

MATERI PEMBEKALAN

JUNIOR WEB PROGRAMMER



HARI KE-5

| | |
|------------------------|-------------------------------------|
| J.620100.023.02 | Membuat dokumen kode program |
| J.620100.025.02 | Melakukan debugging |



**JUNIOR WEB
PROGRAMMER**

J.620100.023.02

**Membuat Dokumen Kode
Program**

7

MEMBUAT DOKUMEN KODE PROGRAM

Objektif:

1. Melakukan Identifikasi Kode Program
 2. Membuat Dokumentasi Modul Program
 3. Membuat Dokumentasi Fungsi, Prosedur Atau Method Program
 4. Men-Generate Dokumentasi
-

1. Melakukan Identifikasi Kode Program

Komentar adalah baris komentar atau baris keterangan, bukan kolom komentar yang biasa ada di blog. Komentar di perlukan untuk memberikan penjelasan kepada orang lain yang membaca kode kita. Komentar sepenuhnya akan diabaikan oleh **PHP** pada saat eksekusi. Komentar yang baik adalah komentar singkat dan tidak terlalu panjang, namun memberikan penjelasan tentang kegunaan kode atau variabel tersebut dibuat.

Selain sebagai tempat membuat penjelasan, sifat komentar yang tidak akan dieksekusi oleh PHP, memberikan keuntungan lain dalam proses pembuatan program. Beberapa baris kode program bisa diberikan komentar untuk mencari tau penyebab error kode program yang sedang dibuat (proses **debugging**).

PHP menyediakan beberapa cara untuk membuat komentar, dan semuanya berasal dari bahasa pemrograman populer lain seperti C, C++, dan Unix Shell.

1. Metode Komentar Unix Shell

Disebut sebagai metode komentar Unix Shell, karena cara memberikan komentar ini berasal dari sistem Unix. Metode ini

menggunakan karakter tanda pagar atau hash mark (#). PHP akan mengabaikan seluruh text yang terdapat setelah tanda pagar sampai akhir baris atau tag penutup PHP (mana yang terlebih dahulu didapati).

Karena sifatnya yang hanya mempengaruhi satu baris saja, metode komentar Unix Shell efektif digunakan untuk membuat komentar pendek.

```
1  <?php
2  $nilai = $p * exp($r * $t); # menghitung bunga majemuk
3  ?>
```

Beberapa programmer juga sering menggunakan karakter # untuk memisahkan bagian kode PHP dengan bagian lainnya, seperti berikut:

```
1  #####
2  ## Falidasi Form Register
3  #####
4  ... Kode program PHP disini
```

Ketika membuat kode program **PHP** dan **HTML** yang saling berkaitan, Komentar **Unix Shell** ini bisa digunakan seperti contoh berikut:

```
1  <?php $nama = "duniaikom"; # Set $nama menjadi duniaikom ?>
2  <br> <?php echo $nama; ?>
3  ... kode HTML berikutnya
```

2. Metode Komentar C++

Metode komentar ini meminjam cara membuat komentar dari bahasa pemrograman C++. Hampir sama dengan metode komentar Unix Shell, metode komentar C++ ini berlaku hanya untuk sebuah baris atau sampai tag penutup PHP, Tetapi kali ini karakter yang digunakan adalah dua kali garis miring (two slashes), yakni "//".

Karena sifatnya yang sama seperti Unix Shell, semua contoh tanda '#' dapat diganti dengan '//', berikut contohnya:

```

1      $nilai = $p * exp($r * $t); // menghitung bunga majemuk
2
3      //////////////////////////////////
4      // Falidasi Form Register
5      //////////////////////////////////
6      ... Kode program PHP disini
7
8      <?php $nama = "duniaikom"; // Set $nama menjadi duniaikom. ?>
9      <br> <?php echo $nama; ?>
10     ... kode HTML berikutnya

```

3. Metode Komentar C

Jika metode komentar Unix Shell dan C++ efektif untuk membuat komentar pendek, untuk membuat komentar yang panjang, PHP meminjamnya dari bahasa C. Metode komentar ini disebut juga tipe komentar *blok* karena sifatnya yang harus diberikan tanda tutup untuk mengakhiri komentar.

Untuk memulai komentar, kita menuliskan sebuah garis miring dan diikuti dengan tanda bintang (*/**). Semua text setelah tanda tersebut akan dianggap sebagai komentar sampai PHP menemukan tanda tutup, yakni karakter bintang dan diikuti dengan garis miring (**/*). Metode komentar C ini dapat mencakup beberapa baris. Berikut adalah contoh penggunaan Metode Komentar C

```

1      <?php
2      /* Dalam bagian ini kita akan membuat
3      beberapa variabel dan memberikan nilai awal.
4      Nilai awal ini hanya sebagai contoh saja, jadi jangan dianggap serius
5      */
6      $nama = "Andi";
7      $a = 10;
8      $situs = "duniaikom";
9      $b= 2014;
10     ?>

```

Metode Komentar C ini juga berguna untuk "mengomentari" beberapa baris program agar tidak dijalankan oleh PHP, Seperti contoh berikut:

```

1  <?php
2  $a= 3;
3  /*
4  bagian ini tidak akan dijalankan oleh PHP
5  $b = 7;
6  $c = 8;
7  */
8  ?>

```

Namun perlu berhati-hati untuk tidak membuat blok komentar yang saling bertumpuk, seperti kode berikut:

```

1  <?php
2  $a = 12;
3  /*
4  $j = 10; /* Ini adalah komentar */
5  $k = 11;
6  Ini adalah komentar
7  */
8  ?>

```

Pada contoh di atas, PHP akan gagal menjalankan kode program dan menghasilkan error disebabkan komentar yang saling berhimpitan (**overlapping**). Secara garis besar, ada beberapa cara untuk memberi keterangan di dalam script PHP yaitu:

1. Gunakan tag `/*` dan diakhiri dengan tag `*/` apabila jumlah komentar lebih dari satu baris.
2. Gunakan tag `//`; tag ini digunakan dalam komentar yang hanya terdiri dari satu baris saja sehingga penulisan baris berikutnya harus diawali dengan tag `//` kembali
3. Gunakan tag `#`; tag ini juga digunakan dalam 1 baris komentar saja.

Algoritma Sorting

Merupakan algoritma yang menempatkan elemen list pada urutan tertentu. Sorting yang efisien sangat dibutuhkan untuk mengoptimisasi penggunaan dari algoritma lain seperti pencarian dan penggabungan yang membutuhkan list teratur untuk berjalan dengan sempurna, yang juga

sering digunakan untuk Canonicalisasi data dan menghasilkan output yang dapat dibaca manusia.

Shell Sort

Merupakan algoritma yang stau jenis dengan insertion sort, dimana pada setiap nilai i dalam n/i item diurutkan. Pada setiap pergantian nilai, i dikurangi sampai 1 sebagai nilai terakhir.

| | a_1 | a_2 | a_3 | a_4 | a_5 | a_6 | a_7 | a_8 | a_9 | a_{10} | a_{11} | a_{12} |
|------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|----------|
| input data: | 62 | 83 | 18 | 53 | 07 | 17 | 95 | 86 | 47 | 69 | 25 | 28 |
| after 5-sorting: | 17 | 28 | 18 | 47 | 07 | 25 | 83 | 86 | 53 | 69 | 62 | 95 |
| after 3-sorting: | 17 | 07 | 18 | 47 | 28 | 25 | 69 | 62 | 53 | 83 | 86 | 95 |
| after 1-sorting: | 07 | 17 | 18 | 25 | 28 | 47 | 53 | 62 | 69 | 83 | 86 | 95 |

Implementasi:

```

echo "//shell sortn";
$data=array(6,5,3,1,8,7,2,4);
function shell_sort($data){
    $n=count($data);
    $k=0;
    $gap[0]=(int) ($n / 2);
    while($gap[$k]>1){
        $k++;
        $gap[$k]=(int)($gap[$k-1]/2);
    }
    for($i=0;$i<=$k;$i++){
        $step=$gap[$i];
        for($j=$step;$j<$n;$j++){
            $temp=$data[$j];
            $p=$j-$step;
            while($p>=0 && $temp<$data[$p]){
                $data[$p+$step]=$data[$p];
                $p=$p-$step;
            }
            $data[$p+$step]=$temp;
        }
    }
    return $data;
}

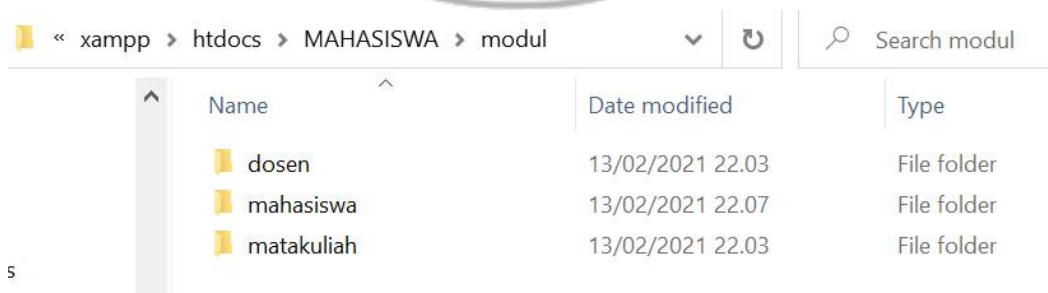
```

```
print_r(shell_sort($data));
```

2. Membuat Dokumentasi Modul Program

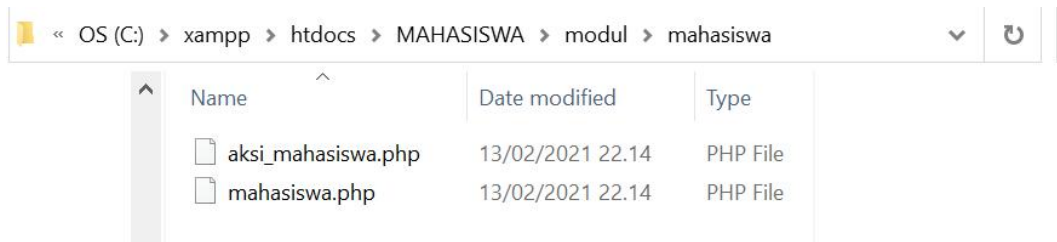
Modul dalam sebuah program PHP bertujuan untuk lebih mudah melakukan manajemen script PHP. Dengan adanya modul memungkinkan seorang programmer untuk melakukan perbaikan pada saat terjadi kesalahan atau pembaharuan script PHP.

Penamaan sebuah modul terserah pada programmer. Membuat modul bisa dilakukan dengan cara membuat folder yang sesuai dengan proses yang ditanganinya. Misalnya proses yang ditanganinya adalah mahasiswa maka modul bisa di tempatkan pada folder transaksi, dan isi folder bisa terdiri dari dua file PHP yaitu dengan contoh tersebut maka bisa dimisalkan isinya adalah mahasiswa.php dan aksi_mahasiswa.php Untuk penjelasan dari isi script kedua contoh file tersebut yaitu mahasiswa.php berisi modul dan aksi_mahasiswa.php berisi aksi untuk menangani proses pada file mahasiswa.php. Gambar 1 adalah contoh tampilan modul dan isi folder modul.



Gambar 1. Contoh Modul dan Isi Folder Modul

Pada contoh gambar di atas, terdapat banyak folder. Isi folder modul pada proses mahasiswa dapat dilihat pada gambar 2.



Gambar 2. Isi Folder Modul pada Proses Mahasiswa

Contoh isi folder modul mahasiswa. Isi file tersebut dipanggil menggunakan beberapa cara. Cara yang bisa digunakan adalah menggunakan switch dan cara yang lain bisa memakai if. Untuk penerapannya tergantung masing-masing lebih yang suka yang mana.

Contoh script pemanggilan modul

Asumsikan saja akan memanggil modul mahasiswa pada sebuah menu. Untuk memanggil dan menampilkan isi modul maka link atau url diarahkan ke modul tujuan. Untuk contoh script pemanggilan modul dan aksi sebagai berikut:

```
1 <a href="?modul=mahasiswa&aksi=tampil">MAHASISWA</a>
```

Pada contoh menu di atas, modul yang dipanggil adalah modul mahasiswa dan aksi yang dilakukan adalah tampil. Berikut adalah contoh pengarahan modul mahasiswa yang dipanggil:

```
1 switch ($_GET['modul'])
2 {
3 case "mahasiswa" :
4     include 'modul/mahasiswa/ mahasiswa.php';
5     break;
6 }
7
```

Script php pemanggilan aksi di bawah ini adalah script yang ada pada file mahasiswa.php.

```
1 switch ($_GET['aksi'])
2 {
3 case "tampil";
4     echo 'disini bisa ditampilkan proses atau memanggil isi tabel database';
5     break;
6 }
7
```

Tujuan pembuatan modul php ini selain membuat lebih mudah manajemen, juga memudahkan programmer lain jika ingin melakukan maintenance koding. Caranya adalah dengan menelusuri kesalahan yang terjadi apabila ada kesalahan langsung pada permodul.

PHP memungkinkan suatu kode yang disimpan dalam suatu file disertakan ke dalam suatu script PHP dengan menggunakan pernyataan include. Hal ini sangat berguna kalau kita mempunyai sederetan kode (misalnya definisi suatu fungsi, definisi suatu konstanta, atau kode yang lain) yang sering digunakan pada berbagai script PHP yang kita buat. Dalam hal ini kita cukup menuliskan kode tersebut sekali saja ke dalam sebuah file, lalu jika memerlukan kode tersebut kita bisa menggunakan fungsi include.

Isi dari PHP file dapat disertakan dalam file PHP yang lain sebelum server mengeksekusinya. Suatu pemrograman yang baik seharusnya program yang besar dipecah menjadi program – program kecil yang selanjutnya disebut dengan modul. Modul – modul kecil tersebut dapat dipanggil sewaktu – waktu jika diperlukan. PHP juga mendukung konsep modul – modul tersebut yang selanjutnya diberi nama modularitas. Modularitas memiliki makna bahwa kita dapat menyisipkan file / modul lain ke dalam file / modul tertentu. Terdapat dua perintah untuk modularitas dalam PHP yaitu menggunakan include() atau require(). Perintah ini dapat digunakan untuk satu file PHP dimasukkan ke dalam file PHP yang lain.

Modularitas ini akan membantu pengembang untuk mengubah tata letak situs web lengkap dengan sedikit usaha. Jika ada perubahan file yang diperlukan maka hanya perlu mengubah file yang disertakan.

Fungsi Include()

Fungsi Include() mengambil semua teks dalam file tertentu dan menyalinnya ke dalam file yang menggunakan fungsi termasuk. Jika ada masalah dalam memuat file maka fungsi include() menghasilkan/ memberitahukan peringatan dan script akan terus tereksekusi.

Contoh kita akan membuat menu umum untuk website, maka buatlah file menu.php dengan konten berikut.

```
<a href="http://www.universitas.com/index.htm">Home</a> -  
<a href="http://www.universitas.com/xml">ebXML</a> -  
<a href="http://www.universitas.com/ajax">AJAX</a> -  
<a href="http://www.universitas.com/perl">PERL</a> <br />
```

Sekarang buatlah sebagai banyak halaman yang diinginkan dan masukkan file ini untuk membuat header. Sebagai contoh sekarang file test.php dapat mengikuti code seperti berikut.

```
<html>  
  <body>  
    <?php include("menu.php"); ?>  
    <p>Ini adalah contoh untuk menunjukkan bagaimana untuk include file PHP!</p>  
  </body>  
</html>
```

Output dari script di atas dapat dilihat pada gambar 3 berikut.



Home - ebXML - AJAX - PERL

Ini adalah contoh untuk menunjukkan bagaimana untuk include file PHP!

Gambar 3. Output Script Menggunakan Include()

Fungsi Require()

Fungsi `require()` mengambil semua teks dalam file tertentu dan menyalinnya ke dalam file yang menggunakan fungsi tersebut. Jika ada masalah dalam memuat file maka fungsi `require()` menghasilkan kesalahan fatal dan menghentikan eksekusi script.

Perbedaan `include()` dan `required()`

`Include()` akan menyertakan dan mengevaluasi seluruh program yang ada di file yang disertakan. Jika terdapat error pada program yang disertakan, maka error akan ditampilkan di layar, dan jika file yang disertakan ternyata tidak ditemukan (mungkin karena lokasi yang salah atau memang file tidak ada), maka program selanjutnya (setelah `include`) akan tetap dijalankan walaupun ditampilkan error.

`Required()` hampir sama dengan `include()`, perbedaannya hanya terletak pada saat file yang disertakan tidak ditemukan, maka perintah-perintah selanjutnya tidak akan dijalankan.

Jadi ada tidak ada perbedaan dalam `require()` dan `include()` kecuali mereka menangani kondisi kesalahan. Dianjurkan untuk menggunakan fungsi `require()` daripada `include()`, karena script tidak harus terus melaksanakan jika file hilang atau misnamed petugas.

Contoh script dengan `include()` diatas, jika diganti dengan fungsi `require()` akan menghasilkan output yang sama. Jika dua contoh script berikut dimana file tidak ada maka akan didapatkan hasil yang berbeda.

```
<html>
  <body>
    <?php include("xxmenu.php"); ?>
    <p>Ini adalah contoh untuk menunjukkan bagaimana untuk include file PHP!</p>
  </body>
</html>
```

Script akan menghasilkan output sebagai berikut:

Ini adalah contoh untuk menunjukkan bagaimana untuk include file PHP!

Jika script yang sama tersebut diganti dengan fungsi require().

```
<html>
<body>
  <?php require("xxmenu.php"); ?>
  <p>Ini adalah contoh untuk menunjukkan bagaimana untuk include file PHP!</p>
</body>
</html>
```

Perbedaan include(), required(), include_once() dan require_once()

Penggunaan include_once() atau require_once() berarti penyisipan hanya dipanggil sekali saja, walaupun disisipkan beberapa kali di dalam sebuah maka dengan menggunakan fungsi include_once() atau require_once() penyisipan tetap dipanggil sekali saja. Beda dengan include() dan require() yang jika disisipkan beberapa kali pada sebuah file maka akan menyebabkan error tau redeclare (deklarasi ganda). Untuk cara penulisan include_once() dan require_once() juga sama seperti penulisan include() dan require().

```
1 include_once('tes.php');
```

atau

```
1 require_once('tes.php');
```

Documentation Generator

Berbicara mengenai dokumentasi, tentu saja sebagai programmer yang baik harus bisa mendokumentasikan segala informasi terkait dengan program yang sudah dibuat. Memang tidak ada definisi dokumentasi seperti apa yang baik dan harusnya dibuat oleh seorang programmer, tetapi dalam kasus documentation generator, sudah ada beberapa orang yang mencetuskan untuk membuat generator (otomatis akan menghasilkan) dokumentasi untuk kode yang sudah dibuat.

JSDoc akan membantu kita dalam menghasilkan dokumentasi yang mudah dilihat dengan format web. Untuk melakukan instalasi, caranya cukup mudah, silahkan mengetikkan kode berikut:

```
npm install jsdoc
```

Setelah berhasil menginstall JSDoc, maka langkah selanjutnya adalah dengan mengetikkan dokumentasi yang sesuai pada module Javascript kita. Berikut adalah contoh dokumentasi.

```
/**
 * Module handler untuk employee
 * @module handler/semmployee
 */
/**
 * Menggunakan utils/date untuk parsing tanggal
 */
var date = require('../utils/date');
/**
 * Menggunakan model dalam kasus ini menggunakan Employee model
 */
var model = require('../models');
/**
 * Sequelize untuk memanggil fungsi model yang menggunakan sequelize
 */
var Sequelize = require('sequelize');
/**
 * fungsi yang di export
 */
module.exports = {
  /**
   * Menampilkan halaman create employee
   * @param {Object} req - Akan berisi request dari router
   * @param {Object} res - Akan berisi result dari request
   */
  viewCreateEmployee: function(req, res) {
    res.render('employee-create', {
      message: req.flash('message')
    });
  },
  /**
   * Menampilkan halaman list employee
   * Menggunakan model Employee untuk query semua employee yang ada
   * @param {Object} req - Akan berisi request dari router
   * @param {Object} res - Akan berisi result dari request
   */
  viewListEmployee: function(req, res) {
    model.Employee.findAll()
      .then(function(employees) {
        res.render('employee-list', {
          employees: employees,
          message: req.flash('message')
        });
      });
  }
};
```

```
});  
},
```

Setelah itu, cukup jalankan perintah berikut:

```
jsdoc namafile.js
```

Maka akan dihasilkan sebuah folder out yang didalamnya terdapat file HTML yang ketika diakses akan memperlihatkan hasil sebagai berikut.

