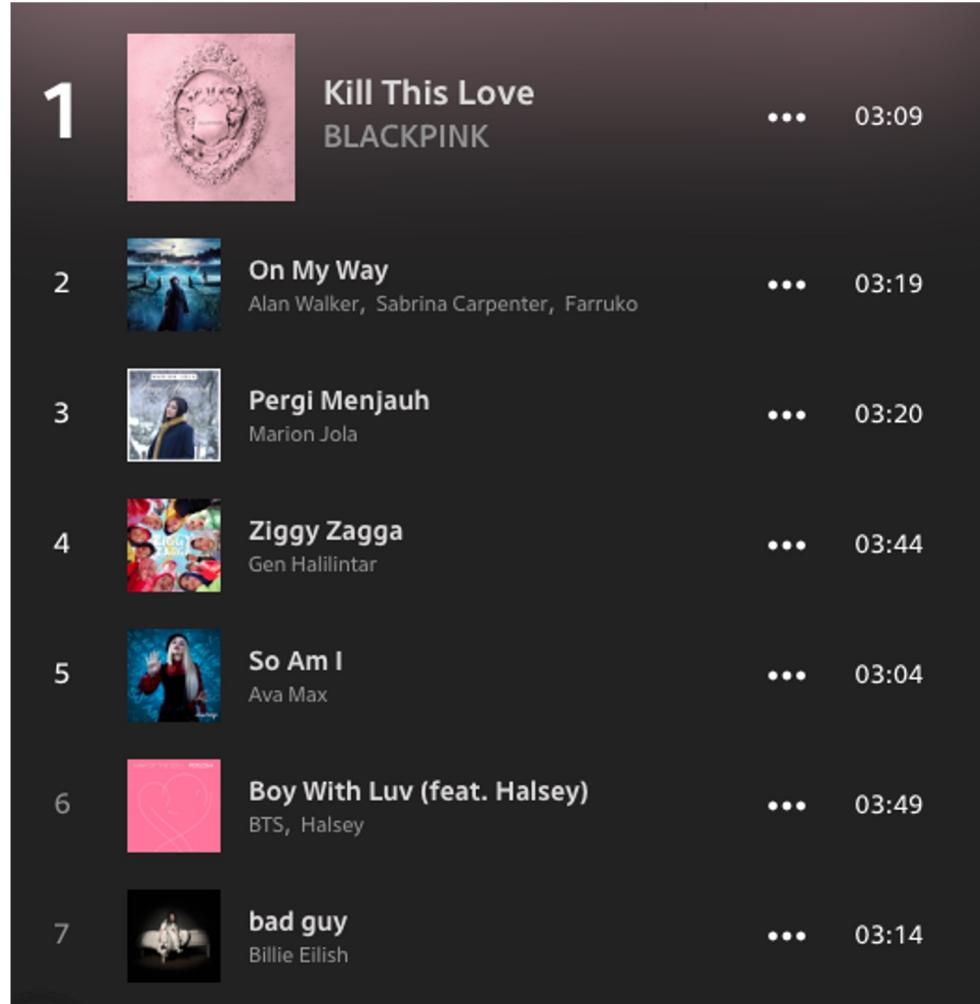
Alex Brush AaBbCcDdEeFfGg

Z Y M m > Alex Brush > TypeSETit > 1 Style > DOWNLOAD OTF

Pratinjau font yang dapat dipilih, bukan hanya nama font.

Mengimplementasikan Kemudahan Interaksi

6) Mengenali daripada mengingat



Pratinjau gambar lebih mudah dikenali oleh user.

Mengimplementasikan Kemudahan Interaksi

7) Fleksibilitas dan efisiensi penggunaan

- Interaksi yang lebih sedikit yang memungkinkan navigasi lebih cepat. Sehingga sistem dapat melayani pengguna yang tidak berpengalaman maupun yang berpengalaman.
- Izinkan pengguna untuk menyesuaikan tindakan yang sering dilakukan.
- Dapat menggunakan singkatan, tombol fungsi, perintah tersembunyi dan fasilitas makro

