

# Membuat File Readme.txt

- ❖ README adalah sebuah berkas atau file yang berisi informasi tentang berkas lain di dalam direktori atau arsip dari perangkat lunak komputer.
  
- ❖ File Readme berisikan informasi tentang :
  - ✓ Petunjuk Konfigurasi
  - ✓ Petunjuk Instalasi
  - ✓ Penjelasan Struktur/Hirarki Program
  - ✓ Informasi hak cipta dan perizinan menggunakan perangkat lunak
  - ✓ ChangeLog SourceCode

## Coding-Standard HTML

- ❖ Semua kode HTML yang akan anda tulis harus valid dan terstruktur.
- ❖ Sebelum mulai menulis kode program, anda harus memeriksa spesifikasi proyek yang anda kerjakan dengan spesifikasi HTML yang anda gunakan.
- ❖ Spesifikasi yang sekarang biasa digunakan adalah HTML5 Document Type Definition.

```
<!DOCTYPE HTML>
```

## Gunakan Lower Case untuk Nama

Elemen dan atribut harus ditulis menggunakan huruf lower case

```
<!-- Correct -->
```

```
<input name="name" type="text" />
```

```
<!-- Wrong -->
```

```
<input name="name" TYPE="text" />
```

## Tag Penutup

Baik elemen yang kosong maupun yang tidak kosong, harus memiliki tag penutup yang sesuai.

### ❖ Elemen yang tidak kosong

```
<h1>My title</h1>  
<p>Some text</p>
```

### ❖ Elemen yang kosong

```
<span></span>
```

## Tag Penutup (2)

Khusus untuk elemen dengan single tag harus menggunakan > di akhir statement.

```
<br>  
<hr>  

```

## Elemen Bersarang

Elemen bersarang (nested element) harus ditulis dengan benar, sesuai dengan susunan tag-nya.

Penulisan yang benar

```
<!-- Correct -->  
<div>  
  <p>Some text</p>  
</div>
```

Penulisan yang salah

```
<!-- Wrong -->  
<div>  
  <p>Some text</div>  
</p>
```

## Nilai dari Atribut

Menuliskan nilai (value) dari sebuah atribut harus menggunakan tanda petik dua [“ ”], walaupun nilai tersebut berupa nilai numerik.

```
<!-- Correct -->  
<input name="age" type="text" size="3" />  
  
<!-- Wrong -->  
<input name=age type=text size=3 />
```

## Indentasi

- ❖ Gunakan 2 spasi untuk indentasi kode.
- ❖ Gunakan indentasi secara konsisten untuk memudahkan dalam membaca kode.
- ❖ Ketika ada sebuah elemen yang didalamnya terdapat lebih dari satu baris kode, berikan indentasi pada konten elemen, antara tag awal dan tag akhir, untuk memudahkan dalam melihat di mana elemen dimulai dan berakhir.

```
<div class="container">
  <header class="header">
    <h1>Site Name<span></span></h1>
  </header>
  <!-- / header -->
  <hr>
  <nav class="navigation">
    <ul>
      <li><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
    </ul>
  </nav>
  <!-- / navigation -->
</div>
<!-- / container -->
```



## Best-Practice untuk PHP

- ❖ Inisialisasi variable
- ❖ Inisialisasi/ Urutan deklarasi variable, method dan properties

## 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
```

## Coding Standard untuk CSS

- Format CSS
- Pemberian Nama
- Komentar

## Format CSS

- Semua dokumen CSS harus menggunakan dua spasi untuk indentasi dan file tidak boleh memiliki dua spasi tambahan (whitespace).
- Gunakan tanda kutip ganda.
- Gunakan notasi steno hanya jika diperlukan.
- Beri spasi setelah: dalam deklarasi properti.
- Letakkan spasi sebelum {dalam deklarasi aturan.
- Gunakan kode warna hex # 000 kecuali menggunakan rgba ()).

## Format CSS (2)

- Selalu berikan properti fallback untuk browser lama.
- Gunakan satu baris per deklarasi properti.
- Selalu ikuti aturan dengan satu baris spasi.
- Selalu mengutip konten url () dan @import ().
- Jangan membuat indentasi blok.

## Format CSS (Contoh)

```
.media {  
  overflow: hidden;  
  color: #fff;  
  background-color: #000; /* Fallback value */  
  background-image: linear-gradient(black, grey);  
}  
  
.media .img {  
  float: left;  
  border: 1px solid #ccc;  
}  
  
.media .img img {  
  display: block;  
}  
  
.media .content {  
  background: #fff url("../images/media-background.png") no-repeat;  
}
```



## Pemberian Nama

- ❖ Semua nama yang digunakan untuk id, kelas dan atribut harus menggunakan huruf kecil, dan menggunakan tanda hubung apa bila diperlukan pemisahan.