Module: handlers/employee

Module handler untuk employee

Source: [employee.js, line 1](#)

module:handlers/employee

fungsi yang di export

Source: [employee.js, line 23](#)

Members

(inner) **date**

Menggunakan utils/date untuk parsing tanggal


Source: [employee.js, line 10](#)

(inner) **model**

Menggunakan model dalam kasus ini menggunakan Employee model

Source: [employee.js, line 14](#)

[Home](#)
Modules
handlers/employee



(inner) Sequelize

Sequelize untuk memanggil fungsi model yang menggunakan sequelize

Source: [employee.js, line 18](#)

Methods

(static) viewCreateEmployee(req, res)

Menampilkan halaman create employee

Parameters:

| Name | Type | Description |
|------|--------|---------------------------------|
| req | Object | Akan berisi request dari router |
| res | Object | Akan berisi result dari request |

Source: [employee.js, line 29](#)

(static) viewListEmployee(req, res)

Menampilkan halaman list employee Menggunakan model Employee untuk query semua employee yang ada

Parameters:

| Name | Type | Description |
|------|--------|---------------------------------|
| req | Object | Akan berisi request dari router |
| res | Object | Akan berisi result dari request |

Source: [employee.js, line 40](#)

Gambar 4. Hasil Pengaksesan Folder Out yang Terdapat File HTML

Fundamental Node.js Design Patterns

Ketika berbicara mengenai design patterns kita akan memikirkan tentang singletons, observers atau factories. Design pattern adalah solusi umum yang dapat digunakan kembali untuk masalah yang sering terjadi.

Singletons

Pattern singleton melakukan restriksi jumlah instantiasi dari sebuah kelas menjadi satu. Membuat singleton dalam Node.js cukup mudah, sebagaimana kode `require` membantu kita. Berikut adalah contohnya.

```
//area.js
var PI = Math.PI;function circle (radius) {
  return radius * radius * PI;
}module.exports.circle = circle;
```

Berapa kali kita menggunakan module ini, kita hanya cukup memanggilnya saja kembali dengan kode `require` .

```
var areaCalc = require('./area');console.log(areaCalc.circle(5));
```


Karena itu, singleton adalah design pattern paling umum yang sering digunakan terutama pada module dalam NPM.

Observers

Sebuah object yang mengelola beberapa dependents/observers dan memberikan notifikasi kepada mereka untuk melakukan perubahan state. Untuk melakukan implementasi observer pattern, dipanggil `EventEmitter`.

```
// MyFancyObservable.js
var util = require('util');
var EventEmitter = require('events').EventEmitter;
function MyFancyObservable() {
  EventEmitter.call(this);
}
util.inherits(MyFancyObservable, EventEmitter);
```

Setelah itu, kita sudah membuat objek yang observable. Selanjutnya kita tinggal menambahkan fungsionalitasnya.

```
MyFancyObservable.prototype.hello = function (name) {
  this.emit('hello', name);
};
```

Setelah ini, objek observable kita bisa emit sebuah event, misalnya:

```
var MyFancyObservable = require('MyFancyObservable');
var observable = new MyFancyObservable();
observable.on('hello', function (name) {
  console.log(name);
});
observable.hello('john');
```

Factories

Pattern factory adalah pattern yang menciptakan dan tidak membutuhkan pembuatan constructor tetapi membutuhkan interface generic untuk membuat objek. Pattern ini sangat berguna ketika pembuatan proses kompleks.

```
function MyClass (options) {
  this.options = options;
}
function create(options) {
  // modify the options here if you want
  return new MyClass(options);
}
module.exports.create = create;
```

Factories juga membuat test menjadi lebih mudah, di mana bisa menginject module yang menggunakan pattern ini.

Dependency Injection

Dependency injection is a software design pattern in which one or more dependencies (or services) are injected, or passed by reference, into a dependent object.

Dalam contoh berikut, membuat `UserModel` yang menggunakan dependency database.

```
function userModel (options) {  
  var db; if (!options.db) {  
    throw new Error('Options.db is required');  
  } db = options.db; return {  
    create: function (done) {  
      db.query('INSERT ...', done);  
    }  
  }  
}  
module.exports = userModel;
```

Kita bisa membuat instance dari `UserModel` dengan:

```
var db = require('./db'); var userModel = require('User')({  
  db: db  
});
```

3. Membuat Dokumentasi Fungsi, Prosedur Atau Method Program

Pengertian Variable Parameter

Variable Parameter adalah sebuah fitur dalam PHP dimana kita bisa membuat fungsi dengan jumlah parameter yang bisa berubah-ubah (*variable*). Umumnya sebuah fungsi membutuhkan parameter yang telah ditentukan sebelumnya, namun dengan beberapa fungsi khusus, PHP membolehkan kita untuk membuat fungsi dengan jumlah parameter tidak dibatasi, bisa 0, 2, 5, bahkan 100 parameter dengan 1 nama fungsi.

Cara Pembuatan Fungsi dengan Variable Parameter

Sebuah fungsi dengan jumlah parameter yang tidak diketahui, mempunyai fleksibilitas yang dapat digunakan untuk kasus-kasus pemrograman khusus.

Sebagai contoh, fungsi `penambahan()`, di mana fungsi ini akan menambahkan seluruh angka yang terdapat di dalam argumennya. Misalkan `penambahan(2,6,8)` akan menghasilkan 16, dan `penambahan(1,2,3,4,5,6)` akan menghasilkan nilai 21. Jika kita ingin fungsi ini mendukung berapapun jumlah argumen, maka fungsi akan menggunakan fitur Variable Parameter.

Untuk membuat sebuah fungsi dengan jumlah parameter yang tidak diketahui, PHP menyediakan 3 fungsi tambahan untuk mengakses argumen yang diinput pada saat fungsi dipanggil. Ketiga fungsi tersebut adalah:

- `func_get_args()`: fungsi ini akan mengembalikan seluruh nilai argumen dalam sebuah fungsi. Hasilnya dalam bentuk array.
- `func_num_args()`: fungsi ini akan mengembalikan banyaknya jumlah argumen dalam pemanggilan fungsi, apakah 1 argumen, 3 argumen, atau 10 argumen.
- `func_get_arg(no_urut_argumen)`: fungsi ini akan mengembalikan nilai dari argumen pada nomor urut yang diberikan kepadanya.

Agar mudah memahami fungsi ketiganya, kita masuk ke dalam kode program:

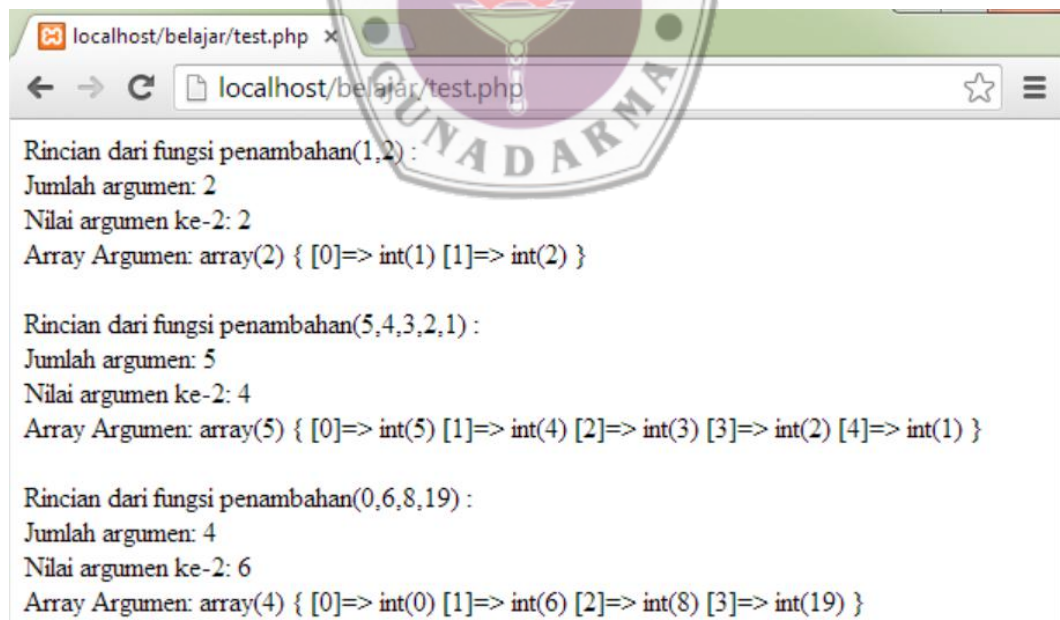
```
1 <?php
2 function penambahan()
3 {
4     //ambil variable parameter
5     $array_argumen = func_get_args();
6     $jumlah_argumen = func_num_args();
7     $nilai_argumen_ke_2 = func_get_arg(1); //index dimulai dari 0
8     //tampilkan hasil variable parameter
9     echo "Jumlah argumen: $jumlah_argumen";
10    echo "<br />";
11
12    echo "Nilai argumen ke-2: $nilai_argumen_ke_2";
13    echo "<br />";
14
15    echo "Array Argumen: ";
16    var_dump($array_argumen);
17 }
```

```

18 echo "<br />";
19 echo "<br />";
20 return;
21 }
22
23 echo "Rincian dari fungsi penambahan(1,2) : ";
24 echo "<br />";
25 penambahan(1,2);
26
27 echo "Rincian dari fungsi penambahan(5,4,3,2,1) : ";
28 echo "<br />";
29 penambahan(5,4,3,2,1);
30
31 echo "Rincian dari fungsi penambahan(0,6,8,19) : ";
32 echo "<br />";
33 echo penambahan(0,6,8,19);
34 ?>
35

```

Gambar 5 berikut adalah tampilan dari script PHP menggunakan 3 fungsi tambahan untuk mengakses argumen



Gambar 5. Output Script PHP Menggunakan 3 Fungsi Tambahan untuk Mengakses Argumen

Pada baris ke-2, didefinisikan fungsi penambahan() tanpa menggunakan parameter. Untuk membuat fungsi variable parameter (di mana jumlah parameternya yang tidak ditentukan) dalam pendefinisian fungsi, dibuat tanpa parameter sama sekali.

Pada baris 5-7, saya menjalankan ke-3 fungsi khusus yang telah dijelaskan sebelumnya. Fungsi-fungsi ini akan mengambil nilai-nilai dari argumen yang diinputkan pada saat pemanggilan fungsi. Lalu nilai ini disimpan kedalam 3 variabel, yakni \$array_argumen, \$jumlah_argumen dan \$nilai_argumen_ke_2

Sebagai catatan, untuk mengambil nilai argumen ke-2, yang nilainya didapat dari fungsi func_get_arg(1). Karena argumen dihitung dari angka 0, sehingga argumen kedua berada di index ke 1. Selanjutnya dari baris 11-20 saya menampilkan hasil masing-masing variabel. Penulisan echo "
" digunakan semata-mata agar tampilan di browser lebih rapi dan mudah dilihat.

Pada pemanggilan fungsi penambahan() pada baris ke 26, 30 dan 34, argumen dipanggil dengan jumlah yang berbeda-beda. Dengan ke-3 fungsi khusus telah sukses di jalankan, untuk membuat fungsi penambahan() yang sebenarnya, tinggal membuat perulangan (looping) untuk menambahkan seluruh argumen-argumen yang ada. Berikut adalah kode program fungsi penambahan versi final:

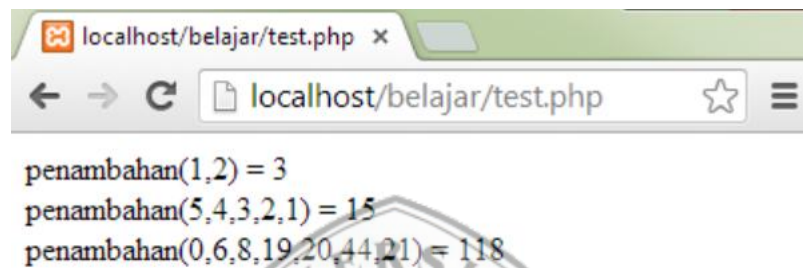
```
1 <?php
2 function penambahan()
3 {
4     //ambil variable parameter
5     $jumlah_argumen = func_num_args();
6
7     //proses penambahan
8     $nilai=0;
9     for ($i = 0; $i < $jumlah_argumen; $i++)
10    {
11        $nilai += func_get_arg($i);
12    }
13    return $nilai;
14 }
15
```

```

16echo "penambahan(1,2) = ".penambahan(1,2);
17echo "<br />";
18
19echo "penambahan(5,4,3,2,1) = ".penambahan(5,4,3,2,1);
20echo "<br />";
21
22echo "penambahan(0,6,8,19,20,44,21) = ".penambahan(0,6,8,19,20,44,21);
23?>

```

Output dari kode program fungsi penambahan dapat dilihat pada gambar 6 berikut:



Gambar 6. Tampilan Output untuk Kode Program Fungsi Penambahan

Fungsi `penambahan()` di atas akan memproses tiap-tiap argumen yang diberikan kepada fungsi tersebut. Perulangan `for` akan memproses perulangan sebanyak argumen yang dimiliki. Dalam pembuatan fungsi PHP, dapat memanfaatkan argumen atau parameter sebagai inputan kedalam fungsi yang dirancang.

Pentingnya Pengecekan Tipe Data Argumen

Dalam pembuatan fungsi PHP, selain merancang cara kerja fungsi, kita juga harus memperkirakan berapa banyak parameter yang dibutuhkan untuk fungsi tersebut. Sebuah fungsi bisa memiliki 1, 2 atau 5 parameter, namun bisa juga tanpa parameter sama sekali.

Tergantung tujuannya, sebuah fungsi umumnya hanya memperbolehkan tipe data tertentu sebagai argumen. Misalnya, untuk fungsi yang berhubungan dengan matematika, biasanya hanya

membutuhkan argumen dengan tipe data angka (integer atau float), dan fungsi penghitung kata, hanya membutuhkan tipe data string sebagai argumen.

Jika fungsi yang dirancang akan digunakan oleh pihak lain, pengecekan tipe data argumen perlu dirancang agar fungsi berjalan sebagaimana mestinya. Jika tipe data parameter tidak sesuai, maka fungsi tidak akan berjalan sebagaimana mestinya, dan biasanya PHP akan mengeluarkan pesan *error*. Cara elegan untuk mengatasi permasalahan ini adalah membuat kode program untuk memeriksa tipe data parameter ini sebelum masuk kepada pemrosesan di dalam fungsi.

Pengecekan tipe data dilakukan pada awal pemrosesan fungsi, dan jika tipe data tidak sesuai, kita bisa membuat pesan bahwa fungsi tidak dapat diproses. Pengecekan apakah suatu argumen merupakan bagian dari tipe data tertentu, dilakukan dengan fungsi khusus yang telah disediakan PHP. Berikut adalah list fungsi pengecekan tipe data dalam PHP:

- `is_array($var)`: fungsi pengecekan apakah tipe data adalah array
- `is_bool($var)`: fungsi pengecekan apakah tipe data adalah boolean
- `is_double($var)`: fungsi pengecekan apakah tipe data adalah float
- `is_float($var)`: fungsi pengecekan apakah tipe data adalah float
- `is_int($var)`: fungsi pengecekan apakah tipe data adalah integer
- `is_integer($var)`: fungsi pengecekan apakah tipe data adalah integer
- `is_long($var)`: fungsi pengecekan apakah tipe data adalah integer
- `is_null($var)`: fungsi pengecekan apakah tipe data adalah null
- `is_numeric($var)`: fungsi pengecekan apakah tipe data adalah angka (integer dan float)
- `is_object($var)`: fungsi pengecekan apakah tipe data adalah objek
- `is_real($var)`: fungsi pengecekan apakah tipe data adalah float

- `is_resource($var)`: fungsi pengecekan apakah tipe data adalah resource (seperti variabel yang menampung koneksi ke database)
- `is_scalar($var)`: fungsi pengecekan apakah tipe data adalah scalar (scalar adalah penyebutan untuk tipe data dasar, seperti integer, float, string atau boolean. Array, object dan resource bukan scalar)
- `is_string($var)`: fungsi pengecekan apakah tipe data adalah string

Cara Pengecekan Tipe Data Argumen Fungsi

Agar lebih mudah dipahami, perhatikan fungsi pangkat() berikut yang dirancang dengan 2 buah inputan atau parameter. Parameter pertama adalah angka yang akan dihitung, dan parameter kedua adalah nilai pangkatnya. `pangkat(2,3)` berarti 2 pangkat 3. `pangkat(2,8)` berarti 2 pangkat 8. Kedua parameter ini harus berupa angka, dan khusus untuk nilai pangkat, harus berupa angka bulat (*integer*). Berikut adalah kode program fungsi pangkat():

```

1 <?php
2 function pangkat($nilai, $pangkat)
3 {
4     if (is_numeric($nilai) AND is_int($pangkat)) //pengecekan tipe data argumen
5     {
6         //Jika argumen sesuai, maka jalankan proses fungsi
7         $hasil=1;
8         for ($i=1;$i<=$pangkat;$i++)
9         {
10             $hasil=$hasil*$nilai;
11         }
12         return $hasil;
13     }
14     else
15     {
16         //Bagian ini akan dijalankan jika tipe data argumen bukan angka
17         return "Tipe data argumen harus berupa angka";
18     }
19 }
20
21 //Test beberapa kasus inputan untuk fungsi pangkat()

```

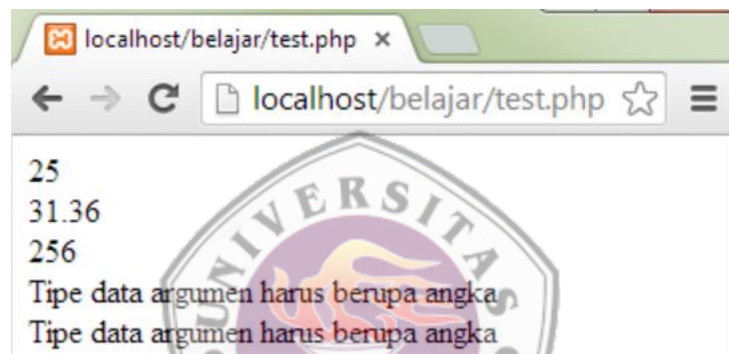


```

22echo pangkat(5,2);
23echo "<br />";
24echo pangkat(5.6,2);
25echo "<br />";
26echo pangkat(2,8);
27echo "<br />";
28echo pangkat(5,2.9);
29echo "<br />";
30echo pangkat("lima",2);
31echo "<br />";
32?>

```

Gambar 7 berikut adalah tampilan output dari kode program fungsi pangkat() script di atas.



Gambar 7. Output Kode Program Fungsi Pangkat()

Fungsi pangkat() dirancang untuk menghitung pangkat dari sebuah angka. Variabel \$nilai dan \$pangkat adalah parameter yang akan menjadi *variabel perantara*.

Pada baris ke-4 dilakukan pengecekan masing-masing parameter di dalam logika IF. Fungsi is_numeric() dan is_int() akan menghasilkan nilai TRUE jika keduanya benar, sehingga keduanya digabungkan kedalam logika AND. Seandainya logika AND ini salah, maka kondisi IF akan bernilai FALSE, dan bagian ELSE akan dijalankan (baris ke-13), yaitu kalimat "*Tipe data argumen harus berupa angka*" untuk memberitahu pengguna fungsi bahwa tipe argumennya harus berupa angka.

Jika kedua kondisi is_numeric() dan is_int() benar, maka dibuat proses perulangan for untuk mencari hasil pemangkatan. Setelah hasilnya

ditemukan, perintah `return` akan mengembalikan nilai tersebut (baris ke-11).

Dari hasil pemanggilan fungsi, kita dapat melihat bahwa logika alur program sudah berjalan benar, dan jika nilai argumen salah, hasil yang ditampilkan bukan kode error PHP, melainkan pesan kesalahan yang lebih informatif.

Dengan menggunakan fungsi seperti `is_numeric()` dan `is_int()` kita dapat melakukan pengecekan tipe data terlebih dahulu sebelum melakukan proses fungsi. Hal ini akan menghindari error program PHP, dan memberikan fleksibilitas untuk melakukan tindakan pencegahan jika tipe data yang diinput bukan yang seharusnya.

Penanganan Exception

Exception merupakan instance dari base class `Exception`, yang dapat kita kembangkan untuk mengenalkan exception kustom kita sendiri. Perlu diperhatikan bahwa penanganan exception itu berbeda dengan penanganan error. Di penanganan error, kita dapat menggunakan fungsi `set_error_handler` untuk mengatur penanganan error kustom kita sehingga ketika error terpicu, dia akan memanggil fungsi penanganan error kustom. Dengan cara itu, kita dapat mengontrol error. Namun umumnya, jenis error tertentu tidak dapat dipulihkan dan memberhentikan pengekseskuan program.