Mengimplementasikan Kemudahan Interaksi

7) Fleksibilitas dan efisiensi penggunaan

	New	Edit	Delete	
	First name	Last name	Position	Office
<input type="checkbox"/>	Airi	Satou	Accountant	Tokyo
<input type="checkbox"/>	Angelica	Ramos	Chief Executive Officer (CEO)	London
<input type="checkbox"/>	Ashton	Cox	Junior Technical Author	San Francisco
<input type="checkbox"/>	Bradley	Greer	Software Engineer	London

Image copyright datatables.net

Memperbaharui data secara langsung di tabel tanpa perlu melalui aksi lain

Mengimplementasikan Kemudahan Interaksi

7) Fleksibilitas dan efisiensi penggunaan

Common Shortcuts

Add Action	Return
New Window	⌘N
Synchronize with Server	⌃⌘S
Clean Up	⌘K
Planning Mode	⌘1
Context Mode	⌘2
Inbox	⌃⌘1
Quick Entry	⌃Space

Quick Entry's shortcut can be customized in Preferences

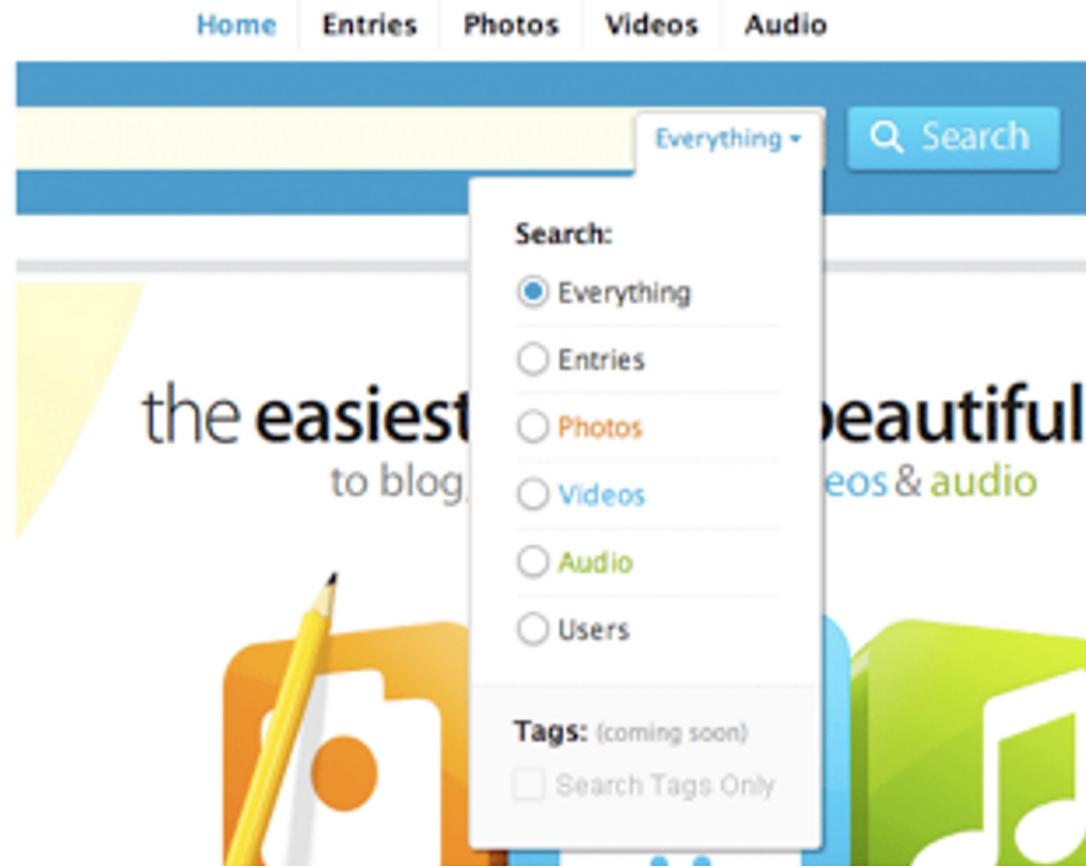
Penggunaan shortcut untuk menjalankan aksi

Mengimplementasikan Kemudahan Interaksi

- **8) Desain estetis dan minimalis**
- Interaksi/infor dengan pengguna dijaga agar tetap minimalis dengan tetap memperhatikan keindahan dan kegunaan
- Tampilan harus dikurangi menjadi hanya komponen yang diperlukan untuk tugas saat ini, sehingga memberikan cara yang jelas dan tidak ambigu untuk menavigasi ke konten lain.