```
/* GOOD */  
.dataset-list {}  
  
/* BAD */  
.datasetlist {}  
.datasetList {}  
.dataset_list {}
```

## Komentar

- ❖ Gunakan komentar untuk menjelaskan apa pun yang memiliki makna ganda atau tidak mudah dipahami orang lain yang membaca kode CSS anda

```
.prose p {  
  font-size: 1.1666em /* 14px / 12px */;  
}  
  
.ie7 .search-form {  
  /*  
    Force the item to have layout in IE7 by setting display to block.  
    See: http://reference.sitepoint.com/css/haslayout  
  */  
  display: inline-block;  
}
```

## Coding Standard untuk PHP

- ❖ Tag PHP
- ❖ Namespace
- ❖ Komentar
- ❖ Keyword Include/Require Once
- ❖ Formatting
- ❖ Fungsi
- ❖ Struktur Kontrol
- ❖ Class

## 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 terbuka pendek sebaiknya TIDAK digunakan.

❖ Benar

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

❖ Salah

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

## Tag Echo Pendek (short echo tag)

Tag gema pendek sebaiknya digunakan dalam file PHP yang berada di dalam HTML (inline).

Masih dapat diterima,

```
< div >  
    < p > <? php echo get_welcome_message (); ?> </ p >  
</ div >
```

Namun sebaiknya:

```
< div >  
    < p > <? = get_welcome_message (); ?> </ p >  
</ div >
```



## 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
```

## 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
```

## 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 berhenti ketika memanggil function die(); dan menampilkan pesan yang dapat dipahami oleh pengguna

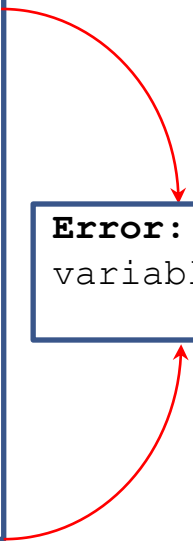
## Metode Menulis Function Penanganan Error Sendiri

- ❖ 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 exception.

## 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 ()` 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 perangkat 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";
?>
```

## 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";

?>
```



## Menggunakan Ukuran Performansi dalam Menuliskan kode sumber

- Menghitung efisiensi program

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

## 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 Switch Case dibanding If

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

### 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  
}
```

## Reference

1. Lemay, L., Coburn, R., and Kyrnin, J. (2016). Sams Teach Yourself HTML, CSS & JavaScript Web Publishing in One Hour a Day, Seventh Edition. Pearson Education, Inc.
3. Anonymous.(n.d.). Apache HTTP Server Documentation Version 2.2. Retrieved from <http://httpd.apache.org/docs/2.2/>.
4. Achour, M., Betz, F. (n.d.), PHP Manual. Retrieved from <http://www.php.net/download-docs.php>.
5. Anonymous. (n.d.). MySQL Reference Manual. Retrieved from <http://downloads.mysql.com/docs/>.
6. Naramore, E., Gerner, J., Le Scouarnec, Y., Stolz, J., Glass, M. K. (2005). Beginning PHP5, Apache, and MySQL® Web Development. Indianapolis, IN: Wiley Publishing, Inc.
7. PHP Tutorial, diakses dari laman <https://www.w3schools.com/php/>, pada 24 Februari 2021
8. Tizag PHP, diakses dari laman <http://www.tizag.com/phpT/comment.php>, pada 24 Februari 2021

## Tools / Lab Online

1. Sublime Text
2. Web Browser (Google Chrome/Mozilla Firefox)



## Summary

- ❖ **Javascript** digunakan untuk membuat website menjadi interaktif
- ❖ **Fungsi** dan **Prosedur** dipakai untuk efisiensi penulisan source code karena dapat digunakan berulang-ulang.
- ❖ **Mengorganisasikan** sumber daya pemrograman memudahkan untuk pemetaan pengembangan program lanjutan maupun kerja tim

## Team Teaching

## Quiz / Games

#JADIJAGOANDIGITAL  
**TERIMA KASIH**



digitalent.kominfo



DTS\_kominfo



digitalent.kominfo



digital talent scholarship