Di sisi lain, exception adalah sesuatu yang dilempar dengan sengaja oleh kode, dan dia diharapkan akan ditangkap di beberapa tempat di dalam aplikasi. Jadi kita dapat mengatakan bahwa exception dapat dipulihkan berlawanan dengan error tertentu yang tidak dapat dipulihkan. Jika sebuah exception yang dilempar itu ditangkap di suatu tempat dalam aplikasi, pengekseskuan program berlanjut dari titik di mana exception ditangkap. Dan exception yang tidak tertangkap di

manapun dalam aplikasi menghasilkan sebuah error, sehingga menghentikan pengeksekusi program.

Alur Kontrol Penanganan Exception

Exception dapat dilempar dan ditangkap dengan menggunakan blok try dan catch. Programmer bertanggung jawab melemparkan exception ketika sesuatu yang tidak diharapkan terjadi. Berikut alur penanganan exception yang ditampilkan di pseudo code.

```
01// code before the try-catch block
02
03try {
04 // code
05
06 // if something is not as expected
07 // throw exception using the "throw" keyword
08
09 // code, it won't be executed if the above exception is thrown
10} catch (Exception $e) {
11 // exception is raised and it'll be handled here
12 // $e->getMessage() contains the error message
13}
14
15// code after the try-catch block, will always be executed
```

Ketika berurusan dengan exception, programmer akan menggunakan pattern yang ditampilkan di potongan kode di atas. Programmer juga dapat menggunakan blok `finally` bersamaan dengan blok `try` dan `catch`. Blok `try` adalah yang digunakan ketika programmer menyangka bahwa kode mungkin menghasilkan exception. Programmer harus selalu membungkus kode tersebut dengan menggunakan `try` dan `catch`.

Melemparkan Exception

Exception mungkin dilempar oleh fungsi yang dipanggil, atau dapat menggunakan kata kunci `throw` untuk melempar exception secara manual.

Contohnya, untuk memvalidasi beberapa input sebelum melakukan operasi apapun, dan melemparkan exception jika data tidak valid.

Perlu diingat bahwa jika melemparkan sebuah exception tetapi tidak mendefinisikan block `catch` yang seharusnya menangani exception tersebut, itu akan menghasilkan fatal error. Jadi pastikan block `catch` selalu didefinisikan jika melemparkan exception pada Aplikasi.

Jika sebuah exception ditangkap oleh block `catch`, objek `Exception` berisi pesan error yang dilempar dengan menggunakan kata kunci `throw`. Variabel `$e` pada contoh di atas adalah instance dari class `Exception`, jadi dia memiliki akses ke semua method class tersebut.

Contoh Exception

Untuk memudahkan pemahaman, akan diperlihatkan contoh dunia nyata untuk mendemonstrasikan penanganan exception di PHP. Asumsikan kita membuat sebuah aplikasi yang memuat konfigurasi dari file **config.php**. Penting bahwa file **config.php** harus ada ketika aplikasi di bootstrap. Jadi, aplikasi tidak dapat berjalan jika file **config.php** tidak ada. Sehingga ini adalah contoh kasus yang sempurna untuk melempar sebuah exception dan memberitahu pengguna bahwa harus memperbaiki masalah ini.

```
01<?php
02try {
03    // init bootstrapping phase
04
05    $config_file_path = "config.php";
06
07    if (!file_exists($config_file_path))
08    {
09        throw new Exception("Configuration file not found.");
10    }
11
12    // continue execution of the bootstrapping phase
13} catch (Exception $e) {
14    echo $e->getMessage();
15    die();
16}
```

17?>

Jika file **config.php** ditemukan, pengeksekusi dilanjutkan secara normal. Di sisi lain, kita akan melempar sebuah exception jika file **config.php** tidak ada. Kita juga bisa memberhentikan pengeksekusi jika ada exception.

Jadi itu adalah bagaimana kita bisa menggunakan exception pada aplikasi. Exception harus dilempar untuk kasus-kasus yang istimewa. Kita tidak perlu melempar exception untuk error-error umum seperti kredensial pengguna yang tidak valid, permisi dari direktori yang tidak sesuai, dll, sesuatu yang akan sering terjadi. Hal-hal tersebut lebih baik ditangani oleh pesan error yang umum di normal alur pengeksekusi aplikasi.

Jadi itu adalah contoh dari penanganan exception menggunakan class `Exception` secara default. Kita juga dapat melakukan penambahan pada class `Exception` dan membuat kustom exception sendiri pada aplikasi.

Bagaimana Membuat Kustom Exception

Di contoh sebelumnya, kita melempar konfigurasi exception dengan menggunakan class `Exception` bawaan. Ini bagus-bagus saja asalkan kita hanya berurusan dengan pesan exception error. Tetapi, terkadang kita mau melakukan sesuatu yang lebih berdasarkan tipe exception yang dilempar. Inilah saat kustom exception berguna. Berikut contoh sebelumnya seperti yang ditampilkan pada potongan kode.

```
01<?php
02class ConfigFileNotFoundException extends Exception {}
03
04try {
05    // init bootstrapping phase
06
07    $config_file_path = "config.php";
08
09    if (!file_exists($config_file_path))
10    {
11        throw new ConfigFileNotFoundException("Configuration file not found.");
```

```

12 }
13
14 // continue execution of the bootstrapping phase
15} catch (ConfigFileNotFoundException $e) {
16 echo "ConfigFileNotFoundException: ".$e->getMessage();
17 // other additional actions that you want to carry out for this exception
18 die();
19} catch (Exception $e) {
20 echo $e->getMessage();
21 die();
22}
23?>

```

Pertama, mendefinisikan class `ConfigFileNotFoundException` yang meng-extend class `Exception` bawaan. Sekarang, itu menjadi kustom exception class dan dapat menggunakannya saat mau melemparkan `ConfigFileNotFoundException` exception di aplikasi.

Selanjutnya, kita menggunakan kata kunci `throw` untuk melempar `ConfigFileNotFoundException` exception jika file **config.php** tidak ada. Perbedaan penting terletak di block `catch`. Mendefinisikan dua block `catch`, dan setiap block digunakan untuk menangkap exception yang berbeda.

Yang pertama menangkap exception bertipe `ConfigFileNotFoundException`. Jadi, jika exception yang dilempar adalah yang bertipe `ConfigFileNotFoundException`, blok ini akan dieksekusi. Jika tipe exception tidak sesuai dengan blok `catch` spesifik manapun, akan masuk ke yang terakhir yang ada untuk menangkap semua pesan exception secara umum.

Blok Finally

Di bagian ini, akan membahas bagaimana menggunakan kata kunci `finally` bersama dengan blok `try` dan `catch`. Terkadang, mau mengeksekusi sepotong kode terlepas dari apakah exception dilempar atau tidak. Inilah di mana dapat menggunakan block `finally`, karena kode

yang kita taruh di block finally akan selalu dieksekusi setelah pengekseskuan blok try dan catch, terlepas dari apakah exception sudah dilempar atau tidak. Untuk memudahkan pemahaman, perhatikan contoh berikut :

```
01 try {  
02 // code  
03  
04 // if something is not as expected  
05 // throw exception using the "throw" keyword  
06  
07 // code, it won't be executed if the above exception is thrown  
08 } catch (Exception $e) {  
09 // exception is raised and it'll be handled here  
10 // $e->getMessage() contains the error message  
11 } finally {  
12 // code, it'll always be executed  
13 }
```

Kode pada contoh di atas kurang lebih sama tetapi dengan tambahan blok `finally` setelah blok `catch`. Seperti yang kita bahas sebelumnya, kode di dalam blok ini akan selalu dieksekusi.

Kasus tipikal yang dapat dipikirkan dari penggunaan blok `finally` secara umum berhubungan dengan pembersihan sumber daya. Contohnya, jika membuka koneksi ke database atau file di disk di dalam blok `try`, dapat melakukan kegiatan pembersihan seperti menutup koneksi di blok `finally` karena dia pasti dijalankan.

Penanganan exception adalah keterampilan koding utama dan harus memikirkan bagaimana exception di tangani ketika membangun aplikasi. Ini akan membantu untuk mendeteksi dan menyelamatkan dari error yang tidak terduga pada aplikasi.

7.4. Men-Generate Dokumentasi

Dokumentasi adalah salah satu syarat untuk memahami kode dan alur kerja dari sebuah aplikasi perangkat lunak. Dengan adanya dokumentasi, para pengembang aplikasi yang terlibat di dalamnya, dapat

lebih mudah memahami alur kerja aplikasi, sehingga dapat membantu proses pengembangan dan juga perbaikan kutu kesalahan (bugs).

Pembuatan dokumentasi dari aplikasi yang dibuat bukan hanya tugas dari *technical writer* yang bertanggung jawab pada aplikasi tersebut, tetapi juga tugas dari para developer-developer yang terlibat di dalamnya. Terlebih lagi dengan semakin rumit dan kompleksnya dari suatu aplikasi, serta mulai ikut sertanya beberapa developer baru yang dilibatkan.

Salah satu cara untuk membuat dokumentasi adalah dengan menulis sebuah blok komentar (block comment) di dalam data kode sumber. Biasanya berbentuk `/***/` di atas fungsi-fungsi di dalam kode sumber. Di dalam blok komentar tersebut, dituliskan dokumentasi dari fungsi-fungsi yang berhubungan di bawahnya.

Cara membuat blok komentar untuk dokumentasi di bahasa pemrograman JavaScript TypeScript, adalah dengan menggunakan panduan standar yang bernama JSDoc. Alat yang dipakai untuk membuat dokumentasi pada latihan ini adalah Webstorm IDE yang dibuat oleh JetBrains/JetBrains IDE. Panduan standar yang seperti JSDoc ini nanti juga dapat digunakan di JetBrains IDE yang lainnya, seperti IntelliJIDEA, Pycharm, PhpStorm, atau Android Studio.

Pertama adalah membuka file TypeScript atau JavaScript yang ingin ditambahkan blok komentar untuk dokumentasi. Buka file yang diberi nama HelloWorldDocs.ts, buat semacam class dan constructor pada file tersebut. Seperti tampilan di bawah ini.


```

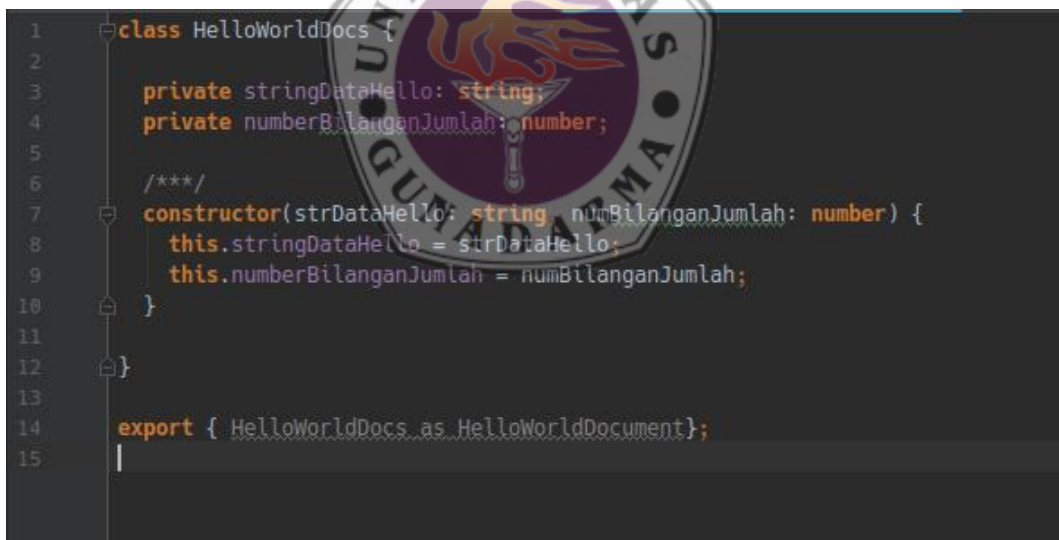
1  class HelloWorldDocs {
2
3      private stringDataHello: string;
4      private numberBilanganJumlah: number;
5
6      constructor(strDataHello: string, numBilanganJumlah: number) {
7          this.stringDataHello = strDataHello;
8          this.numberBilanganJumlah = numBilanganJumlah;
9      }
10 }

```

snippet_artikel_doc1.ts hosted with ❤ by GitHub

[view raw](#)

Pada bagian atas constructor, masukkan blok komentar berupa `/***/`. Cara membuatnya bisa dengan ketik manual atau bisa menggunakan pintasan shortcut **Ctrl + Shift + /** (Linux atau Windows) atau **Cmd + Shift + I** (Mac). Kemudian letakkan kursor pada setelah bintang kedua pada blok komentar tersebut, dan lakukan **Enter**, sehingga blok komentar menjadi berwarna hijau, seperti gambar 8 di bawah ini.



```

1  class HelloWorldDocs {
2
3      private stringDataHello: string;
4      private numberBilanganJumlah: number;
5
6      /***/
7      constructor(strDataHello: string, numBilanganJumlah: number) {
8          this.stringDataHello = strDataHello;
9          this.numberBilanganJumlah = numBilanganJumlah;
10 }
11
12 }
13
14 export { HelloWorldDocs as HelloWorldDocument };
15

```

```

1  class HelloWorldDocs {
2
3      private stringDataHello: string;
4      private numberBilanganJumlah: number;
5
6      /**
7       * Blok komentar
8       */
9      constructor(strDataHello: string, numBilanganJumlah: number) {
10         this.stringDataHello = strDataHello;
11         this.numberBilanganJumlah = numBilanganJumlah;
12     }
13
14 }
15
16 export { HelloWorldDocs as HelloWorldDocument };
17

```

Gambar 8. Membuat Blok Komentar untuk Dokumentasi

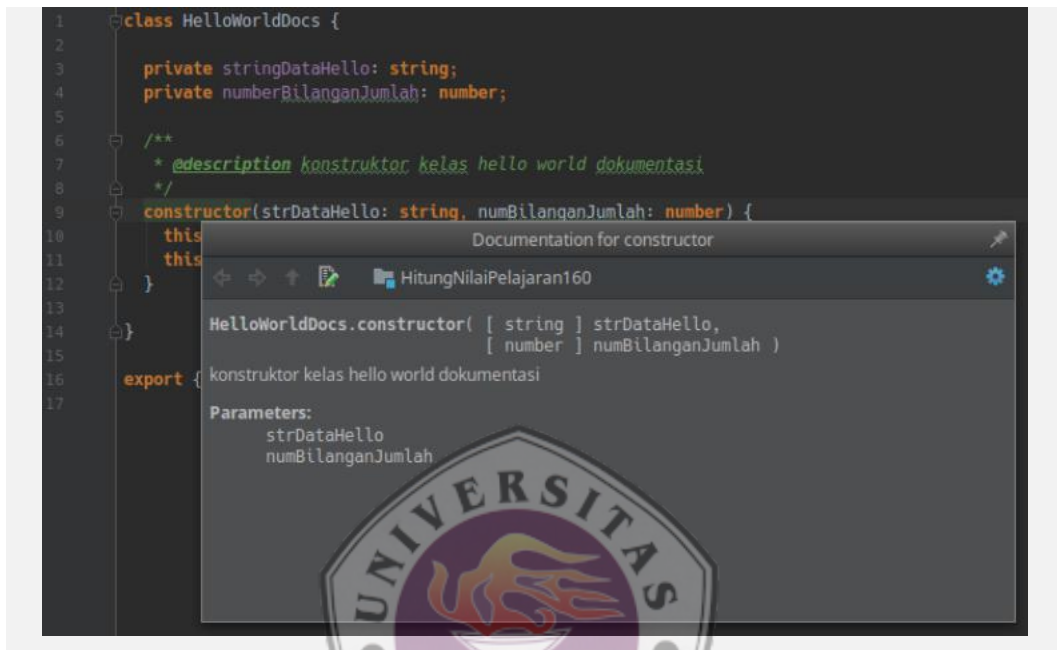
Selanjutnya adalah memasukkan kata-kata untuk dokumentasi pada blok komentar tersebut. Untuk deskripsi dari suatu fungsi atau kelas, gunakan tag “**@description**” di dalam blok komentar tersebut. Misalnya kita tambahkan deskripsi seperti ini: ***@description konstruktor kelas hello world dokumentasi.***

```

1  class HelloWorldDocs {
2
3      private stringDataHello: string;
4      private numberBilanganJumlah: number;
5
6      /**
7       * @description konstruktor kelas hello world dokumentasi
8       */
9      constructor(strDataHello: string, numBilanganJumlah: number) {
10         this.stringDataHello = strDataHello;
11         this.numberBilanganJumlah = numBilanganJumlah;
12     }
13
14 }
15
16 export { HelloWorldDocs as HelloWorldDocument };
17

```

Untuk melihat hasilnya, arahkan kursor ke dalam tulisan constructor, kemudian tekan shortcut Quick Documentation yaitu **Ctrl + Q** (Linux/Windows) atau **F1** (Mac) . Hasilnya akan muncul seperti tampilan gambar 9 di bawah ini.

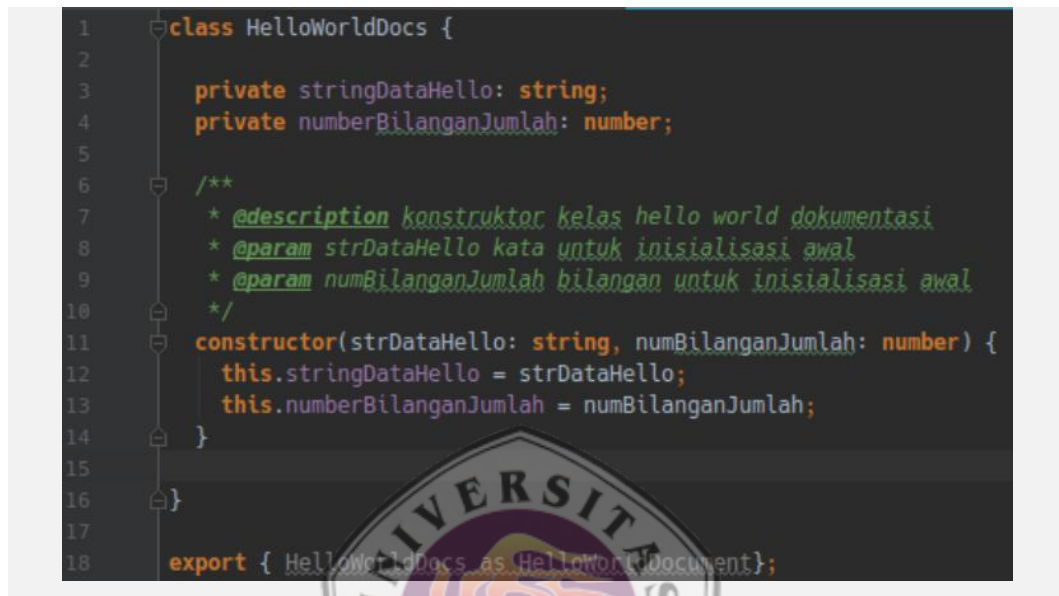


Gambar 9. Hasil Pembuatan Blok Komentar untuk Constructor

Pada tampilan Quick Documentation di atas, tertera keterangan dari constructor yang telah dimasukkan ke dalam blok komentar. Tulisan keterangan di blok komentar muncul di bawah nama constructor kelas yang diberi blok komentar dokumentasi. Pada bagian Parameters, terdapat beberapa parameter yang ada di constructor, namun belum disertai keterangan lebih lanjut.

Berikutnya adalah menambahkan keterangan dari parameter, yang terdapat di dalam constructor ataupun fungsi tertentu. Cara untuk menambahkan keterangan parameter ini adalah dengan menggunakan tag **"@param"** , diikuti dengan nama parameter dan keterangan yang berhubungan dengan parameter itu. Tambahkan keterangan sesuai contoh di bawah ini.

Tambahkan keterangan pada parameter-parameter di konstruktor itu. Jangan lupa untuk *menyamakan nama parameter di dalam fungsi dengan nama parameter yang ada di blok komentar*, sehingga menjadi seperti gambar 10 di bawah ini.