Mengimplementasikan Kemudahan Interaksi

• 8) Desain estetis dan minimalis



Menu pencarian mencontohkan empat prinsip desain visual:

Kontras: teks tebal digunakan untuk dua label dalam pencarian

Pengulangan: teks oranye, biru, dan hijau cocok dengan jenis media

Alignment: alignment kiri yang kuat dari teks, rata bawah drop down

Kedekatan: aturan cahaya digunakan untuk memisahkan tag dari opsi lain

Mengimplementasikan Kemudahan Interaksi

- **8) Desain estetis dan minimalis**

Employee Data						
NAME	POSITION	OFFICE	AGE	START DATE	SALARY	
Airi Satou	Accountant	Tokyo	33	2008/11/28	\$162,700	
Angelica Ramos	Chief Executive Officer (CEO)	London	47	2009/10/09	\$1,200,000	
Ashton Cox	Junior Technical Author	San Francisco	66	2009/01/12	\$86,000	
Bradley Greer	Software Engineer	London	41	2012/10/13	\$132,000	

Padding dan spacing yang cukup menjadikan tabel ini mudah dibaca. Baris header dan konten menggunakan warna yang agak berbeda agar mudah dibedakan

Mengimplementasikan Kemudahan Interaksi

9) Bantu pengguna mengenali, mendiagnosis, dan memulihkan dari kesalahan

- Pesan kesalahan harus diekspresikan dalam bahasa sederhana (tanpa kode), menunjukkan masalah dengan tepat, dan secara konstruktif menyarankan solusi.

Mengimplementasikan Kemudahan Interaksi

9) Bantu pengguna mengenali, mendiagnosis, dan memulihkan dari kesalahan

Or start a new account

Choose a username (no spaces)

bert

⚠ bert is already taken. Please choose a different username.

Choose a password

⚠ Passwords must be at least 6 characters and can only contain letters and numbers.

Retype password

Email address (must be real!)

not an email

⚠ The email provided does not appear to be valid

Send me occasional Digg updates.

Memberikan umpan balik langsung dengan instruksi spesifik

Mengimplementasikan Kemudahan Interaksi

9) Bantu pengguna mengenali, mendiagnosis, dan memulihkan dari kesalahan



Oh no!

It seems the page you were trying to find on my site isn't around anymore (or at least around here).

[Report it missing using my contact form](#) and I'll see what I can do about it.

Whilst you're here why not check out my [articles listing](#) or [browse my blog](#)? You never know - you may just

Menggunakan gambar yang lucu, tetapi memberikan alternatif yang layak (daftar artikel dan tautan blog) dan tindakan (laporkan)

Mengimplementasikan Kemudahan Interaksi

10) Bantuan dan Dokumentasi

- Meskipun lebih baik jika sistem dapat digunakan tanpa dokumentasi, mungkin perlu memberikan bantuan dan dokumentasi.
- Setiap informasi seperti itu harus mudah dicari, difokuskan pada aktifitas pengguna, daftar langkah konkret yang harus dilakukan, dan tidak terlalu besar.

Mengimplementasikan Kemudahan Interaksi

10) Bantuan dan Dokumentasi



Bantuan kontekstual (ini adalah contoh bantuan dalam modul ‘Kolase’) kiat di Picnik jelas dan mudah dinavigasi

Kesimpulan

1. Agar kode program yang kita buat rapi, benar, mudah dibaca dan dipahami baik oleh kita sendiri maupun orang lain, maka penulisan kode program harus mengikuti kaidah yang sudah disepakati oleh konsorsium.
2. Efisiensi kode terhadap penggunaan sumber daya diukur dengan parameter kecepatan eksekusi dan penggunaan memory
3. Efisiensi kode dilakukan dengan tetap memperhatikan faktor kemudahan penggunaan (useability) dan keindahan (estetika)

Referensi

1. <https://gist.github.com/ryansechrest/8138375#3-while-do-while>
2. <https://php7.org/guidelines/psr-1.html>
3. <https://docs.ckan.org/en/ckan-2.7.3/contributing/css.html>
4. <https://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/html.html#H57>
5. <https://github.com/xfiveco/html-coding-standards>
6. <https://www.php.net/manual/en/function.microtime.php>
7. <https://www.interaction-design.org/literature/article/user-interface-design-guidelines-10-rules-of-thumb>
8. <http://designingwebinterfaces.com/6-tips-for-a-great-flex-ux-part-5>