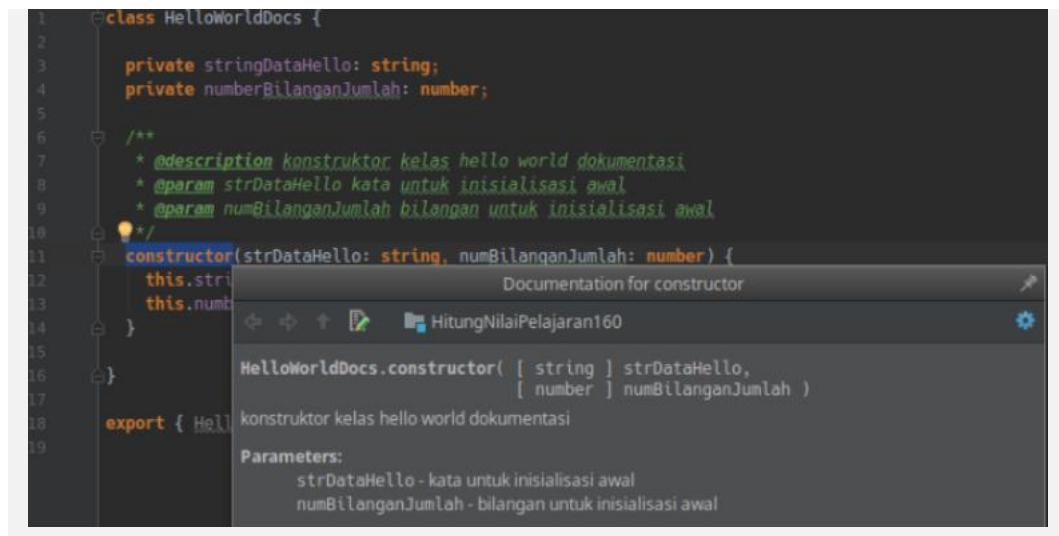
```

1  class HelloWorldDocs {
2
3      private stringDataHello: string;
4      private numberBilanganJumlah: number;
5
6      /**
7       * @description konstruktor kelas hello world dokumentasi
8       * @param strDataHello kata untuk inisialisasi awal
9       * @param numBilanganJumlah bilangan untuk inisialisasi awal
10      */
11     constructor(strDataHello: string, numBilanganJumlah: number) {
12         this.stringDataHello = strDataHello;
13         this.numberBilanganJumlah = numBilanganJumlah;
14     }
15
16 }
17
18 export { HelloWorldDocs as HelloWorldDocument };

```

Gambar 10. Tambahan Dokumentasi di Parameter

Untuk mencoba apakah blok komentar dokumentasi yang dibuat berjalan, arahkan kursor ke tulisan constructor, kemudian tekan **Ctrl + Q** (Linux) atau **F1** (Mac). Hasilnya akan tampak seperti gambar 11 di bawah ini, jika betul memasukkan blok komentar dokumentasinya.



```

1  class HelloWorldDocs {
2
3      private stringDataHello: string;
4      private numberBilanganJumlah: number;
5
6      /**
7       * @description konstruktor kelas hello world dokumentasi
8       * @param strDataHello kata untuk inisialisasi awal
9       * @param numBilanganJumlah bilangan untuk inisialisasi awal
10      */
11     constructor(strDataHello: string, numBilanganJumlah: number) {
12         this.str
13         this.num
14     }
15
16 }
17
18 export { Hell
19

```

Documentation for constructor

HitungNilaiPelajaran160

HelloWorldDocs.constructor([string] strDataHello, [number] numBilanganJumlah)

konstruktor kelas hello world dokumentasi

Parameters:

- strDataHello - kata untuk inisialisasi awal
- numBilanganJumlah - bilangan untuk inisialisasi awal

Gambar 11. Parameter di Dokumentasi dengan Keterangannya

Lalu bagaimana jika ingin menambahkan blok komentar untuk dokumentasi pada fungsi-fungsi, dan bukan konstruktor? Membuat dokumentasi pada fungsi kode tidak berbeda jauh dengan pembuatan dokumentasi pada konstruktor sebelumnya. Cukup gunakan blok komentar `/***/` di atas fungsi yang ingin diberikan dokumentasi. Gambar 12 adalah output blok komentar pada fungsi atau method.



```
18
19
20
21
22
23 getHelloWorld(kataHello: string, jumlahPengali: number) {
24     this.stringDataHello = kataHello;
25     this.numberBilanganJumlah = jumlahPengali;
26
27     const kataHelloReturn = this.stringDataHello + ' ' + this.numberBilanganJumlah;
28 }
29
30
31
32
33
34
35
```



```
16
17 /**
18  *
19  */
20 getHelloWorld(kataHello: string, jumlahPengali: number) {
21     this.stringDataHello = kataHello;
22     this.numberBilanganJumlah = jumlahPengali;
23
24     const kataHelloReturn = this.stringDataHello + ' ' + this.numberBilanganJumlah;
25 }
26
27
28
29
30
31
```

Gambar 12. Menambahkan Blok Komentar pada Fungsi atau Method

Ada juga cara cepat agar blok dokumentasi tersebut langsung menghasilkan nama-nama parameter di dalam fungsinya. Yaitu dengan mengetik `/**/` terlebih dahulu di atas fungsi tersebut, kemudian tekan **Enter**. Maka blok komentar yang dihasilkan akan bersama parameter di dalam fungsi tersebut, seperti tampilan gambar di bawah ini.

```

16
17  /**
18   *
19   * @param {string} kataHello
20   * @param {number} jumlahPengali
21   */
22  getHelloWorld(kataHello: string, jumlahPengali: number) {
23
24      this.stringDataHello = kataHello;
25      this.numberBilanganJumlah = jumlahPengali;
26
27      const kataHelloReturn = this.stringDataHello + ' ' + this.numberBilanganJumlah;
28  }
29
30
31

```

Blok komentar yang dibuat dengan parameter-parameter langsung. Langkah berikutnya adalah menambahkan deskripsi dan penjelasan dari parameter-parameter tersebut. Isi sesuai kebutuhan namun tetap singkat dan jelas. Gambar 13 berikut adalah contoh dari dokumentasi.

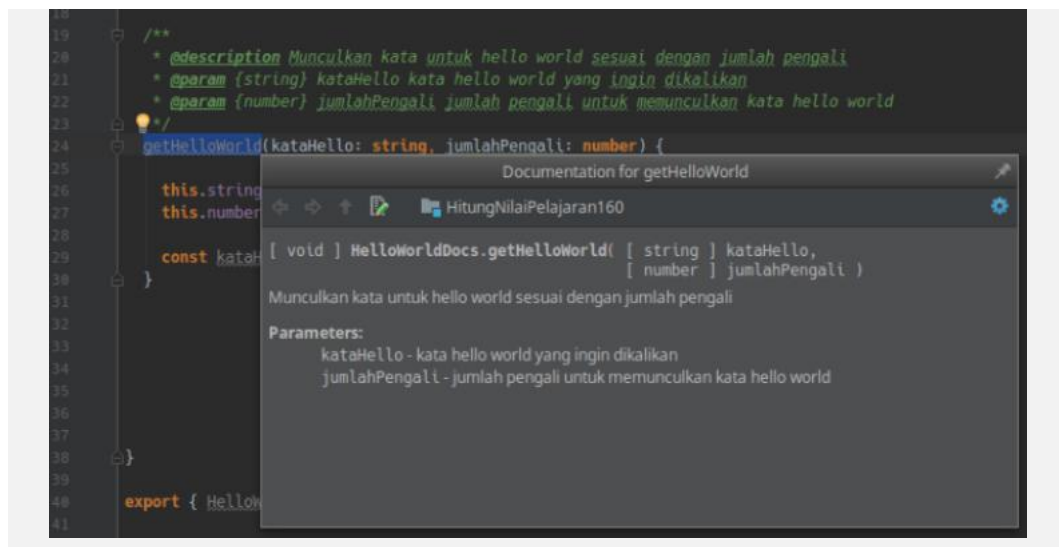
```

17
18
19  /**
20   * @description Munculkan kata untuk hello world sesuai dengan jumlah pengali
21   * @param {string} kataHello kata hello world yang ingin dikalikan
22   * @param {number} jumlahPengali jumlah pengali untuk memunculkan kata hello world
23   */
24  getHelloWorld(kataHello: string, jumlahPengali: number) {
25
26      this.stringDataHello = kataHello;
27      this.numberBilanganJumlah = jumlahPengali;
28
29      const kataHelloReturn = this.stringDataHello + ' ' + this.numberBilanganJumlah;
30  }
31
32
33
34

```

Gambar 13. Contoh Dokumentasi pada Fungsi

Berikutnya adalah periksa apakah format dokumentasi yang ditulis sudah benar atau belum. Yaitu dengan memilih blok nama fungsi (select atau highlight) nama fungsinya, kemudian tekan **Ctrl + Q** (Linux) atau **F1** (Mac) untuk Quick Documentation. Jika tampilan dokumentasi yang muncul seperti gambar 14 di bawah ini, maka cara pembuatan dokumentasi sudah benar.



Gambar 14. Quick Documentation yang Muncul pada Fungsi dengan Blok Komentar yang Benar

Sedangkan potongan kodenya dapat dilihat di bawah ini:

```

1
2  /**
3   * @description Munculkan kata untuk hello world sesuai dengan jumlah pengali
4   * @param {string} kataHello kata hello world yang ingin dikalikan
5   * @param {number} jumlahPengali jumlah pengali untuk memunculkan kata hello world
6   */
7  getHelloWorld(kataHello: string, jumlahPengali: number) {
8
9    this.stringDataHello = kataHello;
10   this.numberBilanganJumlah = jumlahPengali;
11
12   const kataHelloReturn = this.stringDataHello + ' ' + this.numberBilanganJumlah;
13 }
14

```

HelloWorldDocs-3.ts hosted with ❤ by GitHub [view raw](#)

Lalu bagaimana jika ingin menambahkan referensi atau tautan link ke dokumentasi yang berasal dari situs internet lain? Cukup dengan menambahkan sebuah tag parameter `"@see"` di bagian akhir blok komentar. Tag parameter `"@see"` ini ditulis dengan diikuti tautan link situs internet yang menjadi referensi, lalu kemudian sebuah penjelasan singkatnya. Misalnya seperti ini:

"@see <https://googlechrome.github.io/samples/classes-es6/>"

"@see Chrome ES6"

Sehingga blok dokumentasi yang sebelumnya akan menjadi seperti di bawah ini, dengan tambahan tautan link ke dokumentasi luar.

```
1  /**
2   * @description Munculkan kata untuk hello world sesuai dengan jumlah pengali
3   * @param {string} kataHello kata hello world yang ingin dikalikan
4   * @param {number} jumlahPengali jumlah pengali untuk memunculkan kata hello world
5   * @see https://googlechrome.github.io/samples/classes-es6/
6   * @see <a href="https://googlechrome.github.io/samples/classes-es6/">Dokumentasi Chrome ES6</a>
7   */
8  getHelloWorld(kataHello: string, jumlahPengali: number) {
9
10     this.stringDataHello = kataHello;
11     this.numberBilanganJumlah = jumlahPengali;
12
13     const kataHelloReturn = this.stringDataHello + ' ' + this.numberBilanganJumlah;
14 }

```

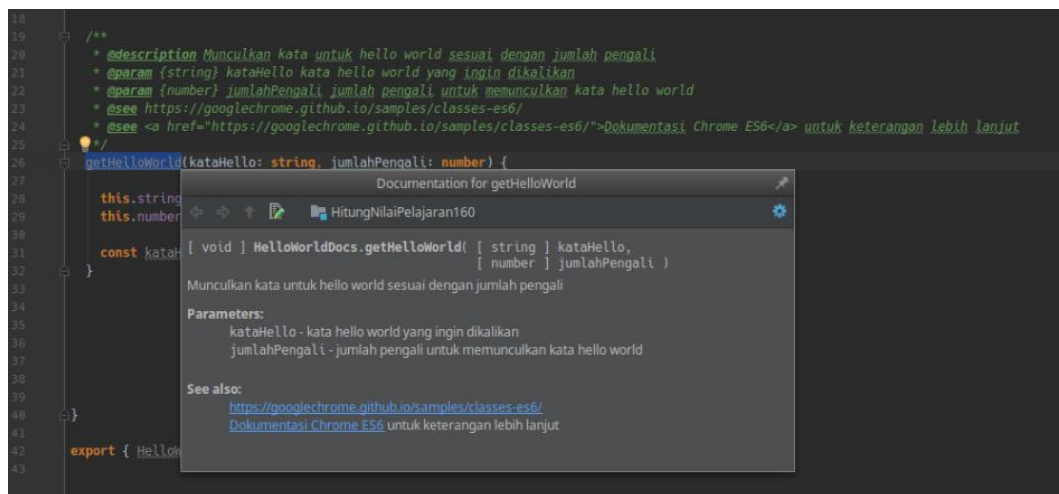
HelloWorldDocs-4.ts hosted with ❤ by GitHub [view raw](#)

Dapat mengecek apakah tag parameter **"@see"** telah berhasil ditambahkan dengan memilih blok nama fungsi (select atau highlight) nama fungsinya, kemudian tekan **Ctrl + Q** (Linux) atau **F1** (Mac) untuk Quick Documentation. Jika tampilan dokumentasi yang muncul seperti gambar 15 dan 16 di bawah ini, maka cara pembuatan dokumentasi sudah benar.

```
17
18
19  /**
20   * @description Munculkan kata untuk hello world sesuai dengan jumlah pengali
21   * @param {string} kataHello kata hello world yang ingin dikalikan
22   * @param {number} jumlahPengali jumlah pengali untuk memunculkan kata hello world
23   * @see https://googlechrome.github.io/samples/classes-es6/
24   * @see <a href="https://googlechrome.github.io/samples/classes-es6/">Dokumentasi Chrome ES6</a> untuk keterangan lebih lanjut
25   */
26  getHelloWorld(kataHello: string, jumlahPengali: number) {
27
28     this.stringDataHello = kataHello;
29     this.numberBilanganJumlah = jumlahPengali;
30
31     const kataHelloReturn = this.stringDataHello + ' ' + this.numberBilanganJumlah;
32 }
33
34

```

Gambar 15 Blok Komentar Dokumentasi dengan Tautan Link Luar



Gambar 16. Blok Komentar Dokumentasi dengan Tautan Link Luar dalam Quick Documentation

Tambahan tag parameter “@see” tadi akan menambahkan sebuah baris dokumentasi baru yaitu “**See also**” yang akan menampilkan rujukan ke tautan link luar yang biasanya berupa halaman web. Jika tautan link berwarna biru tersebut diklik, maka aplikasi Webstorm IDE (dan IDE JetBrains lainnya) akan membuka browser, kemudian halaman web yang dirujuk di Quick Documentation tersebut akan terbuka di browser.

Jika ingin menambahkan tautan link ke situs luar, tetapi tidak di dalam parameter “**See Also**”, melainkan di dalam bagian deskripsi fungsi ataupun di dalam deskripsi parameter. Kita dapat menggunakan tag parameter blok komentar “@link” dengan dirangkap oleh kurung kurawal. Sehingga format tautan linknya menjadi seperti ini:

“{@link https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes| MDN Web Docs}”

atau jika tidak ingin menggunakan alias dapat seperti ini:

“{@link https://30secondsofcode.org}”

Kita tambahkan kedua tambahan link referensi tersebut pada blok deskripsi dan juga pada blok parameter, sehingga tampilan blok dokumentasi kita akan menjadi seperti di bawah ini:

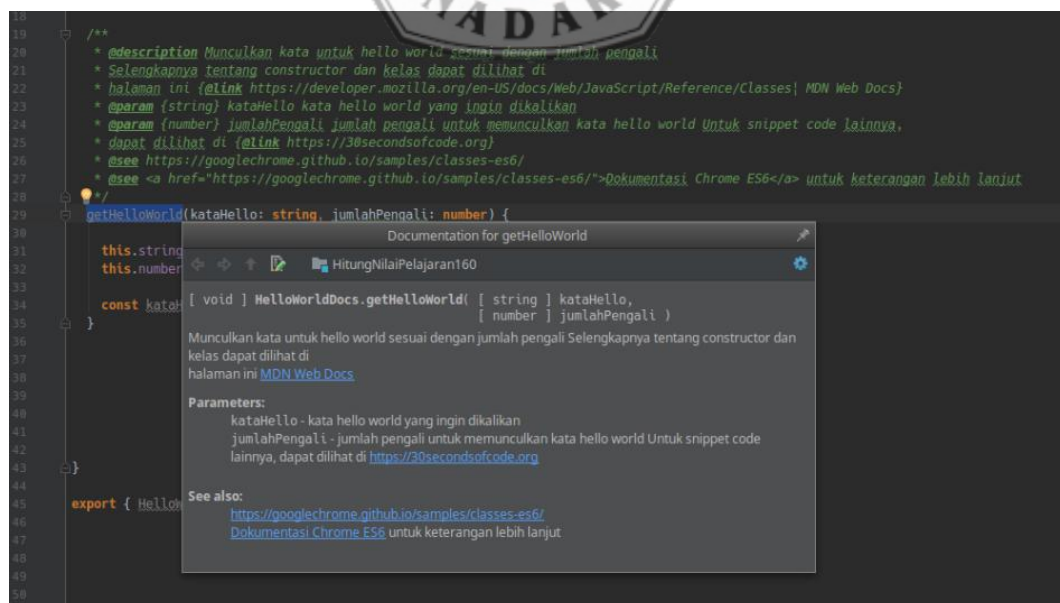
```

1  /**
2   * @description Munculkan kata untuk hello world sesuai dengan jumlah pengali
3   * Selengkapnya tentang constructor dan kelas dapat dilihat di
4   * halaman ini {@link https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes|MDN Web Docs}
5   * @param {string} kataHello kata hello world yang ingin dikalikan
6   * @param {number} jumlahPengali jumlah pengali untuk memunculkan kata hello world Untuk snippet
7   * dapat dilihat di {@link https://30secondsofcode.org}
8   * @see https://googlechrome.github.io/samples/classes-es6/
9   * @see <a href="https://googlechrome.github.io/samples/classes-es6/">Dokumentasi Chrome ES6</a> untuk keterangan lebih lanjut
10  */
11  getHelloWorld(kataHello: string, jumlahPengali: number) {
12
13      this.stringDataHello = kataHello;
14      this.numberBilanganJumlah = jumlahPengali;
15
16      const kataHelloReturn = this.stringDataHello + ' ' + this.numberBilanganJumlah;
17  }

```

HelloWorldDocs-5.ts hosted with ❤ by GitHub [view raw](#)

Kemudian lihat hasilnya dengan menampilkan Quick Documentation, dan hasilnya dapat dilihat di gambar 17 bawah ini:



Gambar 17 Quick Documentation dengan Tambahan Referensi Link Dokumentasi ke Situs Luar

Pada tampilan di atas, dapat dilihat bahwa ada tambahan tautan link ke situs dokumentasi MDN Web Docs di bagian deskripsi fungsi. Selain itu ada tambahan link dokumentasi ke situs internet yang ada di dalam parameter `jumlahPengali`. Jika tautan link tersebut diklik, maka akan terbuka browser dan halaman di dalam tautan link tersebut.

Pada contoh sebelumnya, membuat blok dokumentasi pada sebuah fungsi, tanpa nilai umpan balik. Berikutnya adalah akan membuat blok dokumentasi pada fungsi dengan nilai umpan balik. Kemudian nilai balikkannya diberi sebuah keterangan dokumentasi.

Caranya adalah dengan menggunakan tag parameter **"@return"** di bagian paling bawah setelah tag parameter **"@param"** terakhir. Tag parameter **"@return"** pada blok komentar ini ditulis sambil diikuti oleh tipe balikan dari fungsi tersebut. Bisa berbentuk **"string"**, **"integer"**, **"number"**, tipe array, dan seterusnya. Seperti yang dapat dilihat pada potongan kode di bawah ini.



```

1  /**
2   * @description get hello world dengan nilai balikan string
3   * @param {string} kataHello kata hello world yang ingin dikembalikan
4   * @return {string} data balikan berupa baris kalimat
5   */
6  getHelloWorldReturn(kataHello: string): string {
7
8      this.stringDataHello = kataHello;
9      return 'hello world dengan return ' + kataHello;
10 }
11
12
13
14 /**
15 * @description get hello world dengan nilai balikan array
16 * @param {string} kataHello kata hello world yang ingin dikembalikan
17 * @return {string[]} data balikan berupa array string
18 */
19 getHelloWorldReturnArray(kataHello: string): string[] {
20
21     this.stringDataHello = kataHello;
22     const arrayString: string[] = ['hello world', kataHello, 'return array'];
23     return arrayString;
24 }
25
26
27 /**
28 * @description get hello world dengan nilai balikan array kelas
29 * @param {string} kataHello kata hello world yang ingin dikembalikan
30 * @return {HelloItems[]} Array dari kelas {@link HelloItems} hasil diolah
31 */
32 getHelloWorldReturnArrayClass(kataHello: string): HelloItems[] {
33
34     this.stringDataHello = kataHello;
35     const keySampel = 'hello world key';
36
37     const arrayString: HelloItems[] = [];
38     const helloItem = new HelloItems(this.stringDataHello, keySampel);
39     arrayString.push(helloItem);
40
41     return arrayString;
42 }

```

HelloWorldDocs-6.ts hosted with ❤ by GitHub

[view raw](#)

Potongan kode untuk class HelloItems yang dipakai pada salah satu fungsi di atas.

```

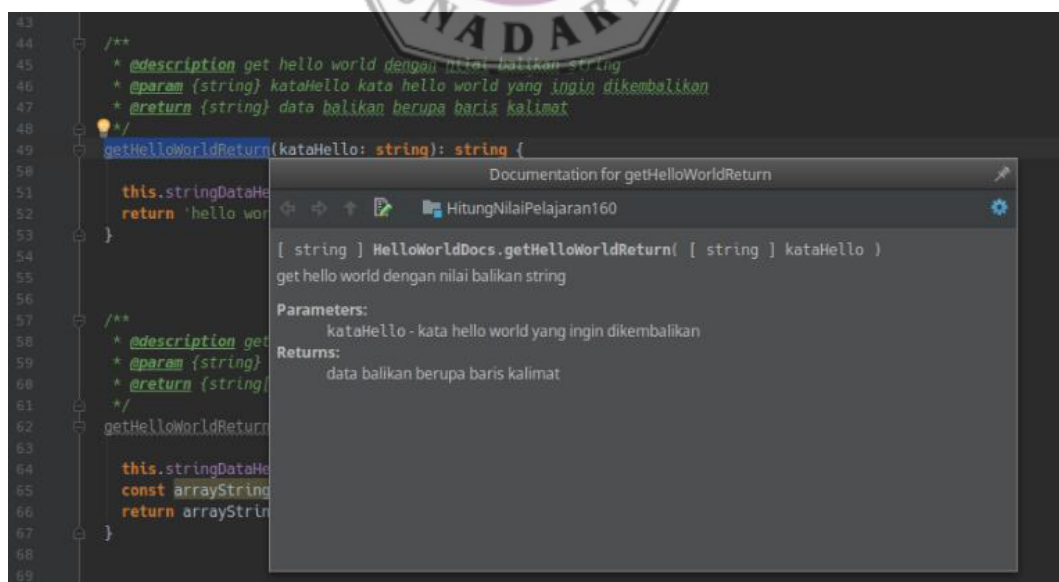
1  /**
2   * kelas Hello Item untuk sampel data kelas
3   */
4   export class HelloItems {
5
6     key: string;
7     value: any;
8
9     /**
10    * @description kelas Hello Items untuk sampel return array
11    * @param {string} datakey nilai data untuk kunci
12    * @param nilaiVal nilai di dalam kata kunci
13    */
14    constructor(datakey: string, nilaiVal: any) {
15      this.key = datakey;
16      this.value = nilaiVal;
17    }
18  }

```

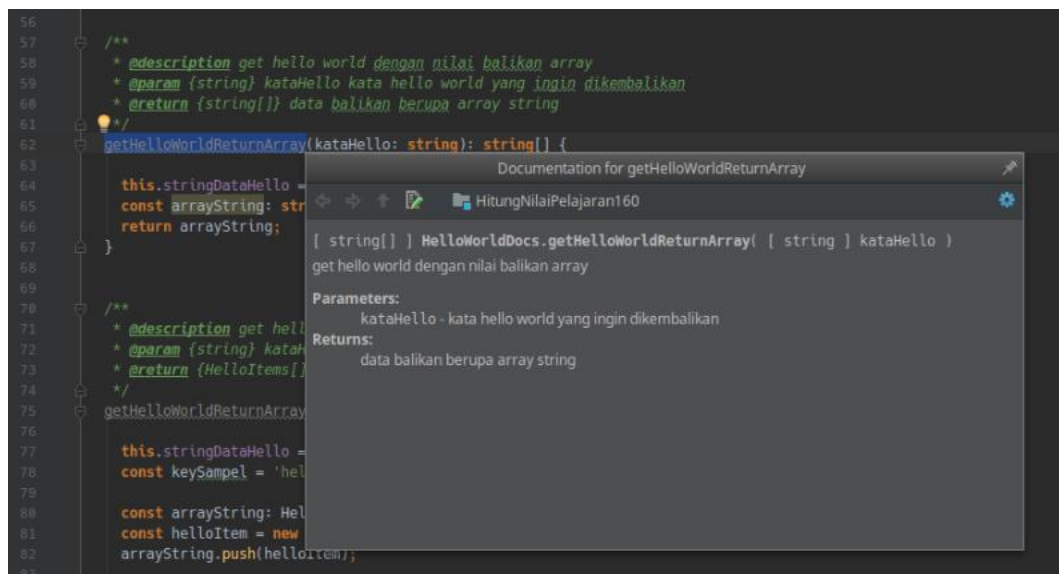
HelloItems.ts hosted with ❤ by GitHub

[view raw](#)

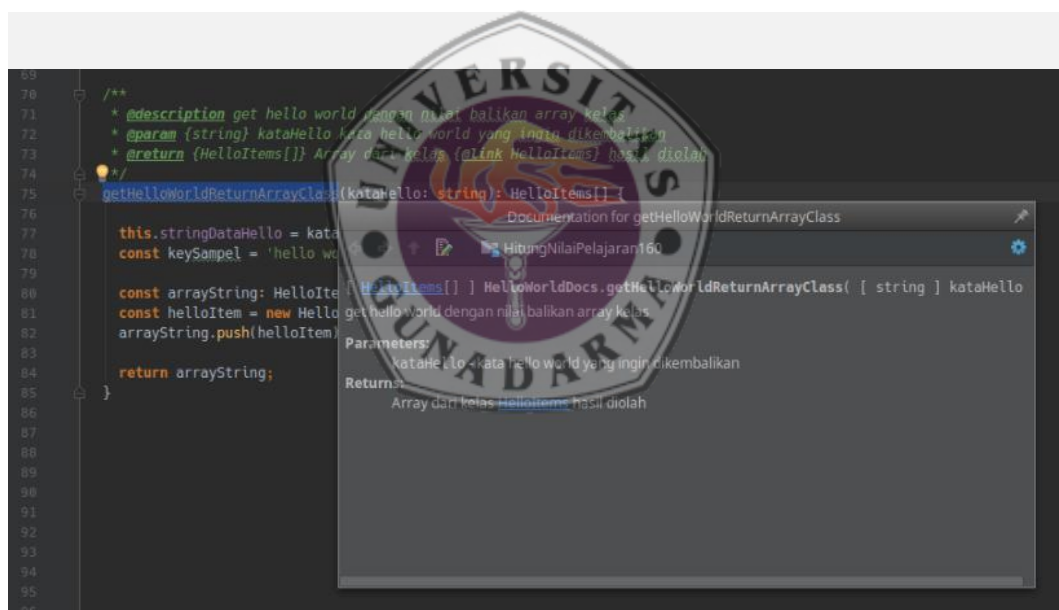
Pada potongan kode di atas, digunakan tag parameter blok dokumentasi “@return” untuk menampilkan dokumentasi nilai balikan dari suatu fungsi. Dapat melihat hasilnya dengan menampilkan Quick Documentation seperti tampilan gambar 18, 19 dan 20 di bawah ini. Di mana di dalam Quick Documentation, akan ada tambahan baris “Returns”.



Gambar 18. Dokumentasi Fungsi dengan Balikan String



Gambar 19. Dokumentasi Fungsi dengan Balikan String Array



Gambar 20. Dokumentasi Fungsi dengan Balikan Class Bentuk Array

Dengan demikian, berikut adalah potongan seluruh kode dari hasil latihan membuat blok dokumentasi di atas.


```

1  import { HelloItems } from './HelloItems';
2
3  class HelloWorldDocs {
4
5      private stringDataHello: string;
6      private numberBilanganJumlah: number;
7
8      /**
9       * @description konstruktor kelas hello world dokumentasi
10      * @param strDataHello kata untuk inisialisasi awal
11      * @param numBilanganJumlah bilangan untuk inisialisasi awal
12      */
13      constructor(strDataHello: string, numBilanganJumlah: number) {
14          this.stringDataHello = strDataHello;
15          this.numberBilanganJumlah = numBilanganJumlah;
16      }
17
18
19      /**
20      * @description Munculkan kata untuk hello world sesuai dengan jumlah pengali
21      * Selengkapnya tentang constructor dan kelas dapat dilihat di
22      * halaman ini {@link https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes}
23      * @param {string} kataHello kata hello world yang ingin dikalikan
24      * @param {number} jumlahPengali jumlah pengali untuk memunculkan kata hello world Untuk snippet
25      * dapat dilihat di {@link https://30secondsofcode.org}
26      * @see https://googlechrome.github.io/samples/classes-es6/
27      * @see <a href="https://googlechrome.github.io/samples/classes-es6/">Dokumentasi Chrome ES6</a>
28      */
29      getHelloWorld(kataHello: string, jumlahPengali: number) {
30
31          this.stringDataHello = kataHello;
32          this.numberBilanganJumlah = jumlahPengali;
33
34          const kataHelloReturn = this.stringDataHello + ' ' + this.numberBilanganJumlah;
35      }
36
37
38      /**
39      * @description get hello world dengan nilai balikan string
40      * @param {string} kataHello kata hello world yang ingin dikembalikan
41      * @return {string} data balikan berupa baris kalimat
42      */
43      getHelloWorldReturn(kataHello: string): string {
44
45          this.stringDataHello = kataHello;
46          return 'hello world dengan return ' + kataHello;
47      }
48
49
50      /**
51      * @description get hello world dengan nilai balikan array
52      * @param {string} kataHello kata hello world yang ingin dikembalikan
53      * @return {string[]} data balikan berupa array string
54      */

```

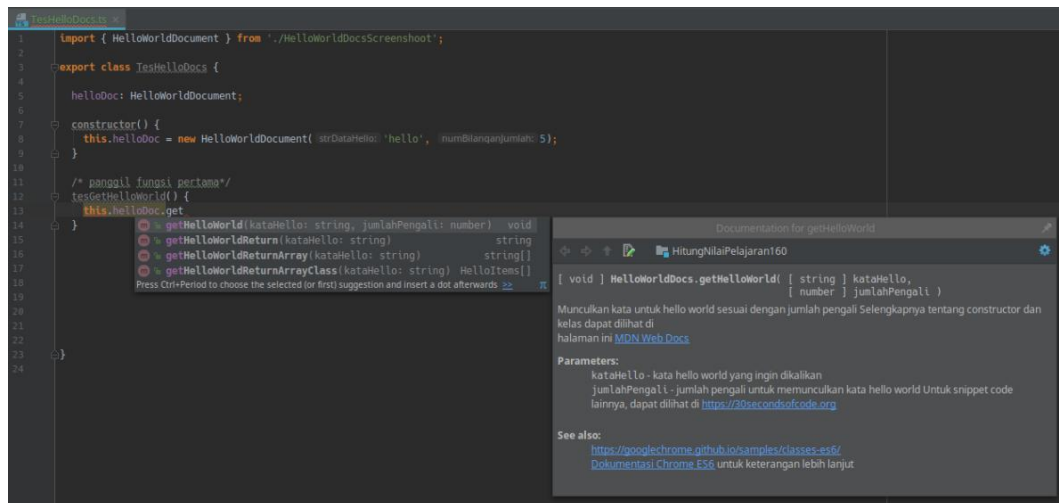
```

55  getHelloWorldReturnArray(kataHello: string): string[] {
56
57      this.stringDataHello = kataHello;
58      const arrayString: string[] = ['hello world', kataHello, 'return array'];
59      return arrayString;
60  }
61
62
63  /**
64   * @description get hello world dengan nilai balikan array kelas
65   * @param {string} kataHello kata hello world yang ingin dikembalikan
66   * @return {HelloItems[]} Array dari kelas {@link HelloItems} hasil diolah
67   */
68  getHelloWorldReturnArrayClass(kataHello: string): HelloItems[] {
69
70      this.stringDataHello = kataHello;
71      const keySampel = 'hello world key';
72
73      const arrayString: HelloItems[] = [];
74      const helloItem = new HelloItems(this.stringDataHello, keySampel);
75      arrayString.push(helloItem);
76
77      return arrayString;
78  }
79
80  }
81
82  export { HelloWorldDocs as HelloWorldDocument };

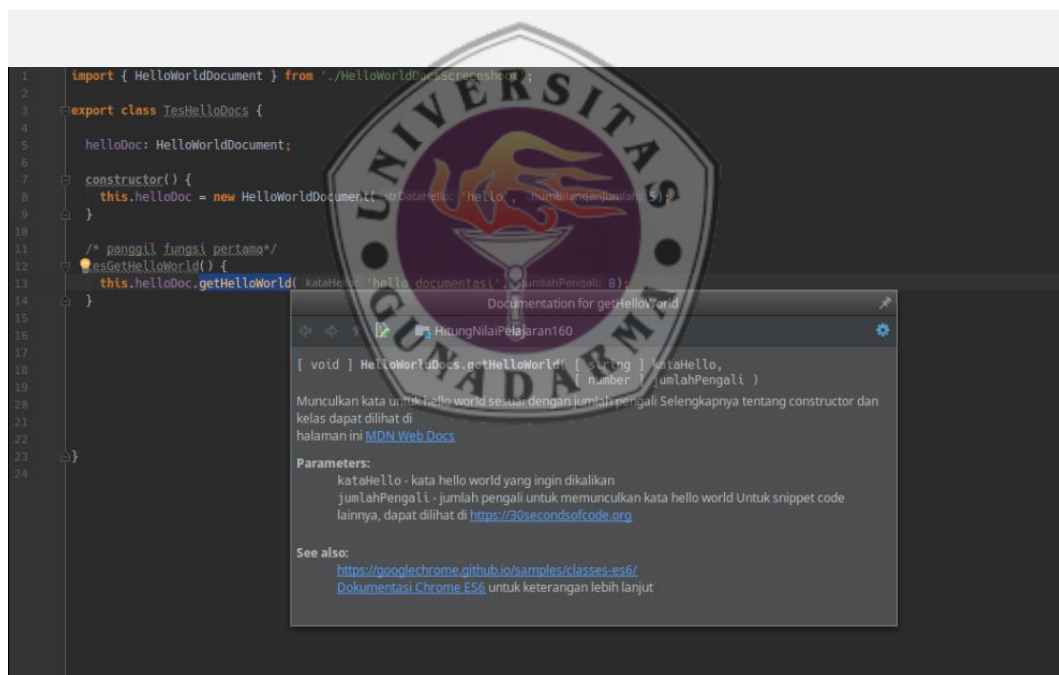
```

Setelah membuat kelas HelloWorldDocsFull.ts di atas, akan mencoba apakah dokumentasi yang dibuat pada kelas tersebut akan muncul ketika fungsi kelas tersebut dipanggil dan pintasan Quick Documentation **Ctrl + Q** (Linux/Windows) atau **F1** (Mac) dijalankan.

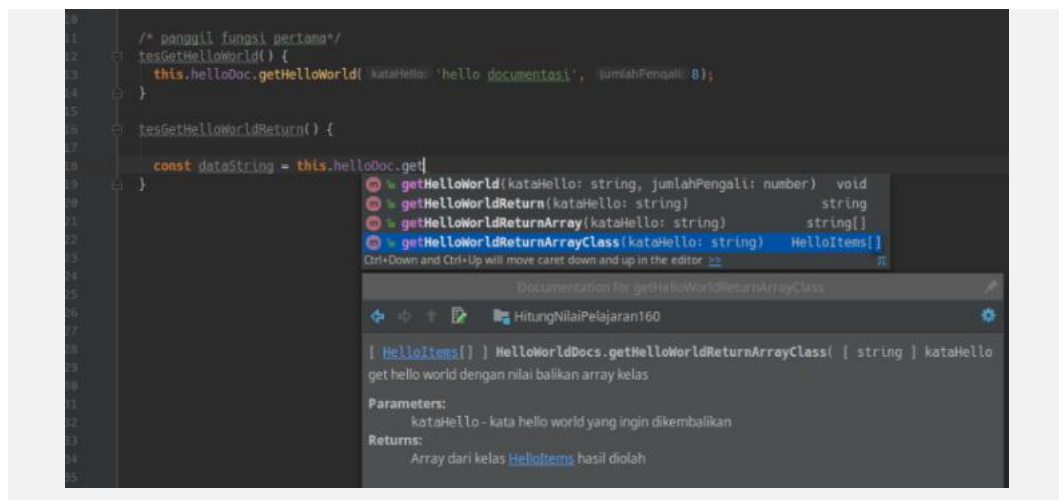
Pertama misalkan kita buat suatu kelas TypeScript baru, lalu impor HelloWorldDocsFull.ts dan inisialisasi kelas tersebut. Lalu panggil fungsi dari kelas HelloWorldDocsFull.ts yang telah diinisialisasi tadi. Ketika suggestion atau snippet fungsi muncul, aktifkan pintasan Quick Documentation. Hasilnya adalah nama fungsi beserta dokumentasi yang telah kita buat akan muncul. Seperti tampilan gambar-gambar 21, 22, 23 dan 24 di bawah ini.



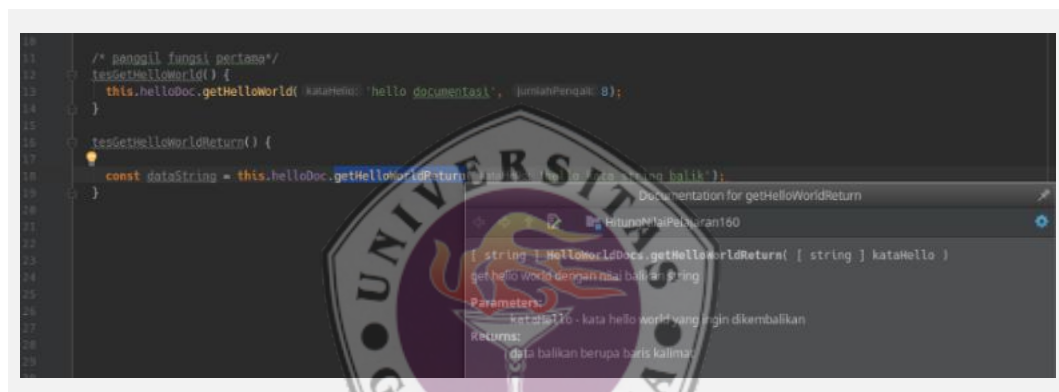
Gambar 21. Fungsi Kelas HelloWorld dengan Dokumentasi, muncul setelah Pintasan Quick Documentation Diaktifkan



Gambar 22. Dokumentasi Fungsi getHelloWorld Tampil



Gambar 23. Dokumentasi untuk Fungsi dengan Nilai Balikan



Gambar 24. Dokumentasi untuk Fungsi dengan Nilai Balikan

Berikut adalah potongan kode kelas TypeScript yang dipakai untuk pengujian tampilan dokumentasi di atas.

```

1  import { HelloWorldDocument } from './HelloWorldDocsFull';
2
3  export class TesHelloDocs {
4
5      helloDoc: HelloWorldDocument;
6
7      constructor() {
8          this.helloDoc = new HelloWorldDocument('hello', 5);
9      }
10
11     /* panggil fungsi pertama */
12     tesGetHelloWorld() {
13         this.helloDoc.getHelloWorld('hello dokumentasi', 8);
14     }
15
16     tesGetHelloWorldReturn() {
17
18         const dataString = this.helloDoc.getHelloWorldReturn('hello kata string balik');
19         const dataStringArray = this.helloDoc.getHelloWorldReturnArray('hello array');
20         const dataStringClass = this.helloDoc.getHelloWorldReturnArrayClass('hello class');
21     }
22
23 }

```

TesHelloDocs.ts hosted with ❤ by GitHub

[view raw](#)

Dengan adanya blok komentar dokumentasi tersebut, akan sangat membantu bagi seorang pengembang ataupun anggota tim pengembang lainnya dalam memahami kode yang telah ditulis. Tentu saja juga membantu jika kita lupa akan kode atau alur kerja dari aplikasi yang telah dibuat. Tentu saja dokumentasi yang baik ini akan membantu proses perawatan kode yang dibuat.

Contoh pembuatan blok dokumentasi di atas, dapat diterapkan pada JetBrains IDE yang lainnya seperti Android Studio, IntelliJ IDEA, PHPStorm, dan seterusnya. Blok dokumentasi tersebut dapat juga diterapkan pada kode-kode yang dibuat pada text editor, seperti Visual Studio Code dan Atom.