Referensi

1. Mohd. Ehmer Khan - Different Forms of Software Testing Techniques for Finding Error , tahun 2010
2. Retno Hendrowati - Perancangan Pengujian Perangkat Lunak Berorientasi Obyek ,2003
3. Mohd. Ehmer Khan - Different Approaches to White Box Testing Technique for Finding error ,tahun 2011
4. Laurie Williams - White-Box Testing, published in 2006
5. Siegel, Shel., Object Oriented Software Testing an Hierarchical Approach, Canada: John Wiley & Sons Inc. 1996
6. Steven Suehring and Janet Valade – PHP, MySQL, Javascript & HTML5 ALL-IN-ONE FOR DUMMIES, New Jersey 2013
7. <https://www.educba.com/errors-in-website/>
8. <https://www.c-sharpcorner.com/UploadFile/051e29/types-of-error-in-php/>
9. <https://www.niagahoster.co.id/blog/error-404-not-found-pada-website/amp/>

Referensi / Bacaan Lebih Lanjut

- <https://www.javatpoint.com/software-engineering-coding>
- <https://www.geeksforgeeks.org/coding-standards-and-guidelines/>
- <https://hackr.io/blog/programming-paradigms>

Kesimpulan

Kesimpulan

- *Coding Guidelines* adalah acuan bagi *developer* untuk membuat kode program yang lebih mudah dibaca dan dipelihara.
- Paradigma pemrograman adalah cara untuk mengklasifikasikan kode program berdasarkan fitur program yang dibuat.
- Galat/error adalah pesan yang akan muncul sesuai dengan kesalahan pada kode program.