Menulis dokumentasi yang baik sangat penting bagi keberhasilan setiap proyek perangkat lunak. Kualitas dokumentasi bahkan bisa lebih penting daripada kualitas kode itu sendiri. Kebanyakan programmer

paling malas untuk membuat dokumentasi dengan alasan tidak mempunyai waktu yang cukup untuk membuat dokumentasi yang baik.

phpDocumentor dirancang untuk memudahkan untuk membuat dokumentasi. Untuk menginstall phpDocumentor, pertama harus mempunyai PEAR, cara installnya silakan dibaca sendiri. Setelah itu run command:

1. Pear install PhpDocumentor


dokumentasinya. Buat folder dan kita buat file-file php, contoh punya 2 file di satu folder, misal di D:\Zend\Apache2\htdocs\tutwordpress\testwillbedoc , dan beri nama WordPress1.php dan WordPress2.php. phpDocumentor mempunyai tag-tag yang berawalan dengan '@' yang merupakan deskripsi program.



```
1  <?php
2  /**
3   * This class is Parent of WordPress2
4   * @author Abdul Malik Ikhsan
5   * @filesource WordPress1.php
6   * @version 1.0
7   */
8  class WordPress1
9  {
10   /**
11    * a static variable
12    * @static
13    */
14    public static $astaticvar = 0;
15
16   /**
17    * @todo initialize all func needed
18    * @access public
19    * @param String $param1
20    * @param String $param2
21    */
22    public function WordPress1($param1,$param2)
23    {
24        //initialize all func needed
25    }
26 }
27 ?>
```

Kemudian file ke 2

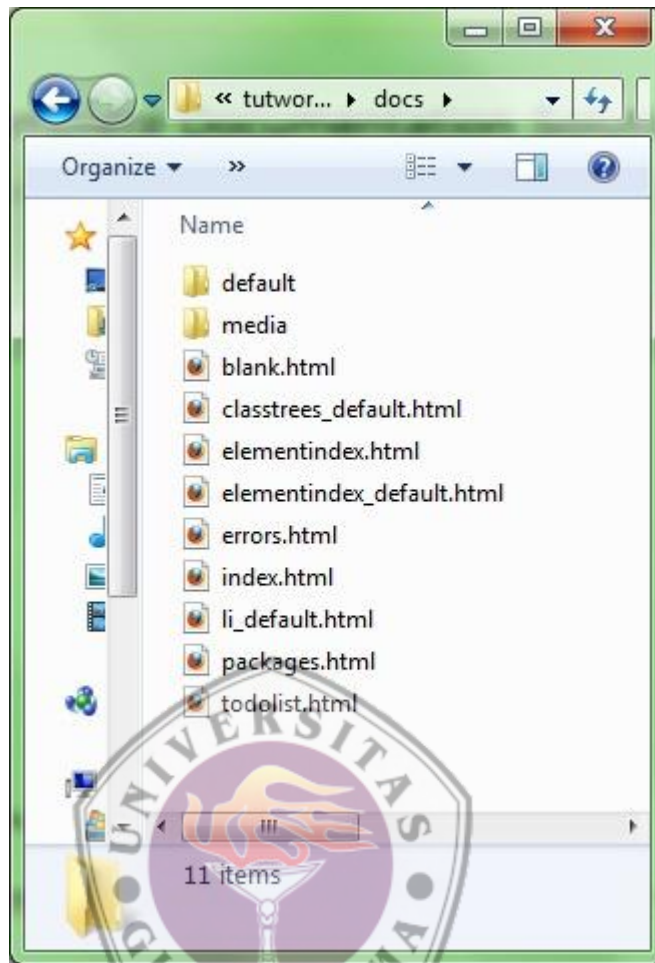
```
1  <?php
2  /**
3   * This class is Child of WordPress1
4   * @author Abdul Malik Ikhsan
5   * @filesource WordPress2.php
6   * @version 1.0
7   */
8
9  require_once 'Wordpress1.php';
10 class WordPress2 extends WordPress1
11 {
12     /**
13      * @todo initialize all func needed
14      * @access public
15      * @param String $param1
16      * @param String $param2
17      * @see WordPress1 Constructor
18      */
19     public function WordPress2($param1,$param2)
20     {
21         //initialize all func needed
22         parent::Wordpress1($param1,$param2);
23     }
24
25     private function staticFunc()
26     {
27
28     }
29
30     public function callerFunc()
31     {
32
33     }
34 }
35 ?>
```



Jika sudah, coba di run di command line, seperti berikut:

```
1  phpdoc -o HTML:frames:phpedit -d D:\Zend\Apache2\htdocs\tutwordpress\testwillbedoc -s -t
   D:\Zend\Apache2\htdocs\tutwordpress\docs
```

Jika berhasil, maka akan terbentuk folder di D:\Zend\Apache2\htdocs\tutwordpress\ dan di dalamnya ada file-file HTML yang tergenerate seperti gambar 25 dan 26 berikut:



Gambar 25 Tampilan folder di D:\Zend\Apache2\htdocs\tutwordpress\

Klik file index.html-nya, maka akan tampil seperti berikut:



Gambar 26 Tampilan Isi dari File index.html



JUNIOR WEB
PROGRAMMER

J.620100.025.02

Melakukan Debugging

8

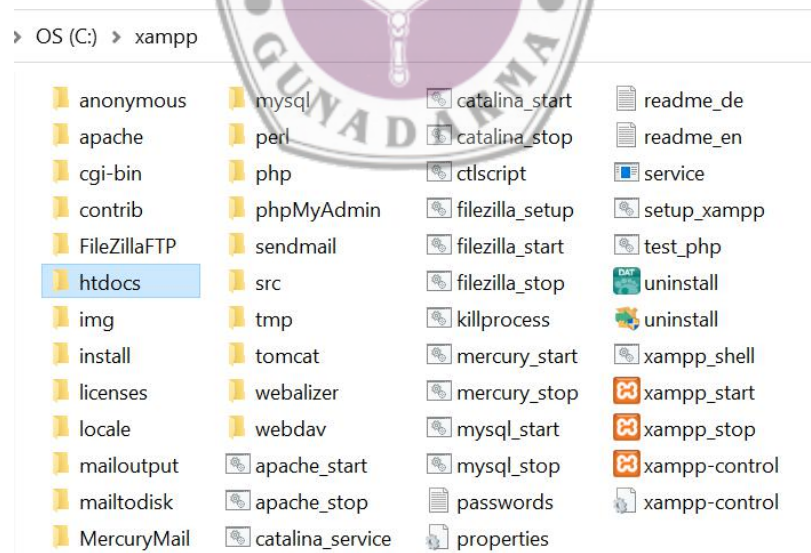
MELAKUKAN DEBUGGING

Objektif:

1. Mempersiapkan Kode Program
 2. Melakukan Debugging
 3. Memperbaiki Program
-

1. Mempersiapkan Kode Program

Agar dapat mengakses halaman PHP dari web browser, pada file PHP yang dibuat melalui aplikasi XAMPP maka tempatkan file PHP ke dalam folder khusus yang merupakan folder home dari web server. Pada aplikasi XAMPP, folder tersebut adalah folder htdocs yang berada di C:\xampp\htdocs. Seluruh file PHP harus diletakkan di dalam folder htdocs ini, seperti tampak pada gambar 1.



Gambar 1. Isi dari Folder XAMPP

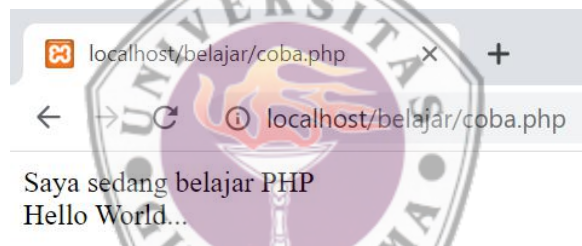
Agar memudahkan pencarian file, maka dapat dibuat folder “belajar” di dalam folder htdocs. Nantinya file yang dibuat akan disimpan

di dalam folder belajar ini. Untuk pembuatan script PHP berikut bisa menggunakan teks editor:

```
1 <?php
2     echo "Saya sedang belajar PHP";
3     echo "<br>";
4     echo "Hello World...";
5 ?>
```

Lalu save sebagai coba.php. File tersebut akan berada pada alamat C:\xampp\htdocs\belajar\coba.php. Untuk menjalankannya, harus menggunakan web browser.

Untuk menjalankan file PHP diatas, buka browser, dan ketik alamat: localhost/belajar/coba.php pada address bar dan tekan Enter, dan akan tampil seperti tampilan di gambar 2 berikut:



Gambar 2. Tampilan Ouput Script di Browser

Output dari script di atas adalah menampilkan beberapa baris text. Beberapa hal yang perlu diperhatikan terkait cara menjalankan file PHP:

- File php yang akan dijalankan harus berada di dalam folder C:\xampp\htdocs, baik itu di dalam folder tersebut, atau folder-folder dibawahnya. Untuk menjalankan di dalam browser, tinggal mengganti alamat C:\xampp\htdocs menjadi localhost. Jika file php berada di luar folder htdocs, maka web server tidak bisa mengaksesnya.
- Setiap file PHP harus ditulis menggunakan akhiran .php.

- Penamaan file PHP harus tanpa spasi dan sebaiknya menggunakan huruf kecil, dapat menggunakan underscore (_) sebagai pengganti spasi.

Beberapa text editor terbaik untuk Web Developer

Tool text editor terbaik haruslah mendukung dua hal ini: Syntax highlighting (warna-warni pada kode) dan auto completion. Pemilihan text editor yang tepat dan sesuai akan sangat membantu menemukan kesalahan dengan cepat, karena pada umumnya editor memiliki pewarnaan yang berbeda antara string, nama variabel maupun perintah bawaan PHP. Lima text editor yang bisa dipilih untuk mendukung mengembangkan sebuah website.

1. Sublime Text 3

Sublime Text merupakan tool text editor yang canggih untuk kode, markup dan prosa. Tool ini memiliki tampilan antarmuka yang apik, fitur yang luar biasa, dan kinerja yang sangat baik. Proyek di dalam Sublime Text menampilkan full content dari workspace, termasuk file yang dimodifikasi dan yang belum disimpan. Kamu dapat berpindah antar proyek dengan cara yang mirip dengan Goto Anything, dan berpindah secara instan, tanpa ada peringatan untuk menyimpannya. Semua modifikasi akan dipulihkan saat proyek kembali kamu buka.

2. Atom Editor

Atom Editor dikembangkan oleh GitHub, lingkungan yang sangat mudah untuk disesuaikan dan mudah untuk dipasang, menjadikan Atom sebagai pilihan dari banyak developer. Atom merupakan text editor modern, yang kamu dapat melakukan apapun untuk menyesuaikannya, dan tetap produktif tanpa menyentuh file konfigurasi.

3. Notepad++

Jika bekerja dengan HTML atau CSS, Notepad++ dapat melakukan banyak hal dan tool ini kamu bisa download gratis. Tool ini menyediakan plugin untuk menambah fungsinya, yang tentunya sangat baik. Seperti misalnya plugin AutoSave untuk menyimpan teks secara otomatis, ColdFusion, Comparison, dan plugin untuk mengatur toolbar.

4. Vim

Sebagai salah satu tool text editor terbaik, Vim dapat melakukan hampir semua yang dibutuhkan untuk memulai pemrograman di C dan 80% dari semua yang dibutuhkan untuk lebih dari 40 bahasa pemrograman dan tipe file disediakan disini. Vim memang terlihat sulit untuk dipelajari, namun untuk orang yang telah memiliki pengetahuan pemrograman tidak menjadi masalah.

5. Brackets

Brackets merupakan tool text editor yang ringan, powerful dan modern. Tool ini memadukan visual ke dalam editor, sehingga kita bisa mendapatkan bantuan yang diinginkan tanpa menghalangi proses kreatif. Brackets merupakan proyek open source, yang didukung oleh komunitas yang aktif. Tool ini didesain untuk mendukung web designer dan front-end developer.

Aturan Dasar Penulisan Kode PHP

Cara penulisan dan aturan dasar penulisan script PHP:

1. Case Sensitivity (perbedaan huruf besar dan kecil) dalam PHP

PHP tidak membedakan huruf besar dan kecil (case insensitive) untuk penamaan fungsi (*function*), nama class, maupun *keyword* bawaan PHP seperti *echo*, *while*, dan *class*. Ketiga baris berikut akan dianggap sama dalam PHP:

```

1  <?php
2  Echo "Hello World";
3  ECHO "Hello World";
4  EcHo "Hello World";
5  ?>

```

Akan tetapi, PHP membedakan huruf besar dan huruf kecil (case sensitive) untuk penamaan variabel, sehingga \$nama, \$Nama dan \$NAMA akan dianggap sebagai 3 variabel yang berbeda. Sering kali *error* terjadi karena salah menuliskan nama variabel, yang seharusnya menggunakan huruf kecil ditulis dengan huruf besar.

```

1  <?php
2  $andi="Andi";
3  echo $Andi; // Notice: Undefined variable: Andi
4  ?>

```

Untuk mengatasi perbedaan ini, disarankan menggunakan huruf kecil untuk seluruh script PHP, termasuk variabel, fungsi maupun class. Jika perlu membuat nama variabel yang terdiri dari 2 kata, karakter spasi bisa digantikan dengan underscore (_).

2. Penulisan Baris Perintah dalam PHP

Statement (baris perintah) di dalam PHP adalah kumpulan perintah PHP yang menginstruksikan PHP untuk melakukan sesuatu. Baris perintah ini bisa terdiri dari satu baris singkat (seperti perintah echo untuk menampilkan text di layar) atau bisa sesuatu yang lebih rumit dan terdiri dari beberapa baris, seperti kondisi if, atau kode perulangan (loop). Berikut adalah contoh beberapa baris perintah dalam PHP:

```

1  <?php
2  echo "Hello, world";
3  sebuah_fungsi(21, "belajar");
4  $a = 1;
5  $nama = " belajar";
6  $b = $a / 25.0;
7  if ($y == $z) {
8      echo "Tampilkan Tabel";
9  }
10 ?>

```

Terlihat dari beberapa contoh baris perintah di atas, PHP menggunakan tanda semicolon (titik koma) ";" sebagai tanda akhir baris perintah.

Kumpulan baris perintah yang menggunakan tanda kurung kurawal seperti kondisi IF atau perulangan (loop) tidak butuh tanda titik koma setelah kurung penutup.

```
1 <?php
2     if (true) {
3         echo "Perintah dijalankan"; // tanda titik koma harus ditulis
4     } // tidak diperlukan tanda titik koma setelah tanda kurung kurawal
5 ?>
```

3. Karakter Spasi dan Tab dalam PHP

Secara umum, karakter spasi dan tab diabaikan ketika mengeksekusi kode program PHP. Anda boleh memecah sebuah *statement* menjadi beberapa baris atau menyatukan beberapa *statement* dalam sebuah baris yang panjang. Seperti contoh pertama berikut:

```
1 <?php
2     echo "Ini kalimat pertama"; echo "Ini kalimat kedua"; $nama=" belajar";
3 ?>
```

Baris perintah contoh kedua ini sama artinya dengan contoh pertama:

```
1 <?php
2     echo "Ini kalimat pertama";
3     echo "Ini kalimat kedua";
4     $nama = " belajar";
5 ?>
```

Walaupun contoh pertama lebih menghemat tempat, namun lebih disarankan contoh kedua, dimana kita mengusahakan agar 1 *statement* berada pada 1 baris saja. Kemudian menambahkan beberapa spasi atau tab di awal untuk memudahkan membaca kode program (indenting).

Keuntungannya untuk beberapa waktu ke depan akan lebih mudah memahami kode program yang dibuat. Menambah baris komentar pada bagian kode yang lebih rumit sebagai penjelasan juga sangat disarankan.

4. Tag Pembuka dan Penutup PHP

Sebuah script PHP ditulis dalam tag pembuka `<?php` dan tag penutup `?>`. Khusus untuk file PHP yang seluruhnya terdiri dari perintah PHP (tidak ada kode HTML di dalamnya), maka tanda kurung penutup `?>` boleh tidak ditulis. Seperti contoh berikut:

```
1 <?php
2     echo "Sedang belajar PHP di Duniaikom";
3     echo "File ini hanya terdiri dari kode PHP saja";
```

Teknik ini sering dipakai untuk menghindari masalah jika kita sudah sering men-include satu file PHP ke file PHP lain. Akan tetapi jika di dalam file tersebut terdapat kode HTML setelah kode PHP, maka tetap harus ditutup dengan tanda `?>`. Pada baris 4 tanda `?>` harus ditulis karena di baris 5 sudah ada kode HTML :

```
1 <?php
2     echo "Sedang belajar PHP di Duniaikom";
3     echo "File ini terdiri dari kode PHP dan HTML";
4     ?>
5     <p>Ini adalah kode HTML</p>
```

Mengenal Struktur dan Perintah Dasar PHP

Dalam penulisanya, script PHP tidak harus berdiri sendiri, tetapi dapat disisipkan di antara kode HTML. Script PHP harus selalu diawali dengan `<?` atau `<?php` dan diakhiri dengan `?>`. Semua teks yang diketik setelah tanda buka script (`<?`) dan tanda tutup script (`?>`) akan diproses sebagai suatu script PHP.

Contoh :

```
<html>
<head>
    <title> echo.php </title>
</head>
<body>
    <?php echo ("cuman coba") ;
    ?>
</body>
</html>
```

Membuat script PHP tidaklah mudah, terkadang seorang programmer banyak melakukan kesalahan dalam kode yang dituliskannya. Kode yang dibuat biasanya tidak sedikit, bisa terdiri dari puluhan bahkan ratusan file dan setiap file bisa terdiri dari ratusan bahkan ribuan baris. Hal tersebut tentu sulit untuk menemukan kesalahan.

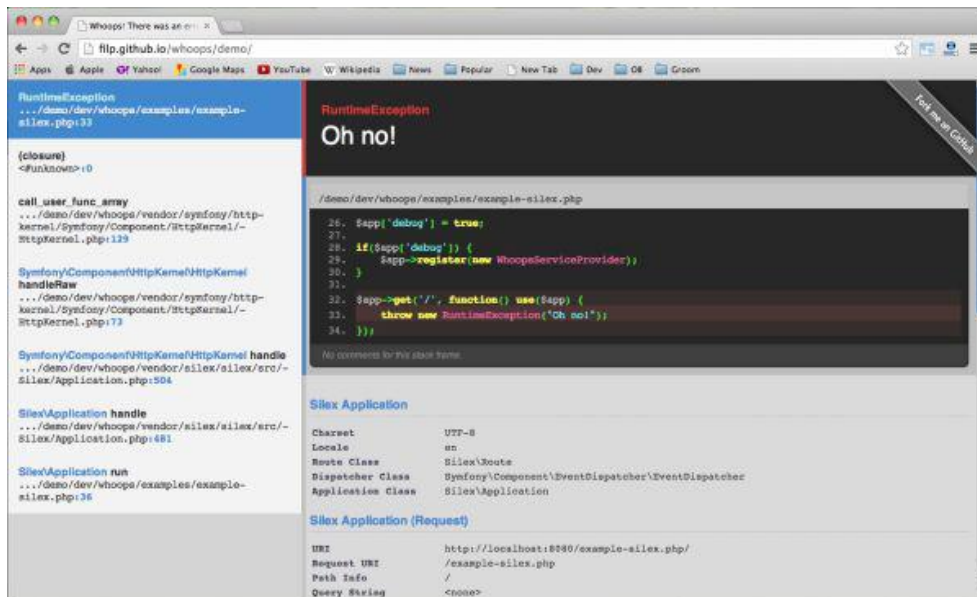
Dibutuhkan sebuah alat *debugging* untuk membantu mencari kesalahan dengan mudah dan mengurangi jumlah *bug*. Berikut ini beberapa alat untuk debug script PHP.

1. Pinba

Pinba adalah mesin penyimpanan MySQL *open source* dan tujuan utamanya adalah untuk membantu pengembangan web guna memantau semua kinerja script PHP. Alat ini bertindak sebagai *real time monitoring server* untuk PHP dan MySQL.

2. Whoops

Whoops merupakan sebuah PHP library untuk menangani *exceptions*, Whoops terbukti menjadi debugger yang sangat kompeten. Kesalahan ditangani di tumpukan menggunakan antarmuka yang mudah untuk dipahami dan enak untuk dilihat. Dilengkapi juga dengan API. Whoops mudah dikonfigurasi dan memiliki fitur yang *user-friendly*.