Motivasi

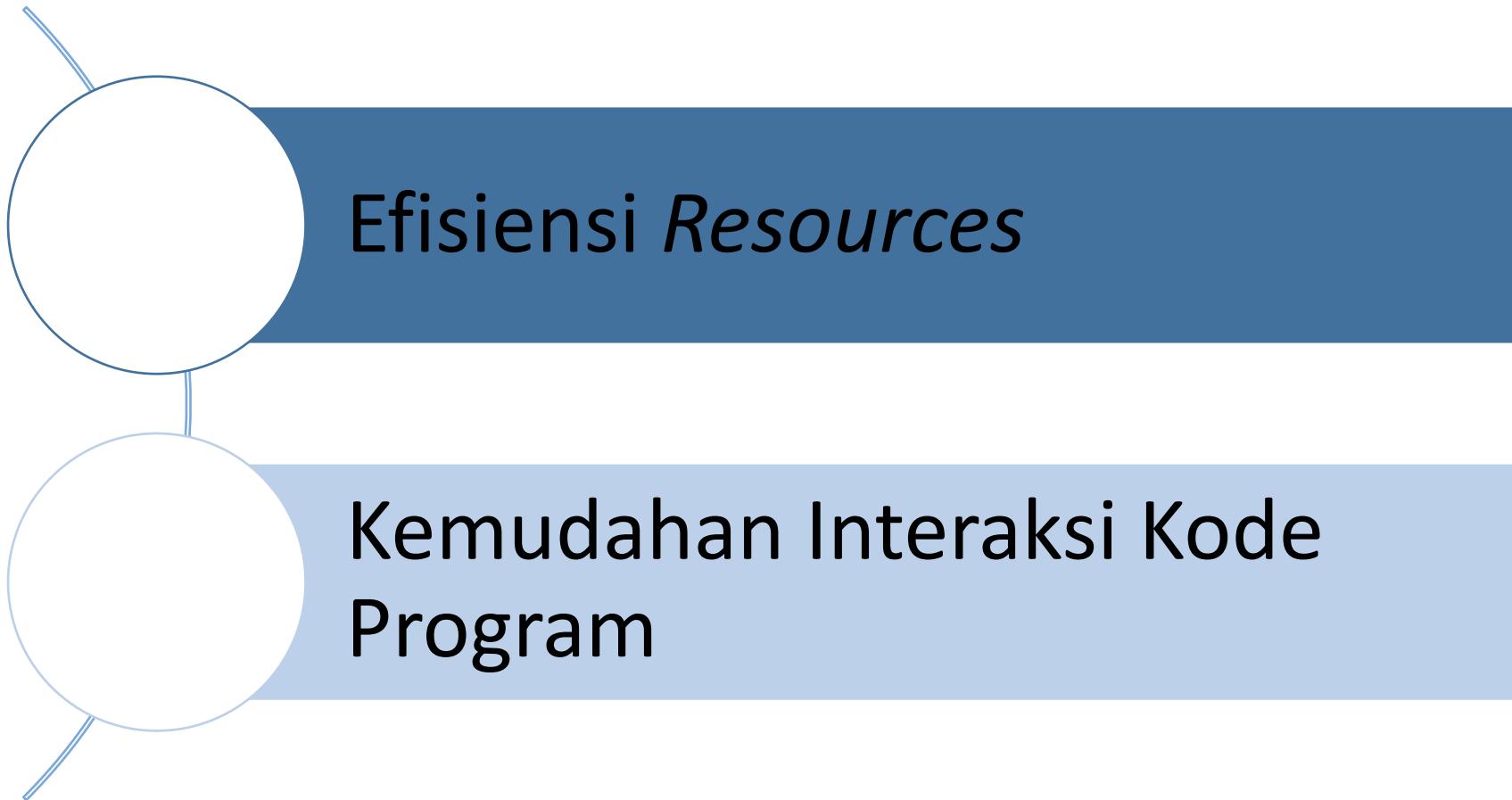


Menggunakan Ukuran Performasi dalam Menuliskan Kode Sumber

Ringkasan Mata Pelatihan

- Unit Kompetensi Acuan: Menulis kode dengan prinsip sesuai *guidelines* dan *best practices*
- Kode Unit Kompetensi Acuan: J.620100.016.01
- Deskripsi singkat: Mata pelatihan ini menentukan kompetensi, pengetahuan dan sikap kerja yang diperlukan dalam menerapkan penulisan kode.
- Tujuan Pembelajaran: Peserta dapat menerapkan penulisan kode yang baik agar kode tersebut dapat dirawat (*Maintainability*).

Agenda



Efisiensi *Resources*

Efisiensi *Resource*

Faktor-faktor yang harus diperhatikan untuk efisiensi:

1. Kemudahan kode program untuk dibaca
2. *Resource* yang dipakai untuk eksekusi kode program

Optimasi kode program dapat dilakukan dengan cara:

1. Penamaan *variable*
2. *Reusable code*
3. Optimasi logika

Penamaan *Variable*

Variable yang digunakan harus mudah dibaca orang lain. *Variable* dapat ditambahkan komentar atau membuat *variable* yang mudah ditebak atau dibaca.

```
$x = "BPPTIK";
$y = "Cikarang, Bekasi";
print $x. ' berlokasi di ' . $y;
```

```
$lembaga = "BPPTIK";
$lokasi = "Cikarang, Bekasi";
print $lembaga. ' berlokasi di ' . $lokasi;
```



Penamaan *variable* menjadi **\$lembaga** dan **\$lokasi** membuat *variable* lebih mudah ditebak dan dibaca

Reusable Code

Kode program yang digunakan berulang kali sebaiknya disimpan dalam bentuk *function* sehingga dapat digunakan kembali untuk keperluan yang lain.

```
$lembaga = "BPPTIK";
$lokasi = "Cikarang, Bekasi";
print $lembaga. ' berlokasi di ' . $lokasi;

$lembaga2 = "Kementerian Kominfo";
$lokasi2 = "Medan Merdeka, Jakarta";
print $lembaga2. ' berlokasi di ' . $lokasi2;
```

```
function location($lembaga, $lokasi){
    print $lembaga. ' berlokasi di ' . $lokasi;
}

location('BPPTIK', 'Cikarang, Bekasi');
location('Kementerian Kominfo', 'Medan
    Merdeka, Jakarta');
```



Function location dapat digunakan berulang kali dengan mengganti isi *variable* yang dibutuhkan

Optimasi Logika

Dari *function* location ingin ditambahakan logika dimana apabila \$lembaga bernilai BPPTIK, maka tulisan BPPTIK yang ditampilkan berwarna biru.

```
function location($lembaga, $lokasi){  
    if($lembaga == 'BPPTIK'){  
        print '<span style="color:blue">' . $lembaga . '</span> : ' . $lokasi;  
    }  
    else{  
        print '<span style="color:black">' . $lembaga . '</span> : ' . $lokasi;  
    }  
  
location('BPPTIK', 'Cikarang, Bekasi');  
echo "<br>";  
location('Kementerian Kominfo', 'Medan Merdeka, Jakarta');
```

BPPTIK : Cikarang, Bekasi

Kementerian Kominfo : Medan Merdeka, Jakarta

```
function location($lembaga, $lokasi){  
    $color = 'black';  
    if('BPPTIK' == $lembaga){  
        $color = 'blue';  
    }  
    print '<span style="color:' . $color . '">' . $lembaga . '</span> : ' . $lokasi;  
}  
  
location('BPPTIK', 'Cikarang, Bekasi');  
echo "<br>";  
location('Kementerian Kominfo', 'Medan Merdeka, Jakarta');
```



Logika *function* dioptimasi dengan hanya satu aksi yang dilakukan, serta **if else** yang berlaku hanya satu saja

Tips: *Source Code* yang Efisien

Berikut adalah beberapa tips untuk membuat program yang lebih efisien:

- **Manajemen *variable***
Gunakan *variable* untuk menyimpan nilai yang akan dipanggil, usahakan tidak ada *variable* yang tidak terpakai.
- **Menyimpan hasil perhitungan kedalam *variable***
Simpanlah hasil perhitungan ke dalam *variable* jika terdapat operasi aritmatika yang dilakukan lebih dari sekali, ini juga berlaku untuk fungsi yang mengembalikan nilai.

Tips: *Source Code* yang Efisien (2)

- **Meminimalisir perulangan bertingkat**

Semakin banyak perulangan bertingkat yang dilakukan, maka akan semakin meningkatkan CPU time. Jika menggunakan perulangan bertingkat, batasi perulangan yang berada paling dalam dan perulangan pertama bernilai lebih besar dibanding perulangan di dalamnya.

- **Menghindari perulangan fungsi**

Jika terdapat perulangan yang memanggil fungsi yang mengembalikan nilai, jadikanlah fungsi tersebut sebuah perulangan.

Kemudahan Interaksi Kode Program

Kemudahan Interaksi Kode Program

Masalah yang muncul dalam pembangunan dan pengembangan program yang dilakukan lebih dari satu orang adalah kode program ditulis sesuai dengan gaya / *style* masing-masing *programmer*. Untuk memudahkan interaksi antar *programmer*, dibutuhkan standar atau *coding guidelines* seperti yang sudah dibahas sebelumnya.

Kemudahan Interaksi Kode Program (2)

Manfaat yang didapatkan *programmer* jika kode program yang dibuat mengikuti *coding guidelines*:

- Kode program mudah dibaca
- Konsistensi penamaan *variable*, fungsi dan semua isi kode program
- Penamaan fungsi merepresentasikan isi di dalamnya
- Kode program dapat dimengerti oleh *programmer* lain

Kesimpulan

Kesimpulan

- *Coding guidelines* dibutuhkan untuk mempermudah interaksi antar *programmer* dalam pembuatan atau *maintenance* program.
- Semakin kompleks kode program yang dibangun maka semakin besar resource yang dibutuhkan untuk menjalankan program tersebut.

Referensi / Bacaan Lebih Lanjut

Referensi / Bacaan Lebih Lanjut

- <https://www.kodinggen.com/tips-ngoding-membuat-source-code-yang-efisien/>

Tujuan Pengujian

1. Menjalankan program untuk menemukan error.
 2. Test case yang bagus adalah yang memiliki kemungkinan terbesar untuk menemukan error yang tersembunyi.
 3. Pengujian yang sukses adalah yang berhasil menemukan error yang tersembunyi.
- Pengujian (Testing) adalah instrumen penting dalam pengembangan aplikasi web untuk mendapatkan produk yang berkualitas dan seperti apa yang diharapkan oleh pengguna.