Gambar 3. Tampilan Whoops

3. Xdebug

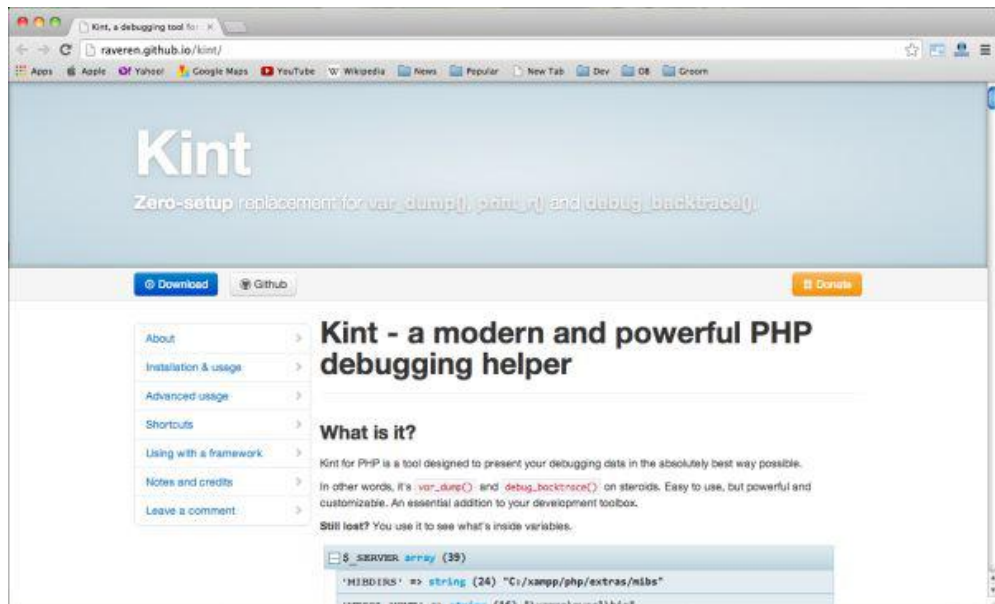
Xdebug pada dasarnya merupakan perluasan yang disediakan oleh *platform* PHP untuk orang-orang yang ingin terhindar dari kesalahan dan mencari cara yang sangat efektif untuk debug mereka. Alat ini juga merupakan alat yang dibuat untuk *profiling* skrip PHP. Alat ini hanya sebatas extension/perluasan saja, mengakses dan menggunakannya tidaklah sulit.

4. Krumo

Krumo adalah contoh yang bagus dari alat *debugging*, yang menampilkan informasi tentang struktur variabel PHP.

5. Kint

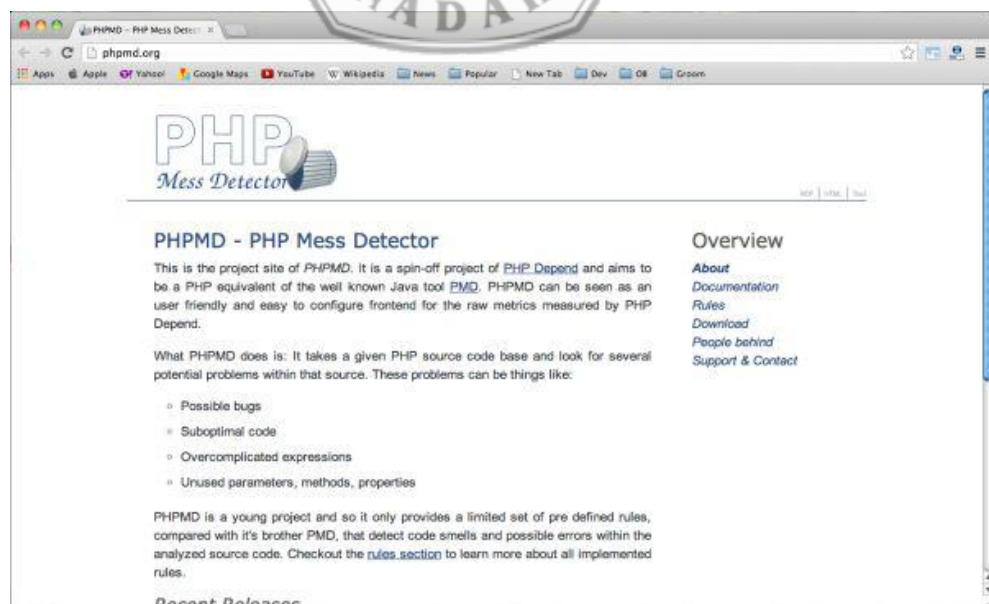
Kint adalah sebuah *opensource debugger*, Kint adalah alternatif yang bagus untuk debugger yang lebih populer seperti `var_dump`, `webgrind`, dan lainnya. Alat ini tidak memerlukan banyak pengaturan namun sangat efisien. Kint adalah alat yang intuitif untuk mendeteksi tipe data yang di dump.



Gambar 4. Tampilan Kint

6. PHPMD

PHPMD adalah alat yang bagus untuk mendapatkan kode sumber dan menemukan kesalahan-kesalahan yang ada di kode tersebut. Setiap jenis kesalahan dapat diselesaikan dengan mudah dengan pembatasan pendefinisian aturan.



Gambar 5. Tampilan PHP Mess Detector

7. PHP Debug Bar

PHP Debug Bar adalah salah satu alat lainnya yang sangat baik untuk proyek-proyek web dan dapat menampilkan data dari semua jenis aplikasi web.

2. Melakukan *Debugging*

Cara Menampilkan Pesan Error untuk Debugging Program PHP

Secara default, konfigurasi yang akan diberikan pada server Apache di Linux adalah mode *production*. Server produksi (*production server*) merupakan server yang digunakan setelah web selesai dikembangkan atau sudah siap dipakai. Server produksi tidak akan menampilkan apa-apa jika terjadi error. Karena itu, harus mengubahnya ke dalam mode *development*. Berikut adalah cara untuk mengubah ke mode *development*:

1. Buka File Konfigurasi Server

Buka file konfigurasi server yang ada di `/etc/php5/apache2/php.ini` dengan teks editor. Bukanya sebagai root. Gunakan perintah `gedit` untuk membukanya, maka perintahnya:

```
sudo gedit /etc/php5/apache2/php.ini
```

Untuk PHP7:

```
sudo gedit /etc/php/7.0/apache2/php.ini
```

2. Ubah Nilai Konfigurasi ke Mode Development

Ubahlah nilai-nilai konfigurasi error-nya menjadi seperti berikut ini.

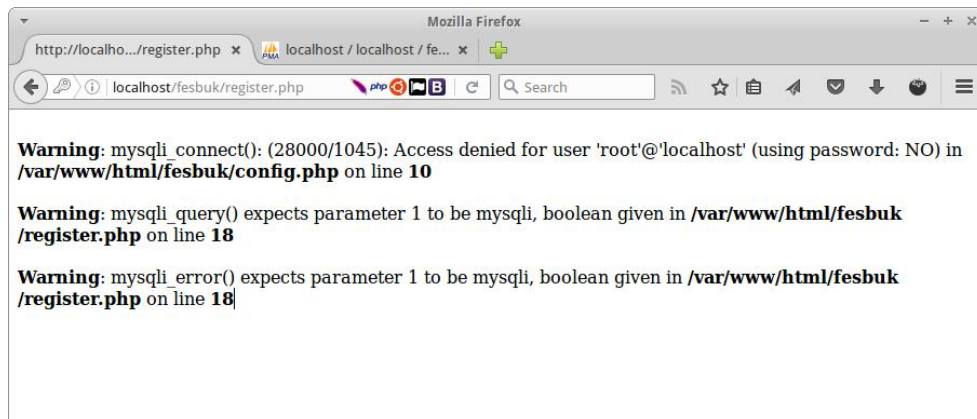
```
error_reporting = E_ALL
display_errors = On
display_startup_errors = On
track_errors = On
```

3. Hidupkan Ulang Service

Nyalakan kembali service apache2 dengan perintah

```
sudo service apache2 restart
```

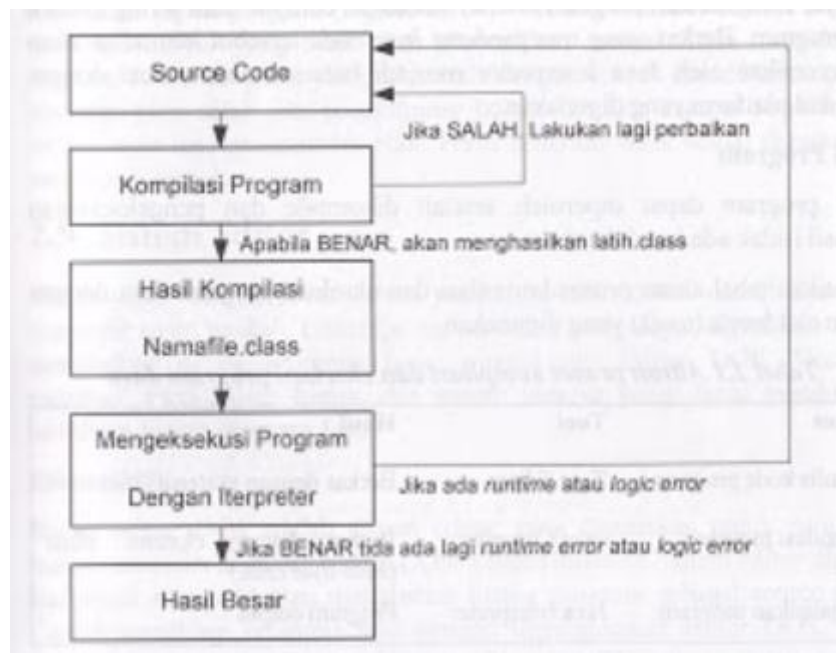
Sekarang buka sebuah file PHP yang memiliki error melalui localhost.



Gambar 6. Tampilan Error File PHP

Langkah-Langkah Kompilasi Dan Eksekusi Program Java

Bahasa pemrograman java menggunakan system interpreter berupa java interpreter. Java tidak menghasilkan file dengan extention exe. Java di fokuskan untuk aplikasi berbasis terdistribusi yang dapat diaktifkan melalui jaringan atau internet. Hasil dari kompilasi java menghasilkan file dengan extention class. File dengan extention class tersebut dapat dijalankan file dengan browser atau melalui applet viewer. Bahasa yang menggunakan penerjemah compiler akan menghasilkan file dengan extention exe dan file tersebut tidak dapat dipanggil oleh suatu browser. Langkah-langkah kompilasi dan eksekusi program java tampak pada gambar 7.



Gambar 7. Langkah Kompilasi dan Eksekusi Program

Source Code

Source code adalah file sumber yang dituliskan dengan extension java yang akan di kompilasi. Dalam hal ini, anda menuliskan sebuah program java dengan menggunakan sebuah editor sistem (editor, Notepad, kawapro, JCreator, atau sistem lain yang dapat mendukung pemrograman java). Contoh nama program latihan1.java.

Kompilasi Program

Kompilasi program adalah mengkompilasi dari file sumber menjadi file dengan extension class.

Hasil Kompilasi

Setelah membuat dan menyimpan kode program, kompilasi file berisi kode program tersebut dengan menggunakan java kompiler. hasil dari kompilasi berupa berkas byte code dengan ekstensi.class misalnya latihan1.class.

Mengeksekusi Program

Mengeksekusi program artinya, menjalankan program setelah dilakukan compile atau sering disebut run program. Berkas yang mengandung byte code tersebut kemudian akan dikonversikan oleh java interpreter menjadi bahasa mesin sesuai dengan jensi dan platform yang digunakan.

Hasil Program

Perhatikan tabel aliran kompilasi dan eksekusi program java dengan sistem alat bantu (tools) yang digunakan

3. Memperbaiki Program

Adanya "error" di dalam program seringkali "error" merupakan sesuatu yang menakutkan dan membingungkan. Menurut Achmatim (2008), terdapat tiga tingkatan error program yaitu error tata bahasa (sintaks), *error runtime* dan error logika. Dari ketiga jenis error program tersebut, sebenarnya yang paling mudah ditangani adalah error yang pertama yaitu error tata bahasa (sintaks) karena compiler atau interpreter langsung memberitahukannya saat program dikompilasi atau dijalankan. Namun demikian, tetap saja diperlukan pengalaman, ketelitian dan kegigihan dalam menangani atau mencari penyebab error yang muncul.

Berbagai macam error program PHP yang paling sering ditemui oleh programmer, dibagi menjadi 10 kategori berdasarkan penyebabnya masing-masing.

1. Kurang Titik-Koma

Jenis error program ini merupakan error yang paling sering ditemui. Penyebabnya karena kurang menambahkan titik-koma (;) pada akhir suatu perintah (*statement*). Solusi dalam mencari letak kesalahan programnya membutuhkan kejelian dan ketelitian. Namun demikian, umumnya akan ditunjukkan baris dimana error tersebut muncul.

Mulailah dari baris tersebut, namun tidak selamanya penyebab kesalahan berada pada baris yang ditunjukkan. Terkadang penyebab kesalahan justru berada di baris atas atau sebelumnya. Berikut ini beberapa contoh program beserta tampilan error yang akan muncul jika program dijalankan:

[Contoh Program 1](#)

```
1. <?php
2. $nim = "1111500111";
3. $nama = "Achmad Solichin"
4. echo $nim;
5. echo $nama;
6. ?>
```

Jika program di atas dijalankan maka akan ditampilkan error sebagai berikut:

```
Parse error: syntax error, unexpected T_ECHO in
/home/achmatim/public_html/artikel_demo/error-php/error1.php on line 4
```

Baris yang ditunjukkan pada tampilan error di atas adalah baris ke-4, namun jika diperhatikan program 1 di atas ternyata penyebab error justru bukan di baris ke-4. Penyebab error berada pada baris 3, dimana statement tidak diakhiri dengan titik-koma (;). Mengapa PHP justru menunjuk ke baris 4? Hal ini disebabkan karena proses pemeriksaan sintaks oleh PHP dilakukan secara berurutan (sekuensial) dari baris pertama hingga terakhir. Pada baris ketiga belum terjadi error, namun statement masih memerlukan adanya titik-koma (;).

Saat PHP berada di baris 4, bukan titik-koma (;) yang ditemukan, namun justru perintah "echo", sehingga disinilah PHP akan menunjukkan errornya. PHP tidak akan mengetahui penyebab errornya dimana, namun PHP akan menunjukkan pada baris mana error mulai terjadi. Tips: periksa program pada baris yang ditunjukkan oleh pesan error, jika tidak ditemukan, periksa baris atasnya, demikian dan seterusnya hingga penyebab error ditemukan.

[Contoh Program 2](#)

```
1. <?php
2. $user = "achmatim";
3. $pass = "123"
4. if ($user == "achmatim" && $pass == "123") {
5.   echo "Login Berhasil";
6. } else {
7.   echo "Login Gagal";
8. }
9. ?>
```

Jika program di atas dijalankan maka akan ditampilkan error sebagai berikut:

```
Parse      error:      syntax      error,      unexpected      T_IF      in
/home/achmatim/public_html/artikel_demo/error-php/error2.php on line 4
```

Sama seperti program 1, penyebab errornya adalah kurang titik-koma. Pada program 1, letak kesalahan diikuti dengan perintah echo sehingga pesan kesalahan mengatakan "unexpected T_ECHO". Sedangkan pada program 2, letak kesalahan diikuti dengan perintah if, sehingga pesan kesalahan yang ditampilkan adalah "unexpected T_IF". Dari pesan kesalahan tersebut, kita dapat mengetahui letak kesalahan pasti berada sebelum perintah if, tepatnya pada baris ke-3, statement tidak diakhiri dengan titik-koma (;).

[Contoh Program 3](#)

```
1. <?php
2. $day = date ("D")
3. switch ($day) {
4.   case 'Sun' : $hari = "Minggu"; break;
5.   case 'Mon' : $hari = "Senin"; break;
6.   case 'Tue' : $hari = "Selasa"; break;
7.   case 'Wed' : $hari = "Rabu"; break;
8.   case 'Thu' : $hari = "Kamis"; break;
9.   case 'Fri' : $hari = "Jum'at"; break;
10.      case 'Sat' : $hari = "Sabtu"; break;
11.      default : $hari = "Kiamat";
12. }
13. echo "Hari ini hari <b>$hari</b>";
14. ?>
```

Jika program di atas dijalankan maka akan ditampilkan error sebagai berikut:


```
Parse error: syntax error, unexpected T_SWITCH in
/home/achmatim/public_html/artikel_demo/error-php/error3.php on line 3
```

Program 3 letak kesalahan berada sebelum perintah switch, tepatnya pada baris ke-2, statement tidak diakhiri dengan titik-koma (;). Jadi, jika muncul pesan kesalahan yang mirip seperti di atas, langsung cari perintah switch dan perhatikan baris perintah sebelumnya.

[Contoh Program 4](#)

```
1. <?php
2. /* contoh 1 */
3. $i = 1
4. while ($i <= 10) {
5.     echo $i++; }
6. ?>
```

Jika program di atas dijalankan maka akan ditampilkan error sebagai berikut:

```
Parse error: syntax error, unexpected T_WHILE in
/home/achmatim/public_html/artikel_demo/error-php/error4.php on line 4
```

Pada contoh program 4 di atas, letak kesalahan berada tepat di atas perintah while, tepatnya di baris ke-3. Statement seharusnya diakhiri dengan titik-koma.

[Contoh Program 5](#)


```
1. <?php
2. function cetak_ganjil ($awal, $akhir) {
3.     for ($i=$awal; $i<$akhir; $i++) {
4.         if ($i%2 == 1) {
5.             echo "$i ";
6.         }
7.     }
8. }
9. //pemanggilan fungsi
10.$a = 10;
11.$b = 50;
12.echo "<b>Bilangan ganjil dari $a sampai $b : </b><br>"
13.cetak_ganjil($a, $b);
14.?>
```


Jika program di atas dijalankan maka akan ditampilkan error sebagai berikut:

```
Parse error: syntax error, unexpected T_STRING, expecting ';' or ':' in /home/achmatim/public_html/artikel_demo/error-php/error5.php on line 13
```

Pada contoh program 5 di atas, sudah jelas bahwa letak kesalahannya pada baris ke-12 dimana statement tidak diakhiri titik-koma. *Error* dengan pesan “unexpected T_STRING” perlu dicermati dengan lebih teliti karena *error* tidak hanya terjadi karena kurangnya titik-koma, namun dapat juga karena kurang koma (,) atau operator penggabungan string berupa titik (.). Namun demikian, umumnya terjadi karena penulisan string yang tidak lengkap.

Contoh Program 6



```
1. <?php
2. echo "test"
3. function cetak_ganjil ($awal, $akhir) {
4.   for ($i=$awal; $i<$akhir; $i++) {
5.     if ($i%2 == 1) {
6.       echo "$i ";
7.     }
8.   }
9. }
10. //panggilan fungsi
11. $a = 10;
12. $b = 50;
13. echo "<b>Bilangan ganjil dari $a sampai $b : </b><br>";
14. cetak_ganjil($a, $b);
15. ?>
```

Jika program di atas dijalankan maka akan ditampilkan error sebagai berikut:

```
Parse error: syntax error, unexpected T_FUNCTION, expecting ';' or ':' in /home/achmatim/public_html/artikel_demo/error-php/error6.php on line 3
```

Contoh program 6 di atas mirip dengan contoh program sebelumnya, hanya saja letak kesalahannya sekarang berada pada baris ke-2, sebelum pendefinisian fungsi. Pada baris tersebut, seharusnya diakhiri dengan titik-koma (;).

Contoh Program 7

```
1. <?php
2. $nim = "0411500400";
3. $nama = 'Chotimatul Musyarofah';
4. $umur = 23;
5. $nilai = 82.25;
6. $status = TRUE;
7. echo "NIM : " . $nim . "<br>";
8. echo "Nama : $nama<br>";
9. print "Umur : " . $umur; print "<br>";
10. printf ("Nilai : %.3f<br>", $nilai);
11.
12. if ($status)
13.     echo "Status : Aktif"
14. else
15.     echo "Status : Tidak Aktif";
16. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Parse error: syntax error, unexpected T_ELSE, expecting ';' or ';' in
/home/achmatim/public_html/artikel_demo/error-php/error7.php on line 14
```

Letak kesalahan pada contoh program 7 bukanlah pada baris ke-14 seperti ditunjukkan di atas, namun terletak di baris ke-13 (sebelum perintah ELSE) dimana statement tidak diakhiri dengan titik-koma.

2. Kekurangan atau Kelebihan Kutip

Penyebab error program yang kedua berkaitan dengan tanda kutip, baik kutip tunggal (') maupun kutip ganda ("). Seperti kita ketahui bahwa di PHP, string dapat dituliskan dengan diapit oleh tanda kutip tunggal (') dan kutip ganda ("). Error program akan terjadi jika penyajian tanda kutip yang tidak tepat, misalnya tidak ditutup (kurang kutip) atau kelebihan kutip. Berikut ini beberapa contoh program PHP yang error karena permasalahan tanda kutip:

Contoh Program 8

```
1. <?php
2. $nim = "0411500400;
3. $nama = 'Chotimatul Musyarofah';
4. echo "NIM : " . $nim . "<br>";
5. echo "Nama : $nama";
6. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Parse    error:    syntax    error,    unexpected    T_STRING    in
/home/achmatim/public_html/artikel_demo/error-php/error8.php on line 4
```

Pesan kesalahan pada program 8 di atas menunjuk pada baris ke-4, tetapi penyebab kesalahan justru ada di baris ke-2. Hal tersebut terjadi karena pada baris ke-2 terdapat string yang dibuka dengan kutip ganda (") namun tidak ada penutup atau pasangannya. Pada proses kompilasi, PHP akan memeriksa sintaks program secara berurutan. Saat ditemukan tanda kutip, maka program akan menganggap teks berikutnya sebagai sebuah string hingga ketemu pasangannya. Program 8 di atas tidak menemukan pasangan dari kutip ganda (") yang ada di baris kedua pada baris yang sama, namun justru ditemukan kutip ganda (") pada baris ke-4. Kutip ganda pada baris keempat dianggap sebagai pasangan (penutup) kutip ganda pada baris kedua. Setelah ditemukan penutup dari kutip tersebut, maka program akan menganggap bahwa penyajian string (statement) sudah lengkap dan harus ditutup dengan titik-koma atau operator penggabungan string. Ternyata program tidak menemukan penutup statement setelahnya, sehingga terjadilah error di baris ke-4. Untuk mengatasi error pada program 8 di atas tentunya cukup dengan menambahkan kutip ganda sebelum titik-koma pada baris kedua. Letak kesalahan seringkali berada pada baris yang sama sekali berbeda dengan petunjuk pada pesan kesalahannya.