Prinsip Pengujian

- ❖ Harus bisa dilacak hingga sampai ke kebutuhan customer.
- ❖ Harus direncanakan sejak model dibuat.
- ❖ Prinsip Pareto: 80% error uncovered.
- ❖ Dari lingkup kecil menuju yang besar.
- ❖ Tidak bisa semua kemungkinan diuji.
- ❖ Dilakukan oleh pihak ketiga yang independen.

Secara umum dari proses testing adalah melakukan verifikasi, validasi, dan mendeteksi terjadinya error pada aplikasi tersebut. Dari ketiga hal tersebut diharapkan dapat menemukan masalah – masalah atau kesalahan dan dari hasil penemuan tersebut dapat dilakukan suatu pembenahan.

Testability

- ❖ **Kemudahan untuk diuji.**
- ❖ **Karakteristiknya:**

Operability: mudah digunakan.

Observability: mudah diamati.

Controlability: mudah dikendalikan.

Decomposability: mudah diuraikan.

Simplicity: lingkup kecil, semakin mudah diuji.

Stability: jarang berubah.

Understandability: mudah dipahami.

Desain Testing

- ❖ **Black box testing**

- Memastikan fungsional Software berjalan.

- Kesesuaian input dengan output.

- Tidak memperhatikan proses logic internal.

- ❖ **White box testing**

- Pengamatan detail prosedur.

- Mengamati sampai level percabangan kondisi dan perulangan.

Black Box Testing – Equivalence Partitioning

Contoh: Input NIM dalam Sistem Akademik

- Jika dikosongi?
- Jika diisi dengan format yang salah?
- Jika diisi dengan NIM yang benar?

Black Box Testing – Analisa Nilai Batas

1. Menguji untuk input di sekitar batas atas maupun bawah sebuah range nilai yang valid.
2. Menguji nilai maksimal dan minimal.
3. Menerapkan (1 & 2) untuk output.
4. Menguji batas struktur data yang dipakai. Misal ukuran array.

White Box Testing

- Yang dibutuhkan > Source code
- Menguji lebih “dekat” tentang detail prosedur perangkat lunak.
- Yang diselidiki: logical path (jalur logika) perangkat lunak

Mengapa Source Code ?

Dengan source code, dapat dilakukan pengujian tentang:

- Structural Testing process
- Program Logic-driven Testing
- Design-based Testing
- Examines the internal structure of program

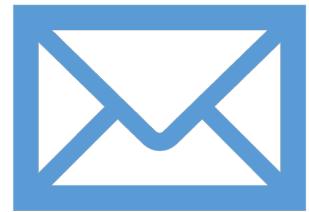
Keuntungan

Menghasilkan program yang benar dan sempurna 100%, karena:

- Mengerjakan seluruh keputusan logika
- Mengerjakan seluruh loop (sesuai batas)
- Menjamin seluruh jalur independen dalam modul dikerjakan minimal 1x
- Mengerjakan seluruh data internal yang menjamin validitas dengan syarat:
 - Mendefinisikan semua logical path
 - Membangun kasus untuk pengujian – Mengevaluasi hasilnya
 - Menguji secara menyeluruh

Mungkinkah dilakukan Pengujian White Box

- ❖ Ya!
- ❖ Tidak dilakukan secara menyeluruh.
- ❖ Cukup dilakukan pada jalur logika yang penting.
- ❖ Kombinasikan dengan black box testing



Kantor:

Balai Pelatihan dan Pengembangan
Teknologi Informasi dan Komunikasi
Kementerian Kominfo

Website: <https://bpptik.keminfo.go.id>

Email: bpptik@keminfo.go.id

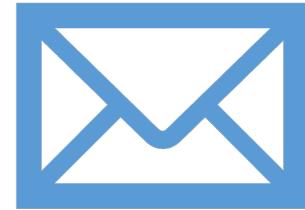
Twitter: @bpptik

Facebook: @bpptik

Instagram: @bpptik

Google Plus: +bpptikkemkominfo

Terima Kasih



Kantor:

Balai Pelatihan dan Pengembangan
Teknologi Informasi dan Komunikasi

Kementerian Kominfo

Website: <https://bpptik.kominfo.go.id>

Email: bpptik@kominfo.go.id

Twitter: @bpptik

Facebook: @bpptik

Instagram: @bpptik

Google Plus: +bpptikkemkominfo