[Contoh Program 9](#)

```
1. <?php
2. $nim = "0411500400";
3. $nama = 'Chotimatul Musyarofah;
4. echo "NIM : " . $nim . "<br>";
5. echo "Nama : $nama";
6. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Parse error: syntax error, unexpected T_ENCAPSED_AND_WHITESPACE in
/home/achmatim/public_html/artikel_demo/error-php/error9.php on line 3
```

Penyebab error pada program 9 di atas sudah cukup jelas, yaitu kurangnya pasangan tanda kutip tunggal (`) pada baris ketiga. Sama seperti error pada program sebelumnya, kita juga dapat menganalisa kesalahan program dengan memperhatikan warna *source code* yang ditampilkan oleh editor.

[Contoh Program 10](#)

```
1. <?php
2. $nim = 0411500400";
3. $nama = 'Chotimatul Musyarofah';
4. echo "NIM : " . $nim . "<br>";
5. echo "Nama : $nama";
6. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Parse error: syntax error, unexpected "" in
/home/achmatim/public_html/artikel_demo/error-php/error10.php on line 2
```

Perhatikan baris kedua program 10 di atas. Terlihat bahwa nilai dari variabel \$nim bertipe string, sehingga harus diapit dengan tanda kutip. Inisialisasi variabel \$nim tidak diawali dengan tanda kutip ganda (``) sehingga menyebabkan error program.

3. Kurang Operator Penggabungan String

Seperti diketahui bahwa suatu string dapat digabungkan dengan variabel atau string lainnya menggunakan operator penggabungan titik (.). Error akan ditampilkan jika pada penggabungan string yang tidak menggunakan operator tersebut. Berikut ini contoh program PHP yang error dalam hal penggabungan string.

Contoh Program 11

```
1. <?php
2. $nim = "0411500400";
3. $nama = 'Chotimatul Musyarofah';
4. echo "NIM : " $nim . "<br>";
5. echo "Nama : $nama";
6. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

Parse error: syntax error, unexpected T_VARIABLE, expecting ',' or ';' in /home/achmatim/public_html/artikel_demo/error-php/error11.php on line 4

Perhatikan baris keempat program 11 di atas. Antara string "NIM : " dan variabel \$nim seharusnya terdapat operator penggabungan titik (.), seperti pada penggabungan \$nim dan string "
". Selain dengan menggunakan operator penggabungan, string dan variabel juga dapat digabungkan dengan memasukkan nama variabel ke dalam string yang diapit dengan kutip ganda, seperti pada contoh baris ke-5 program 11 di atas.

4. Kurang atau Kelebihan Kurung Kurawal

Pada bahasa pemrograman PHP, kurung kurawal, baik { maupun }, digunakan untuk mengelompokkan suatu blok perintah tertentu, seperti pada struktur kondisi IF-ELSE, perulangan WHILE, pendefinisian fungsi, dan sebagainya. Penanganan error program yang berkaitan dengan kelengkapan kurung kurawal terkadang sedikit sulit

dilakukan, terutama jika program sudah cukup kompleks dan gaya penulisan program (*coding style*) tidak baik dan teratur. Kerapihan dalam penulisan program sangat berguna bagi programmer dalam mengelola program di kemudian hari, maupun dalam melakukan penelusuran kesalahan (error).

Gaya penulisan program atau *coding style* memang setiap programmer memang berbeda, namun demikian biasakanlah untuk mengikuti *coding style* yang baik. Di antara *coding style* yang baik adalah konsistensi dalam penulisan nama variabel, fungsi dan class, penggunaan indentasi (*indenting*) untuk blok dengan level yang berbeda, penambahan keterangan atau komentar program dan sebagainya. Dapat mengikuti salah satu *coding standard* yang sudah ada dan banyak digunakan oleh programmer di seluruh dunia, seperti Zend Framework Coding Standard dan PEAR Coding Standard.

Beberapa contoh program berikut ini akan mempermudah pemahaman dalam menangani kesalahan program terkait kurung kurawal.

Contoh Program 12

```
1. <?php
2. $day = date ("D");
3. switch ($day) {
4.     case 'Sun' : $hari = "Minggu"; break;
5.     case 'Mon' : $hari = "Senin"; break;
6.     case 'Tue' : $hari = "Selasa"; break;
7.     case 'Wed' : $hari = "Rabu"; break;
8.     case 'Thu' : $hari = "Kamis"; break;
9.     case 'Fri' : $hari = "Jum'at"; break;
10.    case 'Sat' : $hari = "Sabtu"; break;
11.    default : $hari = "Kiamat";
12.
13. echo "Hari ini hari <b>$hari</b>";
14. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Parse      error:      syntax      error,      unexpected      $end      in  
/home/achmatim/public_html/artikel_demo/error-php/error12.php on line 15
```

Saat menemukan pesan kesalahan seperti di atas, terkadang bingung sendiri karena kesalahan justru menunjuk pada baris yang tidak ada perintah apapun yaitu baris ke-15. Error tersebut merupakan salah satu indikasi bahwa penyebab error adalah kurangnya kurung kurawal penutup. Yang harus dilakukan adalah dengan memeriksa setiap blok dalam program dan pastikan bahwa setiap kurawal buka { memiliki pasangan }. Disinilah kerapihan penulisan program akan sangat membantu penelusuran. Pada contoh program 12 di atas letak kesalahannya ada di baris ke-12 dimana kurawal buka pada blok SWITCH-CASE belum memiliki pasangan (belum ditutup).

[Contoh Program 13](#)

```
1. <?php  
2. function cetak_ganjil ($awal, $akhir) {  
3.   for ($i=$awal; $i<$akhir; $i++) {  
4.     if ($i%2 == 1) {  
5.       echo "$i ";  
6.     }  
7.   }  
8. }  
9. }  
10. //pemanggilan fungsi  
11. $a = 10;  
12. $b = 50;  
13. echo "<b>Bilangan ganjil dari $a sampai $b : </b><br>";  
14. cetak_ganjil($a, $b);  
15. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Parse      error:      syntax      error,      unexpected      '}'      in  
/home/achmatim/public_html/artikel_demo/error-php/error13.php on line 9
```

Berbeda dengan contoh program 12, pada contoh program 13 justru kelebihan kurung kurawal penutup sehingga ditampilkan error program seperti di atas. Pada pesan kesalahan ditunjukkan bahwa

error berada pada baris ke-9, namun sebenarnya kelebihan kurawal berada di baris ke-8. Hapus kurawal } pada baris ke-8. Sekali lagi disini gaya penulisan program yang baik akan sangat membantu proses identifikasi kesalahan. Jika salah dalam menghapus kurung kurawal seringkali akan menimbulkan error yang lain, sehingga diperlukan ketelitian dan pengalaman dalam menangani error serupa.

5. Kesalahan Nama Variabel

Variabel merupakan tempat penyimpanan suatu nilai dalam program. Variabel terdiri dari 2 (dua) jenis, yaitu variabel bawaan (*built-in*) dari PHP dan variabel yang didefinisikan oleh *programmer*. Untuk jenis yang kedua, kita dapat mendefinisikan variabel sesuai kebutuhan, namun penamaannya harus mengikuti aturan-aturan yang telah ditetapkan PHP. Beberapa aturan dasar antara lain harus diawali dengan tanda \$ dan diikuti dengan huruf atau garis bawah (_), tidak boleh diawali dengan angka, tidak boleh mengandung karakter khusus seperti spasi, tanda '\$' dan tanda '+', dan seterusnya. Kesalahan dalam penamaan variabel dapat menyebabkan error program.

Berikut ini contoh error program yang disebabkan karena kesalahan penamaan variabel.

[Contoh Program 14](#)

```
1. <?php
2. $nim = "0411500400";
3. $nama = 'Chotimatul Musyarofah';
4. $umur = 23;
5. $3nilai = 82.25;
6. $status = TRUE;
7. echo "NIM : " . $nim . "<br>";
8. echo "Nama : $nama<br>";
9. print "Umur : " . $umur; print "<br>";
10. printf ("Nilai : %.3f<br>", $nilai);
11.
12. if ($status)
13.     echo "Status : Aktif";
14. else
15.     echo "Status : Tidak Aktif";
16. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Parse error: syntax error, unexpected T_LNUMBER, expecting T_VARIABLE or '$' in  
/home/achmatim/public_html/artikel_demo/error-php/error14.php on line 5
```

Pada baris ke-5 program 14 di atas terdapat pendefinisian variabel `$3nilai`. Nama variabel tersebut tidak diijinkan karena setelah tanda `$` terdapat karakter angka. Seharusnya setelah tanda `$` diikuti oleh huruf atau karakter garis-bawah (`_`). Untuk memperbaiki program di atas, tentu dengan mengganti nama variabel sesuai dengan ketentuan penamaan variabel yang benar.

6. Variabel Belum Didefinisikan

Dalam hal penanganan variabel, PHP memang memiliki keleluasaan seperti variabel tidak perlu dideklarasikan. Kita dapat langsung mendefinisikan isi dari variabel tanpa perlu dipusingkan dengan deklarasi atau tipe data dari variabel. Tipe data akan ditentukan oleh PHP berdasarkan isi dari variabelnya. Namun demikian, kita tetap perlu memahami bahwa pada suatu kondisi, penggunaan variabel yang belum terdefiniskan sebelumnya dapat menyebabkan munculnya error program. Sebagai contoh dalam program 15 berikut ini.

[Contoh Program 15](#)

```
1. <?php  
2. for($i=1; $i<=10; $i++) {  
3.   $total = $total + $i;  
4. }  
5. echo $total;  
6. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Notice: Undefined variable: total in  
/home/achmatim/public_html/artikel_demo/error-php/error15.php on line 3
```

Maksud dari program 15 di atas adalah ingin menghitung dan menampilkan total dari 10 bilangan bulat yang dimulai dari 1. Dengan demikian terdapat variabel `$total` yang akan diisikan hasil penjumlahan untuk setiap bilangan dari 1 hingga 10. Perhatikan baris ke-3 program dimana terdapat perintah `$total = $total + $i;`. Pesan kesalahan muncul karena pada saat perulangan baru dimulai (`$i = 1`) nilai awal dari `$total` yang ditambahkan dengan nilai `$i` belum terbentuk atau belum ada. Untuk mengatasi error tersebut, kita perlu mendefinisikan nilai awal dari `$total` sebelum perulangan untuk menghitung total. Karena `$total` pasti akan bernilai bilangan bulat, maka kita dapat memberi nilai awal `$total` dengan 0, sehingga perintah yang perlu ditambahkan adalah `$total = 0`.

7. Mengakses Index Array yang Tidak Ada

Seperti halnya variabel, dalam hal penanganan array, PHP juga tidak memerlukan pendeklarasian awal maupun penentuan jumlah elemen dari array. Akan tetapi, pengaksesan elemen array yang tidak ada atau belum terbentuk akan menyebabkan error seperti pada contoh program 16 berikut ini.

Contoh Program 16

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

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Notice: Undefined offset: 4 in /home/achmatim/public_html/artikel_demo/error-
php/error16.php on line 4
```

Pada program 16 di atas terdapat sebuah variabel array dengan nama `$arrBuah` yang didefinisikan berisi 4 elemen. Index dari elemen

array secara default diawali dengan 0, sehingga elemen array tersebut dapat diakses dengan index 0, 1, 2 dan 3. Pesan kesalahan terjadi karena pada baris ke-4, program mengakses elemen dengan index 4, dimana index array tersebut tidak ada atau belum terbentuk. Jika ingin mengakses elemen terakhir dari \$arrBuah tersebut, maka seharusnya menggunakan index 3.

8. Pembagian dengan Nol

Perhitungan matematika di dalam program seringkali diperlukan, misalnya dalam kaitannya dengan perhitungan gaji, perhitungan total pemesanan barang, dan sebagainya. Terkait dengan perhitungan matematis tersebut, kita perlu jeli dalam penerapannya, terutama jika berkaitan dengan pembagian. Dalam matematika, pembagian dengan bilangan 0 akan menghasilkan nilai yang tidak terdefinisi. Dalam program, pembagian dengan bilangan 0 akan menyebabkan adanya error "Division by zero". Berikut ini contoh sederhananya:

Contoh Program 17

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

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Warning: Division by zero in /home/achmatim/public_html/artikel_demo/error-
php/error17.php on line 5
```

Perhatikan program 17 di atas. Nilai variabel \$a adalah 10, \$b adalah 0. Pada baris ke-5 terdapat perintah untuk menampilkan hasil pembagian \$a / \$b, yang berarti 10 / 0. Tentu hal tersebut akan menyebabkan error seperti terlihat pada contoh di atas.

9. Memanggil Fungsi yang Belum Terdefinisi

Setiap fungsi, baik fungsi built-in maupun fungsi buatan, yang dipanggil dalam program PHP, harus sudah didefinisikan terlebih dahulu. Fungsi built-in sudah didefinisikan oleh PHP sehingga tidak perlu didefinisikan lagi, contohnya fungsi `strlen()`, `addslashes()`, `explode()`, `date()` dan sebagainya. Sementara itu, fungsi buatan harus didefinisikan sendiri oleh programmer dan letak pendefinisian fungsi dapat diakses oleh bagian program yang memanggilmnya. Pemanggilan fungsi yang belum terdefinisi atau kesalahan penulisan nama fungsi saat pemanggilan sering terjadi sehingga menyebabkan pesan kesalahan.

Pada contoh program berikut ini terdapat kesalahan penulisan fungsi `addslashes()` sehingga menyebabkan error "Call to undefined function".

[Contoh Program 18](#)

```
1. <?php
2. $str = "Is your name O'Reilly ?";
3. $str2 = addslashes ($str);
4. $str3 = stripslashes ($str2);
5. echo "<b>String asli</b> : $str";
6. echo "<br><b>addslashes()</b> : $str2";
7. echo "<br><b>stripslashes()</b> : $str3";
8. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Fatal error: Call to undefined function addslashes() in
/home/achmatim/public_html/artikel_demo/error-php/error18.php on line 3
```

10. Menyertakan File (Include) yang Tidak Ada

Pada aplikasi berbasis PHP, menyertakan (include) file sudah biasa dilakukan. Misalnya pada saat program harus menyertakan library. Untuk menyertakan file lain dalam PHP dapat menggunakan fungsi `include()`, `include_once()`, `require()` atau `require_once()`. Masing-masing memiliki perbedaan penggunaannya. Penyertaan file yang tidak benar, baik karena

filenya tidak ada, path atau lokasi file yang salah maupun akses terhadap file tidak diijinkan, dapat menyebabkan error program. Contohnya pada program 19 berikut ini.

Contoh Program 19

```
1. <?php
2. include "koneksi.php";
3. echo "testing";
4. ?>
```

Jika program di atas dijalankan maka akan ditampilkan pesan kesalahan sebagai berikut:

```
Warning: include(koneksi.php): failed to open stream: No such file or directory in
/home/achmatim/public_html/artikel_demo/error-php/error19.php on line 2 Warning:
include(): Failed opening 'koneksi.php' for inclusion
(include_path='.:usr/share/php:usr/share/pear') in
/home/achmatim/public_html/artikel_demo/error-php/error19.php on line 2
```

File koneksi.php yang disertakan dalam program 19 di atas tidak ada sehingga pesan kesalahan “failed to open stream” muncul. Dengan demikian, kita harus pastikan bahwa file yang kita include-kan benar-benar ada dan diakses dengan benar.

Jika dilihat dari 10 kategori berdasarkan penyebabnya masing-masing, paling tidak ada 4 jenis tipe galat (error) pada PHP, di antaranya adalah sebagai berikut:

1. Parse Errors (syntax errors)

Parse Error ini terjadi jika ada kesalahan sintaks dalam script dan pesan kesalahan akan muncul pada *outputnya* ketika dijalankan. *Parse error* akan menghentikan proses eksekusi script. Ada banyak alasan ketika terjadinya *parse error* di PHP. Alasan umum pada *parse error* adalah sebagai berikut:

- Kutipan yang tidak ditutup
- Kelebihan atau kekurangan tanda kurung
- Kurung kurawal yang tidak ditutup
- Kurang titik koma

Contoh:

```
<?php
echo "Cat";
echo "Dog"
echo "Lion";
?>
```

Output:

Pada kode di atas tidak menuliskan titik koma di baris kedua. Ketika itu akan ada terjadinya parse error atau syntax error yang menghentikan eksekusi script, seperti pada gambar berikut:



2. Fatal Errors

Fatal error terjadi ketika PHP mengerti kode yang telah ditulis, namun apa yang diminta oleh kode tidak dapat dilakukan. *Fatal error* akan menghentikan eksekusi script. Jika mencoba untuk mengakses fungsi yang belum didefinisikan, maka outputnya adalah *fatal error*. Contoh:

```
<?php
function fun1() {
    echo "CodePolitan";
}
fun2();
echo "Fatal Error !!";
```


Output:

Pada kode di atas kita mendefinisikan fungsi `fun1()` tapi kita memanggil fungsi lain yaitu `fun2()` yang mana belum terdefinisi. Seperti pada gambar berikut:



3. Warning Errors

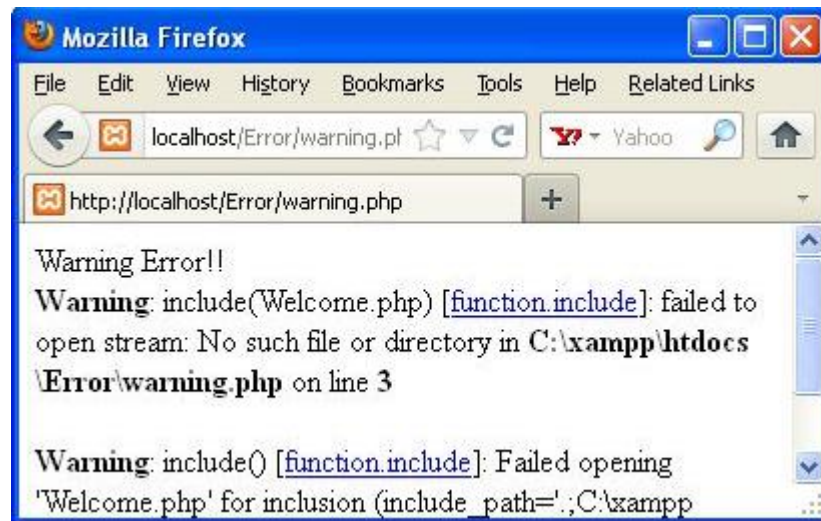
Warning error tidak akan menghentikan eksekusi dari script. Alasan utama dari *warning error* yaitu menyertakan file yang tidak ada atau mengisi jumlah parameter yang tidak pas saat memanggil suatu fungsi.

Contoh :

```
<?php
echo "Warning Error!!";
include ("welcome.php");
```

Output:

Pada kode di atas menyertakan file `welcome.php`, namun bila file tersebut tidak ada dalam direktori, maka akan muncul *warning error*. Tapi hal itu tidak akan menghentikan eksekusi script, akan terlihat pesan *Warning Error!!* diikuti pesan warning error, seperti pada gambar berikut:



4. Notice Errors

Notice error sama halnya dengan *Warning Error* yaitu ketika terjadi *notice error* eksekusi script tidak akan berhenti. *Notice Error* akan terjadi ketika mencoba untuk mengakses variabel yang belum didefinisikan.

Contoh:

```
<?php
$a="CodePolitan";
echo "Notice Error !!";
echo $b;
```

Output:

Pada kode di atas mendefinisikan variabel yang bernama `$a`, tapi yang dipanggil adalah variabel lain yaitu `$b`, yang tidak didefinisikan. Sehingga akan muncul *notice error* tapi eksekusi dari script tidak berhenti, akan muncul pesan *Notice Error !!*. Seperti pada gambar berikut:

