Muhammad Zen | Sayuti Rahman
Haida Dafitri | Risko Liza
Rachmat Aulia | Nurjamiyah

PEMROGRAMAN WEB

UNTUK PEMULA HINGGA MAHIR

PEMROGRAMAN WEB
UNTUK PEMULA HINGGA MAHIR

Muhammad Zen, S.T., M. Kom
Dr. Sayuti Rahman
Haida Dafitri, S.T., M.Kom
Risko Liza, S.T., M.Kom
Rachmat Aulia, S.Kom., M.Sc.IT
Nurjamiyah, S.Kom., M.Cs

Tahta Media Group

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Pelindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

i. penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
iii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
iv. penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN
CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202295338, 27 November 2022
Pencipta		
Nama	:	**Muhammad Zen, S.T., M. Kom, Dr. Sayuti Rahman dkk**
Alamat	:	Dusun 1 Bukit Gantung Langkat, Kota Langkat, SUMATERA UTARA, 20851
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Muhammad Zen, S.T., M. Kom, Dr. Sayuti Rahman dkk**
Alamat	:	Dusun 1 Bukit Gantung Langkat, Kota Langkat, SUMATERA UTARA, 20851
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Buku**
Judul Ciptaan	:	**PEMROGRAMAN WEB UNTUK PEMULA HINGGA MAHIR**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	27 November 2022, di Medan
Jangka waktu pelindungan	:	Berlaku selama hidup Pencipta dan terus berlangsung selama 70 (tujuh puluh) tahun setelah Pencipta meninggal dunia, terhitung mulai tanggal 1 Januari tahun berikutnya.
Nomor pencatatan	:	000411082

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n Menteri Hukum dan Hak Asasi Manusia
Direktur Jenderal Kekayaan Intelektual
u.b.
Direktur Hak Cipta dan Desain Industri

Anggoro Dasananto
NIP.196412061991031002

Disclaimer:
Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Muhammad Zen, S.T., M. Kom	Dusun 1 Bukit Gantung Langkat
2	Dr. Sayuti Rahman	Perumahan Pondok 6, Desa Kolam, Deliserdang
3	Haida Dafitri, S.T., M.Kom	Jl. Kebon Agung No. Mulio Rejo Sunggal Deli Serdang
4	Risko Liza, S.T., M.Kom	Jl. Bono 16. Medan Sumatra Utara
5	Rachmat Aulia, S.Kom., M.Sc.IT	Jl.Eka Surya, Komp.Royal Monaco Blok C No.2, Gedung Johor, Medan
6	Nurjamiyah, S.Kom., M.Cs	Jl. Huta V Manik Rejo, Silampuyang, Kec. Siantar Simalungun

LAMPIRAN PEMEGANG

No	Nama	Alamat
1	Muhammad Zen, S.T., M. Kom	Dusun 1 Bukit Gantung Langkat
2	Dr. Sayuti Rahman	Perumahan Pondok 6, Desa Kolam, Deliserdang
3	Haida Dafitri, S.T., M.Kom	Jl. Kebon Agung No. Mulio Rejo Sunggal Deli Serdang
4	Risko Liza, S.T., M.Kom	Jl. Bono 16. Medan Sumatra Utara
5	Rachmat Aulia, S.Kom., M.Sc.IT	Jl.Eka Surya, Komp.Royal Monaco Blok C No.2, Gedung Johor, Medan
6	Nurjamiyah, S.Kom., M.Cs	Jl. Huta V Manik Rejo, Silampuyang, Kec. Siantar Simalungun

PEMROGRAMAN WEB
UNTUK PEMULA HINGGA MAHIR

Penulis:
Muhammad Zen, S.T., M. Kom
Dr. Sayuti Rahman
Haida Dafitri, S.T., M.Kom
Risko Liza, S.T., M.Kom
Rachmat Aulia, S.Kom., M.Sc.IT
Nurjamiyah, S.Kom., M.Cs

Desain Cover:
Tahta Media

Editor:
Tahta Media

Proofreader:
Tahta Media

Ukuran:
xv, 156 , Uk: 15,5 x 23 cm

ISBN: 978-623-8070-19-0

Cetakan Pertama:
November 2022

PENERBIT TAHTA MEDIA GROUP
(Grup Penerbitan CV TAHTA MEDIA GROUP)
Anggota IKAPI (216/JTE/2021)

PRAKATA

Assalamualaikum wr. wb.

Segala puji bagi Allah SWT. yang telah memberikan rahmat-Nya yang tak terhingga, sehingga buku ini dapat kami selesaikan. Adapun buku ini berjudul **PEMROGRAMAN WEB UNTUK PEMULA HINGGA MAHIR**. Buku ini kami persembahkan untuk siapa saja yang ingin belajar pemrograman web dari dasar. Harapan kami untuk para pembaca adalah dapat mendalami pemrograman web dengan mudah melalui buku ini.

Pemrograman web merupakan salah satu pemrograman dalam bidang komputer yang berkembang pesat seiring kemajuan internet. Sehingga banyak dari masyarakat umum maupun pelajar atau mahasiswa mempelajarinya. Buku ini berisi materi berkaitan dengan algoritma seperti kondisi, perulangan dan array. Buku ini juga memperkenalkan beberapa bahasa seperti HTML, CSS, PHP dan SQL kepada pemula.

Banyak kode yang perlu dipelajari untuk membangun sebuah aplikasi berbasis web, namun semua itu biasanya tidak dikerjaan seorang diri. Apalagi banyak kerangka kerja yang memudahkan kita mengembangkan aplikasi. Dalam pembuatan alikasi berbasis web, proses pekerjaan minimal terbagi dua yaitu tampilan dan logika atau program. Untuk yang ingin belajar membuat tampilan dapat memperdalam keahlian desain tampilan halaman web yaitu mempelajari HTML dan CSS. Sedangkan yang ingin mempelajari logika atau program dapat memperdalam PHP dan basis data.

HTML adalah salah satu bahasa yang wajib dipelajari oleh pemula. Buku ini menerangkan fungsi HTML untuk mebuat tampilan halaman web. Tampilan seperti isi halaman maupun layout. Setiap materi sudah disusun sedemikan rupa sehingga mudah dipahami bagi pemula.

Setelah mempelajari HTML, pembaca akan diajak memahami pemrograman web secara menyeluruh. PHP merupakan bahasa pemrograman yang berjalan di sisi server. Bahasa pemrograman ini sangat populer sehingga mudah untuk mencari informasi seperti penanggulangan kesalahan. Dalam materi PHP penulis menyajikan materi dari dasar, sampai kebutuhan-

kebutuhan algoritma seperti array dan lainnya. Buku ini juga membahas tentang session dan cookies.

Hal yang paling penting dalam membuat aplikasi sistem informasi adalah manipulasi basis data yang dikenal dengan *Create*, *Read*, Update, Delete (CRUD). Pada buku ini ada contoh kasus aplikasi sederhana yang mengandung manipulasi basis data CRUD.

Materi terakhir di dalam buku ini adalah CSS sebagai cara untuk memperindah tampilan halaman web. Pada buku ini dibahas cara menuliskan kode CSS diantaranya *inline CSS*, internal CSS, dan eksternal CSS.

Terima kasih kepada Universitas Pembangunan Panca Budi Medan yang telah mendukung secara moril maupun materil. Terima kasih juga kepada rekan-rekan penulis dan keluarga besar kami yang telah memberikan dukungannya baik materil maupun moril. Akhir kata, kami ucapkan terima kasih kepada semua pihak yang tidak dapat kami sebutkan satu persatu dan telah membantu sampai terbitnya buku ini.

Penulis

DAFTAR ISI

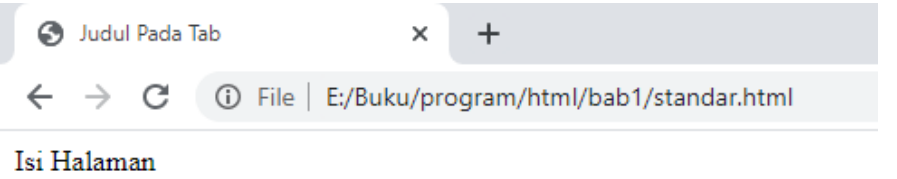
DAFTAR GAMBAR

HTML adalah singkatan dari *Hypertext Markup Language*. HTML merupakan bahasa yang digunakan untuk membuat tampilan halaman web. HTML dieksekusi pada *web browser*, sehingga tanpa web server pun kita dapat menjalankan kode HTML dan mendapatkan hasil tampilannya. HTML bukanlah bahasa pemrograman yang dapat memproses sebuah logika.

Kode-kode pada HTML disebut *tag*. Contohnya tag <p>, dibaca tag p. Tag HTML umumnya berpasangan seperti <html> dan </html>, namun adapula yang tunggal seperti
 atau
. Berikut ini adalah tag dasar HTML dari sebuah halaman *website*.

```
<html>
<head>
        <title>Judul Pada Tab</title>
</head>
        <body>
                Isi Halaman
        </body>
</html>
```

Untuk menjalankan kode HTML tidak diperlukan server. Sehingga untuk membukanya cukup dengan melakukan klik dua kali (*double click*) pada file HTML tersebut. Maka *web browser default* akan memproses kode HTML tersebut. Hal ini menandakan bahwa kode HTML diterjemahkan oleh *web browser* untuk menghasilkan tampilan. Pada Gambar 1.1 dapat dilihat alamat dari file yang dibuka oleh *web browser* pada bagian *address bar*. Hal tersebut terjadi karena file dibuka dengan cara mengklik dua kali (*double click*).



Gambar 1. 1. Tampilan Standar Kode HTML

Kode HTML tidak mengenal enter, tab atau lebih dari satu spasi. Sehingga tampilan tidak akan berubah meskipun anda menambahkan enter, tab atau banyak spasi. Spasi yang dianggap oleh HTML hanya satu spasi.

```
<title>Judul Pada Tab</title>
```

Sama Dengan

```
<title>
        Judul Pada Tab
</title>
```

Pada contoh kode di atas, kedua kode HTML akan menghasilkan tampilkan hasil yang sama. Penggunaan enter untuk menurunkan kode HTML ke baris baru ataupun tab untuk membuat kode maju ke kanan. Semua bertujuan agar kode program terlihat rapi dan juga mudah dibaca saja.

1.1 HEADING

Tag *heading* digunakan untuk menuliskan judul pada artikel halaman web. Seperti membuat artikel umumnya diperlukan judul dengan ukuran yang lebih besar daripada isi paragraf. Untuk membuat judul dengan HTML terdapat enam pilihan tag *heading* yaitu:

```
<H1>Judul Artikel</H1>
<H2>Judul Artikel</H2>
<H3>Judul Artikel</H3>
<H4>Judul Artikel</H4>
<H5>Judul Artikel</H5>
<H6>Judul Artikel</H6>
```

Secara tampilan tag <h1> akan menampilkan judul paling besar. Hal ini memiliki arti bahwa tag <h1> digunakan untuk judul besar. Kemudian tag <h2> adalah sub judulnya dan begitu seterusnya sampai tag <h6>.

Gambar 1. 2. Tampilan *Heading*

Pada Gambar 1.2 dapat dilihat perbedaan antara *heading* 1 sampai dengan *heading* 6. Bila diimplementasikan dengan benar maka halaman web akan tampil dengan baik.

1.2 PARAGRAF

Hal yang umum dilakukan untuk membuat artikel di halaman web adalah membuat paragraf. Tag yang digunakan untuk membuat paragraf adalah dan . Penulisan paragraf dapat dilihat pada kode program berikut:

```
<html>
<head>
        <title>Judul Pada Tab</title>
</head>
<body>
<h1>Judul Artikel</h1>
<p> Ini adalah sebuah paragraf tanpa perataan teks. Ini adalah sebuah paragraf tanpa perataan teks. Ini adalah sebuah paragraf tanpa perataan teks. Ini adalah sebuah paragraf tanpa perataan teks. Ini adalah sebuah paragraf tanpa perataan teks.</p>
</body>
</html>
```

Pada kode HTML di atas terlihat tag dasar HTML. Pada bagian tag <body> berisi artikel yang terdiri dari judul dan paragraf. Untuk membuat paragraf, tuliskan isi paragraf ditengah tag buka dan tag tutup. Dengan menggunakan tag tersebut maka antara satu paragraf dengan paragraf lainnya akan berpisah satu baris seperti umumnya paragraf. Kode program diatas menghasilkan tampilan seperti tersaji pada Gambar 1.3.

Gambar 1. 3. Tampilan Paragraf

Seperti terlihat pada Gambar 1.3, dapat dilihat sebuah artikel dengan satu paragraf. Antar paragraf akan berpisah satu baris, dimana satu paragraf dihitung dari tag buka sampai tag tutup.

1.3 FORMAT TEKS

Ada banyak format teks yang bisa kita buat dihalaman web layaknya di *Microsoft Word*. Namun pada kesempatan ini kita akan mempelajari beberapa format teks diantaranya:

1. Tebal
2. Miring
3. Garis Bawah
4. *Superscript*
5. *Subscript*

1.3.1 Tebal

Untuk membuat teks tebal tag yang digunakan adalah tebal. Teks yang akan ditebalkan letakkan di antara tag buka dan tutup.

1.3.2 Miring

Untuk membuat teks miring tag yang digunakan adalah miring. Teks yang akan dimiringkan letakkan di antara tag buka dan tutup.

1.3.3 Garis Bawah

Untuk membuat teks bergaris bawah tag yang digunakan adalah <u>garis bawah</u>. Teks yang akan dibuat garis bawah letakkan di antara tag buka dan tutup.

1.3.4 Superscript

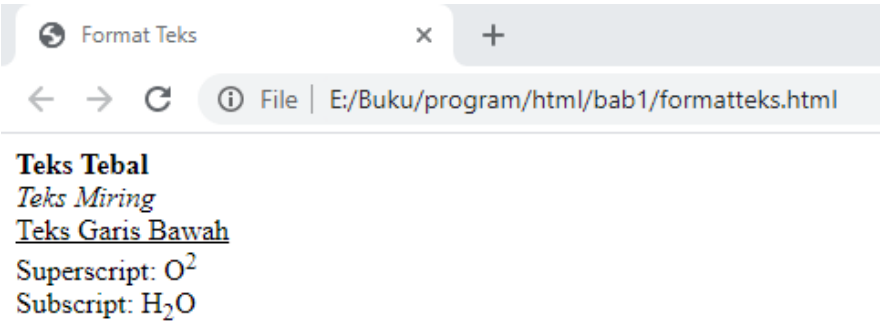
Untuk membuat teks *superscript* tag yang digunakan adalah ^{superscript}. Teks yang akan dibuat *superscript* letakkan di antara tag buka dan tutup.

1.3.5 Subscript

Untuk membuat teks *subscript* tag yang digunakan adalah _{subscript}. Teks yang akan dibuat *subscript* letakkan di antara tag buka dan tutup.

```
<html>
<head>
        <title>Format Teks</title>
</head>
<body>
<b>Teks Tebal</b><br>
<i>Teks Miring</i><br>
<u>Teks Garis Bawah</u><br>
Superscript O<sup>2</sup><br>
Subscript H<sub>2</sub>O
</body>
</html>
```

Pada kode HTML di atas dapat dilihat bahwa teks yang diletakkan diantara tag buka dan tutup. Maka akan mendapat efek atau format dari tag tersebut. Seperti tebal, miring, garis bawah dan lain-lain.



Gambar 1. 4. Tampilan Format Teks

Pada gambar di atas dapat dilihat hasil dari tag-tag yang diperuntukkan sebagai format teks. Seperti tebal, miring, garis bawah, dan lain-lain.

1.4 DAFTAR

Daftar atau *list* umumnya digunakan untuk menampilkan informasi agar mudah untuk dibaca. Pada *microsoft word* kita mengenal *bullets* dan *numbering*, sedangkan untuk halaman web dikenal dengan istilah daftar tidak berurut (*unordered list*) dan daftar berurut (*ordered list*).

1.4.1 Daftar Tidak Berurut

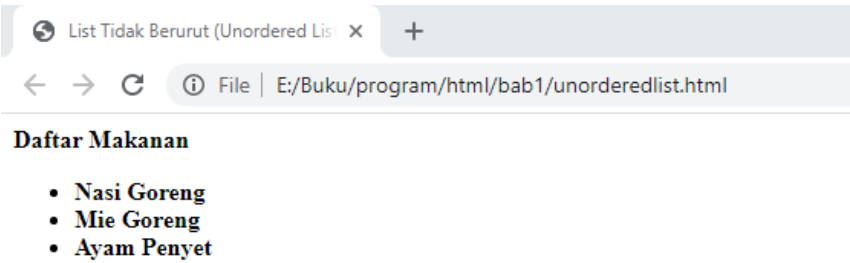
Daftar tidak berurut dikenal dengan istilah *unordered list* dengan tag yaitu <ul> dan . Setiap daftarnya diletakkan diantara <li> daftar . Daftar tak berurut ini seperti bullets di *microsoft word*.

```
<html>
<head>
        <title>List Tidak Berurut (Unordered List)</title>
</head>
<body>
        <h4>Daftar Makanan
        <ul>
                <li>Nasi Goreng</li>
                <li>Mie Goreng</li>
                <li>Ayam Penyet</li>
        </ul>
</body>
</html>
```

Kode program di atas lengkap dengan tag dasar HTML. Tag <ul> berperan sebagai induk dan tag <li> sebagai anak. Di dalam kode HTML kita akan menjumpai tag-tag seperti demikian. Dimana tag tersebut tidak sebatas tag buka dan tag tutup saja. Melainkan berisi tag anak yang dibutuhkan untuk mencapai tujuan dan diistilahkan dengan elemen.

Tag <ul> berfungsi menandai bahwa daftar yang dibuat adalah daftar tidak berurut. Sedangkan tag <li> berfungsi sebagai penanda dari daftar-daftar yang tersedia. Kedua tag saling membutuhkan untuk membuat daftar tidak berurut. Kode program diatas jika dibuka dengan

browser akan menapilkan *list* atau daftar sesuai format yang diberikan, seperti ditampilkan pada Gambar 1.5.



Gambar 1. 5. Tampilan Daftar Tidak Berurut

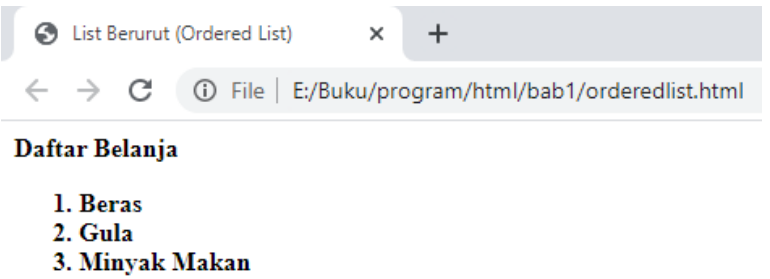
Pada Gambar 1.5, dapat dilihat hasil dari tag <ul> dan tag <li> yaitu sebuah daftar tidak berurut. Daftar tidak berurut ditandai dengan simbol (bulat kecil) disetiap daftarnya. Dapat dilihat setiap daftar akan dibuat baris baru. Hal ini menandakan tag <li> memiliki sifat membuat baris baru, meskipun dituliskan sejajar horizontal.

1.4.2 Daftar Berurut

Daftar berurut dikenal dengan *ordered list* dengan tag <ol> dan . Setiap daftarnya diletakkan diantara <li> daftar . Daftar berurut ini seperti *numbering* di *microsoft word*.

```
<html>
<head>
	<title>List Berurut (Ordered List)</title>
</head>
<body>
	<h4>Daftar Belanja
	<ol>
		<li>Beras</li>
		<li>Gula</li>
		<li>Minyak Makan</li>
	</ol>
</body>
</html>
```

Pada kode HTML di atas dapat dilihat perbedaan untuk membuat daftar berurut hanya pada tag <ol> atau *ordered list*. Untuk memudahkan dalam mengingatnya cukup memperhatikan awalan huruf tag tersebut. Kode program diatas akan menghasilkan seperti yang ditampilkan pada Gambar 1.6.



Gambar 1. 6. Tampilan Daftar Berurut

Pada Gambar 1.6, dapat dilihat tampilan daftar berurut. Daftar tersebut ditandai dengan awalan angka atau huruf. Penggunaan daftar berurut adalah untuk menggambarkan tingkat kepentingan dari isi daftar dan mengetahui jumlah daftar dengan mudah.

1.5 TABEL

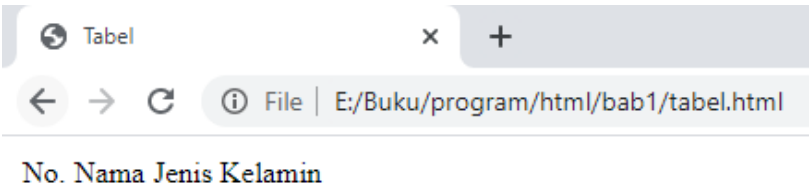
Tabel digunakan untuk menampilkan data yang banyak agar mudah untuk dibaca dan juga untuk merapikan tampilan (*layout*) halaman web. Membuat tabel di halaman web berarti menuliskan kode tabel HTML. Tag yang digunakan untuk membuat sebuah tabel sekurang-kurangnya ada tiga tag yaitu: <table> </table>, <tr> </tr>, dan <td> </td>. Berikut kode untuk membuat tabel sederhana:

```
<html>
<head>
	<title>Tabel</title>
</head>
<body>
	<table>
		<tr>
			<td>No.</td>
			<td>Nama</td>
			<td>Jenis Kelamin</td>
		</tr>
	</table>
</body>
</html>
```

Agar mudah paham dalam membuat tabel, anda juga dapat menuliskan setiap kolom atau sel ke kanan sesuai tampilan sebenarnya. Tag <tr> berarti meminta baris baru kemudian <td> adalah kolom atau sel nya. Kode diatas dapat ditulis juga seperti dibawah. Secara penulisan kode, penulisan ini kurang baik dari pada kode HTML di atas. Namun untuk pemula cara ini cukup baik agar mudah dalam membuat sebuah tabel.

```
<html>
<head>
	<title>Tabel</title>
</head>
<body>
	<table>
		<tr> <td>No.</td> <td>Nama</td> <td>Jenis Kelamin</td>
</tr>
	</table>
</body>
</html>
```

Pada kode HTML di atas terlihat perbedaan dari kode sebelumnya. Namun hasil yang ditampilkan akan tetap sama. Sesuai dengan penjelasan sebelumnya bahwa kode HTML mengabaikan enter. Berikut tampilan tabel tanpa border sesuai kode program diatas seperti disajikan pada Gambar 1.7.



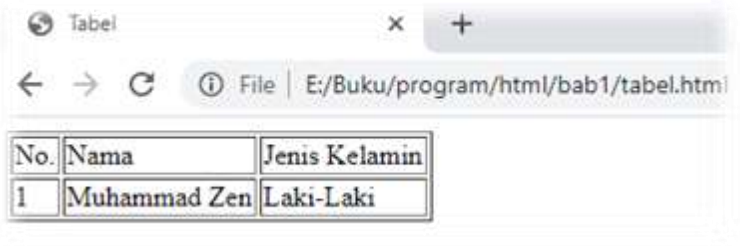
Gambar 1. 7. Tabel Tanpa Border

Untuk menambahkan *border* cukup dengan menambahkan atribut border pada tag <table>. Brikut ini contohnya kode programnya :

```
<html>
<head>
	<title>Tabel</title>
</head>
<body>
	<table border="1">
		<tr>
			<td>No.</td>
			<td>Nama</td>
			<td>Jenis Kelamin</td>
		</tr>
		<tr>
			<td>1</td>
			<td>Muhammad Zen</td>
			<td>Laki-Laki</td>
		</tr>
	</table>
</body>
</html>
```

Pada kode program di atas atribut border=1 diletakkan pada tag <table>, maka tabel tersebut akan menampilkan border. Semakin besari nilai atribut tersebut, maka semakin tebal border tabel tersebut.

Pada kebutuhan lain, border sengaja dibuat tidak tampak. Hal tersebut berguna untuk merapikan tampilan dan bukan untuk menyajikan data yang banyak seperti guna tabel umumnya.



Gambar 1. 8. Tabel Menggunakan Border

Pada Gambar 1.8 tampak border sebuah tabel. Garis border terlihat seperti dua garis. Hal ini dapat diubah menjadi satu garis dengan mengatur atribut cellspacing=0. Terdapat atribut lain pada tag <tabel> seperti perataan. Sebuah tabel dapat ditampilkan rata kiri, tengah, dan juga rata kanan dengan atribut align.

1.5.1 Mewarnai Tabel

Mewarnai tabel sangat berguna untuk memperindah atau meperjelas tabel. Untuk mewarnai tabel tambahkan properti bgcolor pada tag <table> atau <tr> atau <td>, kemudian sesuaikan *value* (warna) sesuai keinginan. Untuk mewarnai seluruh tabel properti bgcolor di tag <table>, sedangkan untuk mewarnai per baris di tag <tr>. Terakhir untuk mewarnai setiap sel properti bgcolor di tag <td>.

```
<html>
<head>
	<title>Tabel</title>
</head>
<body>
	<table border="1" bgcolor="green">
		<tr>
			<td>No.</td>
			<td>Nama</td>
			<td>Jenis Kelamin</td>
		</tr>
		<tr>
			<td>1</td>
			<td>Muhammad Zen</td>
			<td>Laki-Laki</td>
		</tr>
	</table>
</body>
</html>
```

Pada kode HTML di atas tampak atribut bgcolor di tuliskan pada tag <table>. Maka semua tabel akan berwarna. Dengan kata lain, tabel tersebut memiliki satu warna, seperti terlihat pada Gambar 1.9.

Gambar 1. 9. Mewarnai Seluruh Tabel

Selain mewarnai seluruh tabel, anda juga dapat mewarnai baris demi baris tabel dengan warna-warna yang berbeda.

```
<html>
<head>
	<title>Tabel</title>
</head>
<body>
	<table border="1">
		<tr bgcolor="#ccddcc">
			<td>No.</td>
			<td>Nama</td>
			<td>Jenis Kelamin</td>
		</tr>
		<tr>
			<td>1</td>
			<td>Muhammad Zen</td>
			<td>Laki-Laki</td>
		</tr>
		<tr bgcolor="#dddddd">
			<td>2</td>
			<td>Indah</td>
```

```
                <td>Perempuan</td>
            </tr>
        </table>
</body>
</html>
```

Pada kode HTML di atas dapat dilihat bahwa atribut bgcolor juga dapat dituliskan pada tag <tr>. Sehingga warna latar belakang hanya sebatas satu baris saja. Sehingga warna tiap baris dapat berbeda-beda.

Merwarnai baris satu persatu memang cukup merepotkan. Namun semua itu akan mudah bila dibantu oleh bahasa pemrograman. Pada materi HTML ini pemahaman tentang desain tampilan yang diutamakan. Untuk mengoptimalkan hal tersebut, maka perlu dibantu program.

Gambar 1. 10. Mewarnai Baris Tabel

Contoh di atas mewarnai baris dengan nilai warna berupa kode hexadesimal. Penulisannya diawali tanda #. Formatnya adalah "RRGGBB", dua untuk nilai *red*, dua untuk *green* dan dua untuk *blue*. Terakhir anda dapat mewarnai setiap sel tabel.

```
<html>
<head>
        <title>Tabel</title>
</head>
<body>
        <table border="1">
                <tr>
                        <td>No.</td>
                        <td bgcolor="#dddddd">Nama</td>
```

```
                <td>Jenis Kelamin</td>
            </tr>
            <tr>
                <td>1</td>
                <td bgcolor="#dddddd">Muhammad Zen</td>
                <td>Laki-Laki</td>
            </tr>
            <tr>
                <td>2</td>
                <td bgcolor="#dddddd">Indah</td>
                <td>Perempuan</td>
            </tr>
        </table>
</body>
</html>
```

Pada kode HTML di atas, dapat dilihat cara mewarnai sel. Untuk mendapatkan warna yang sama pada sebuah kolom. Perlu untuk mewarnai seluruh sel pada kolom tersebut. Pada kode di atas, sel yang diwarnai adalah seluruh sel dengan urutan kedua. Sehingga tampak kolom ke dua seperti diwarnai dengan warna yang sama.

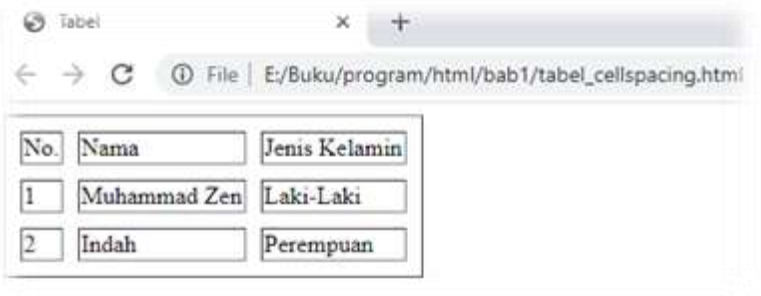
Gambar 1. 11. Mewarnai Sel Tabel

1.5.2 Cellspacing

Cellspacing berfungsi memberi jarak setiap sel. Jarak yang dimaksud adalah jarak sel dengan sel di sekitarnya. Atribut yang digunakan adalah *cellspacing*. Berikut ini contohnya:

```
<html>
<head>
	<title>Tabel</title>
</head>
<body>
	<table border="1" cellspacing="10">
		<tr>
			<td>No.</td>
			<td>Nama</td>
			<td>Jenis Kelamin</td>
		</tr>
		<tr>
			<td>1</td>
			<td>Muhammad Zen</td>
			<td>Laki-Laki</td>
		</tr>
		<tr>
			<td>2</td>
			<td>Indah</td>
			<td>Perempuan</td>
		</tr>
	</table>
</body>
</html>
```

Pada kode HTML di atas atribut *cellspacing* diletakkan pada tag <table>. Fungsi *cellspacing* tersebut hampir sama dengan margin. Dimana jarak yang diatur adalah jarak luar kotak atau sel. Semakin besar nilai yang diberikan pada atribut *cellspacing* maka semakin lebar pula jarak antar sel tersebut.



Gambar 1. 12. *Cellspacing* Tabel

Pada Gambar 1.12 tampak sebuah tabel dengan nilai *cellspacing* sepuluh. Apabila tabel tersebut diperuntukkan untuk mengatur tata letak. Maka cukup dengan menghilangkan border tabel tersebut. Maka isi antar sel akan semakin berjauhan bila nilai *cellspacing* diperbesar.

1.5.3 Cellpadding

Cellpadding fungsinya untuk mengatur jarak isi sel dengan bordernya. *Padding* berarti jarak kotak sel terhadap isi sel. *Cellpadding* merupakan kebalikan dari *cellspacing*. Semakin besar nilai dari *cellpadding* maka jarak border sel dengan isi semakin lebar. Berikut ini contoh kode programnya:

```
<html>
<head>
	<title>Tabel</title>
</head>
<body>
	<table border="1" cellpadding="10">
		<tr>
			<td>No.</td>
			<td>Nama</td>
			<td>Jenis Kelamin</td>
		</tr>
		<tr>
			<td>1</td>
```

```
                <td>Muhammad Zen</td>
                <td>Laki-Laki</td>
            </tr>
            <tr>
                <td>2</td>
                <td>Indah</td>
                <td>Perempuan</td>
            </tr>
        </table>
</body>
</html>
```

Pada kode HTML di atas atribut *cellpadding* diletakkan pada tag <table>. Nilai *cellpadding* yang diberikan pada tabel tersebut adalah sepuluh. Semakin besar nilainya, maka semakin jauh jarak antara isi sel dengan bordernya.

Gambar 1. 13. *Cellspadding* Tabel

Pada Gambar 1.13 terlihat jarak isi sel (teks) dengan border tabel cukup jauh. Nilai yang diberikan pada atribut *cellpadding* tabel di atas adalah sepuluh. Semakin besar nilai yang diberikan, maka semakin jauh jarak isi sel dengan bordernya.

1.6 ATRIBUT

Atribut pada HTML digunakan untuk mengatur tampilan HTML. Atribut dapat juga diartikan sebagai pengaturan tambahan sebuah tag HTML. Atribut tersebut diletakkan pada tag buka HTML. Beberapa tag mempunyai properti yang sama namun tidak sedikit pula yang berbeda. Formatnya **namaatribut=”nilai”**. Berikut ini contohnya:

```
<h1 align=”center”> Judul Artikel</h1>
<input type=”number” name=”no_telp”>
```

Pada kode HTML di atas dapat dilihat bahwa atribut adalah kode tambahan yang letaknya dalam tag buka. Pada atribut tersebut yang paling menonjol adalah adanya tanda sama dengan. Bagian sebelah kiri adalah nama atribut, sedangkan bagian kanan adalah nilai.

Pada baris pertama tag HTML yang dituliskan adalah tag <h1>. Tag tersebut dapat pengaturan tambahan melalui atribut align (perataan). Atribut tersebut berfungsi untuk mengatur perataan teks. Seperti umumnya, perataan terdiri dari rata kiri, rata kanan, rata kiri-kanan dan rata tengah.

Pada baris kedua, tag yang dituliskan adalah tag <input>. Tag tersebut adalah tag tunggal. Dimana tag tersebut tidak berpasangan dengan tag tutupnya. Pada tag ini diperlihatkan bahwa atribut sebuah tag cukup banyak. Pada sebuah tag boleh dituliskan lebih dari satu atribut dengan cara dipisahkan oleh spasi.

1.7 *UNIFORM RESOURCE LOCATOR* (URL)

URL fungsinya untuk menemukan alamat suatu file. Baik berupa file HTML, PHP, CSS, JS, gambar maupun yang lain. Ada dua jenis URL yaitu absolut URL dan relatif URL. Memahami cara menggunakan URL sangat penting dalam belajar HTML. Karena jika URL salah, maka file akan gagal dimuat.

Secara umum URL diketikkan di *address bar web browser*. Namun di dalam kode HTML, URL dituliskan sebagai nilai dari sebuah atribut. Misalnya pada tag <a>, URL digunakan sebagai nilai dari atribut href. Masih banyak lagi tag-tag HTML yang menggunakan nilai URL.

Ada dua jenis URL yang sering digunakan yaitu absolut URL dan relatif URL. Absolut URL berarti alamat tersebut tetap. Sedangkangkan Relatif URL berarti URL tersebut relatif atau bisa berubah tergantung yang memanggil alamat tersebut.

1.7.1 Absolut URL

Absolut URL menemukan file dari nama domain kemudian folder dan file itu berada. Contohnya : http://localhost/buku/url.html. URL tersebut adalah absolut URL untuk menemukan alamat file url.html yang di terlusuri dari root folder yaitu buku. Pada contoh URL di atas terlihat bahwa file url. html berada dalam folder buku. URL tersebut punya ciri-ciri awalan http://. Kemudian diikuti nama domain atau localhost jika lokal. Ini berarti file HTML yang akan diakses dengan absolut URL harus berada di folder htdocs. URL jenis ini sangat cocok untuk aplikasi yang besar dan kompleks. Pengalamatan-pengalamatan sumber jauh lebih mudah ditelusuri menggunakan absolut URL untuk aplikasi yang sangat besar.

1.7.2 Relatif URL

Berbeda dengan absolut URL, relatif URL menemukan file berdasarkan lokasi file pemanggil. Sehingga bentuk URL akan berubah-ubah terhadap siapa yang memanggil. Contohnya seperti berikut ini.

../gambar/bg.jpg

URL di atas adalah relatif url untuk menemukan file bg.jpg yang ditelusuri dari alamat file pemanggil. Langkah pertama keluar folder ".." lalu masuk kedalam folder gambar "/gambar" terakhir nama filenya "/bg.jpg".

1.8 HYPERLINK (LINK)

Sebuah sistem atau aplikasi terdiri dari beberapa halaman yang terhubung dengan halaman utama. Link berfungsi untuk berpindah dari satu halam ke halaman lain. Tag yang digunakan link adalah tag <a> dan . Format yang digunakan untuk berpindah kehalaman adalah Objek . URL

diisi alamat file yang dituju, misalnya HTML, PHP atau gambar. Objek yang digunakan dapat berupa teks, gambar, atau tombol.

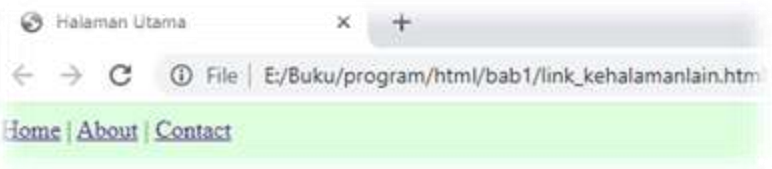
URL yang digunakan pada *hyperlink* kali ini adalah jenis relatif URL. Untuk program sederhana atau awal belajar pemrograman, URL ini mudah dipelajari. Karena relatif URL tidak perlu meletakkan file HTML dalam file htdocs. URL pun menjadi lebih pendek ketika file yang dipanggil berada dalam satu folder dengan file yang memanggil.

1.8.1 *Link* ke Halaman Lain

Berpindah ke halaman lain adalah fungsi utama dari *link*. *Link* paling umum terdapat pada halaman utama. Pada halaman utama *link* dibuat untuk memanggil halaman lain yang berisi informasi yang tidak ditampilkan pada halaman utama. Berikut ini contohnya:

```
<!DOCTYPE html>
<html>
<head>
        <title>Halaman Utama</title>
</head>
<body bgcolor="#ddffdd">
<a href="home.html">Home</a> | <a href="about.html">About</a> | <a href="contact.html">Contact</a>
</body>
</html>
```

Pada kode program di atas dapat dilihat nilai dari atribut href berisi URL. Halaman ini menandakan *link* di atas adalah *link* ke bagian halaman. Jenis URL yang digunakan adalah relatif URL. Relatif url ini ditandai dengan URL tidak diawali *localhost* atau nama domain. URL pada atribut href yang berisi nama file saja mengindikasikan bahwa file tujuan berada dalam satu folder dengan pemanggilnya.

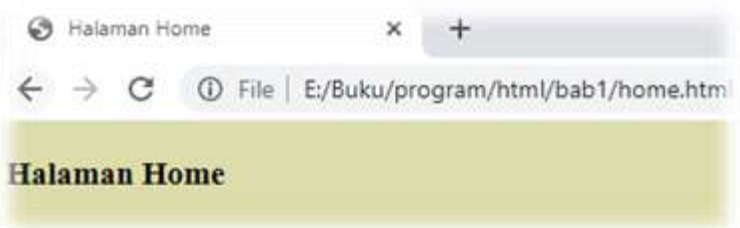


Gambar 1. 14. Link ke Halaman Lain

Pada halaman ini dapat dilihat tiga buah menu yaitu *Home*, *About* dan *Contact*. Masing-masing menu tersebut akan menuju ke halaman lain. File di atas di akses atau dibuka dengan cara mengklik dua kali file tersebut. Sehingga tampil *link* seperti di atas.

```
<!DOCTYPE html>
<html>
<head>
        <title>Halaman Home</title>
</head>
<body bgcolor="#ddddaa">
<h3>Halaman Home</h3>
</body>
</html>
```

Kode program di atas hanya contoh halaman saja. Halaman tersebut berjudul halaman *home* dengan isi berupa teks bertuliskan Halaman *Home*.



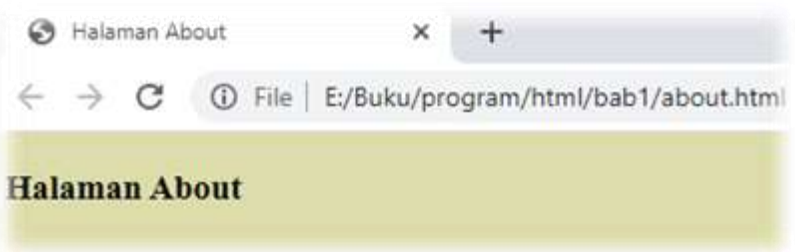
Gambar 1. 15. Halaman *Home*

Gambar 1.15 menunjukkan tampilan halaman *home* ketika menu *home* pada Gambar 1.14 diklik. Ketika berpindah halaman, URL pada *address bar web browser* juga ikut berubah. Dapat dilihat file yang sedang dibuka adalah home.html.

Saat terjadi *error* atau halaman tidak ditemukan, penting untuk memperhatikan URL pada *address bar web browser*. Umumnya kesalahan dapat diidentifikasi melalui URL yang di tampilkan di *address bar*.

```
<!DOCTYPE html>
<html>
<head>
        <title>Halaman About</title>
</head>
<body bgcolor="#ddddaa">
<h3>Halaman About</h3>
</body>
</html>
```

Kode HTML di atas hanya untuk menampilkan halaman sederhana yang bertuliskan halaman *about*. Kode HTML di atas sudah lengkap dengan tag dasar HTML. Tag <DOCTYPE html> di atas berperan dalam deklarasi versi HTML yaitu HTML 5. Sangat penting untuk mendeklarasikan HTML 5 karena beberapa fitur tidak bekerja karena kurangnya hal tersebut.

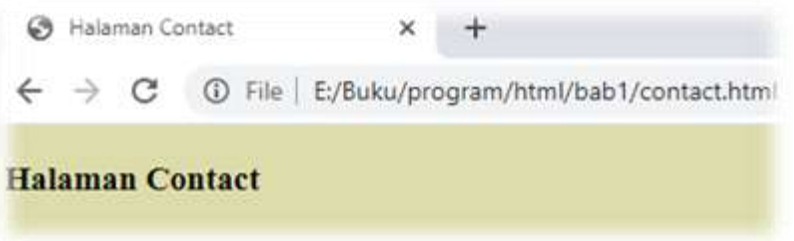


Gambar 1. 16. Halaman *About*

Pada Gambar 1.16, dapat dilihat tampilan halaman *about*. Halaman tersebut hanya bertuliskan halaman *about*. Karena pada materi ini hanya membuktikan link yang dibuat berhasil. Perhatikan juga URL di atas. URL tersebut menunjukkan file yang dibuka adalah about.html.

```
<!DOCTYPE html>
<html>
<head>
        <title>Halaman Contact</title>
</head>
<body bgcolor="#ddddaa">
<h3>Halaman Contact</h3>
</body>
</html>
```

Kode HTML di atas dibuat untuk menampilkan halaman *contact*. Halaman ini biasanya berisi informasi kontak yang dapat dihubungi. Namun halaman serderhana ini hanya difungsikan untuk menampilkan tulisan halaman *contact* saja.



Gambar 1. 17. Halaman *Contact*

Pada Gambar 1.17, tampak URL membuka file bernama contact.html. Membagi-bagi halaman ini merupakan teknik dalam desain web yang baik. Apabila terlalu banyak informasi yang akan ditampilkan dalam halaman web, maka baiknya membagi informasi tersebut ke dalam halaman-halaman lain. Sehingga fungsi *link* adalah menyatukan kembali atau membuat akses terhadap informasi yang terpisah-pisah tersebut.

1.8.2 *Link* ke Bagian Halaman

Sebenarnya tidak semua *link* berpindah ke halaman lain. Ada *link* yang difungsikan untuk menunjuk bagian tertentu dari sebuah halaman. Contohnya ketika berkunjung kehalaman web tertentu. Ketika menu di *klik* maka halaman tersebut akan bergeser kebawah tepat dimana informasi tersebut di tampilkan. Hal tersebut dapat dijadikan ilustrasi meskipun ada tambahan kode lain.

Link ke bagian halaman punya ciri-ciri tanda pagar # di URL. Pernahkah anda melihat tanda pagar pada alamat URL? tanda pagar tersebut menandakan *link* ke bagian halaman. Setelah tanda pagar adalah nama dari alamatnya. Untuk lebih jelasnya perhatikan kode program berikut:

```
<!DOCTYPE html>
<html>
<head>
        <title>Halaman Utama</title>
</head>
<body bgcolor="#ddffdd">
<a name="atas"></a>
<a href="#home">Home</a> | <a href="#about">About</a> | <a
href="#contact">Contact</a>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><
br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><
br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br>
<a name="home"> </a> Ini adalah halaman home. Halaman home
menampilkan informasi umum. <a href="#atas">Ke atas</a>

<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><
br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
```

```
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><
br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br>
<a name="about"></a>
Ini adalah halaman about. Halaman about menjelaskan tentang website ini.
<a href="#atas">Ke atas</a>

<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><
br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><
br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br>
<a name="contact"></a>
Ini adalah halaman contact. Halaman contak menampilkan nomor telepon,
email atau alamat yang dapat dihubungi. <a href="#atas">Ke atas</a>
</body>
</html>
```

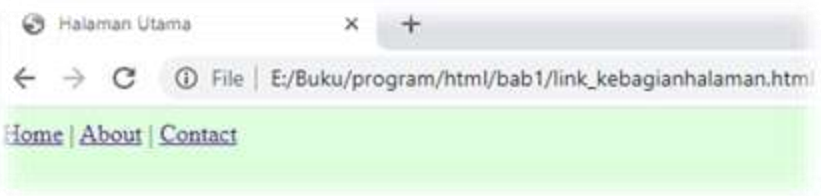
Pada kode HTML di atas terdapat tiga *link* kebagian halaman masing-masing adalah *home*, *about* dan *contact*. Masing-masing URL diawali tanda pagar yaitu #home, #about dan #contact. Perhatikan link ke bagian halaman beriku. Kode berikut adalah bagian pemanggil. Tag yang digunakan adalah tag <a> dan atribut nya adalah href.

```
<a href="#home">Home</a>
```

Tag yang digunakan sebagai tujuan *link* juga tag <a>. Namun atributnya berbeda yaitu atribut name. Nilai dari atribut name tersebut tidak perlu menggunakan tanda pagar lagi.

```
<a name="home"> </a>
```

Saat *link home* diklik maka halaman otomatis bergulir kebawah tepat pada tujuan *link* tersebut. Tujuan tersebut adalah tag <a> dengan atribut name. Perhatikan halaman awal berikut.



Gambar 1. 18. Pertama dijalankan atau *Link* ke Atas diklik

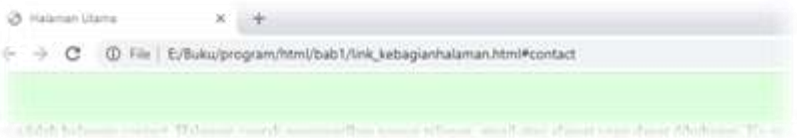
Pada Gambar 1.18, dapat dilihat *link* atau menu yaitu *home*, *about*, dan *contact*. *Link* tersebut merupakan *link* ke bagian halaman. Ketika *link* tersebut diklik maka secara otomatis halaman akan bergulir kebawah. Ketika *link home* diklik maka halaman akan bergulir kebagian *home*, begitu juga dengan *about* dan *contact*.

Gambar 1. 19. *Link Home* diklik

Perhatikan alamat URL di atas, terlihat tanda #home setelah nama file link_kebagianhalaman.html. Bagian ini otomatis naik ke atas atau bergulir ketika *link* atau menu *home* di klik. Diujung tulisan terlihat *link* dengan label Ke atas. Jika diklik maka menu akan muncul kembali (bergulir ke atas).

Gambar 1. 20. *Link About* diklik

Perhatikan alamat URL di atas, terlihat tanda #about setelah nama file link_kebagianhalaman.html. Bagian ini otomatis naik ke atas atau bergulir ketika link atau menu *about* di klik. Diujung tulisan terlihat *link* dengan label Ke atas. Jika diklik maka menu akan muncul kembali (bergulir ke atas).



Gambar 1. 21. *Link Contact* diklik

Perhatikan alamat URL di atas, terlihat tanda #contact setelah nama file link_kebagianhalaman.html. Bagian ini otomatis naik ke atas atau bergulir ketika *link* atau menu *contact* di klik. Diujung tulisan terlihat link dengan label Ke atas. Jika diklik maka menu akan muncul kembali (bergulir ke atas).

1.9 FORMULIR

Interaksi pengguna dengan sistem atau aplikasi berbasis web salah satunya dapat menggunakan formulir (*form*). Formulir sebagai salah satu sarana input atau masukan bagi sistem. Berikut ini tag form di HTML <form> </form>. Atribut yang penting dari *tag* <form> adalah *action* dan *method*. *Action* berisi URL data akan dikirimkan sedangkan *method* adalah metode pengiriman. Sedangkan data yang akan dikirim tempatnya di elemen *form*.

<form action=”url” method=”POST atau GET”> </form>.

1.9.1 Elemen *Form*

Elemen *form* adalah *user interface* yang dapat berinteraksi dengan pengguna, baik di klik mauput di input. Berikut ini beberapa contoh element *form*.

1. Input *Text*

Input *text* berfungsi untuk menerima input *text* dari pengguna. *Tag* HTML yang digunakan adalah *tag* <input> dengan *attribut* type=”text” . Berikut ini contohnya.

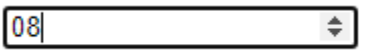
```
<input type="text" name="nama" maxlength="9" required="">
```

Gambar 1. 22. Input *Text*

2. Input *Number*

Input *number* berfungsi untuk menerima input angka dari pengguna. *Tag* HTML yang digunakan adalah *tag* <input> dengan *attribut* type=”number” . Berikut ini contohnya.

```
<input type="number" name="nohp" maxlength="12" required="">
```

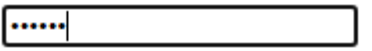


Gambar 1. 23. Input *Number*

3. Input *Password*

Input *password* berfungsi untuk menerima input *password* dari pengguna. *Tag* HTML yang digunakan adalah *tag* <input> dengan *attribut* type=”password” . Berikut ini contohnya.

```
<input type="password" name="password" required="">
```

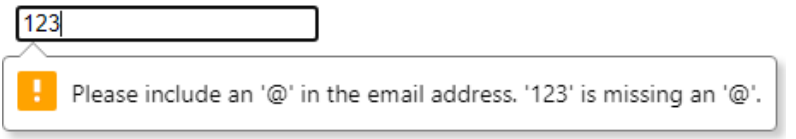


Gambar 1. 24. Input *Password*

4. Input Email

Input email berfungsi untuk menerima input email dari pengguna. *Tag* HTML yang digunakan adalah *tag* <input> dengan *attribut* type=”email” . Berikut ini contohnya.

```
<input type="email" name="email" required="">
```

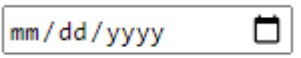


Gambar 1. 25. Input Email

5. Input *Date*

Input *date* berfungsi untuk menerima input tanggal dari pengguna. *Tag* HTML yang digunakan adalah *tag* <input> dengan *attribut* type=”date” . Berikut ini contohnya.

```
<input type="date" name="tanggal" required="">
```

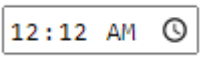


Gambar 1. 26. Input *Date*

6. Input *Time*

Input *time* berfungsi untuk menerima input waktu dari pengguna. *Tag* HTML yang digunakan adalah *tag* <input> dengan *attribut* type=”time” . Berikut ini contohnya.

```
<input type="time" name="jam" required="">
```



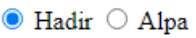
Gambar 1. 27. Input *Time*

7. Radio

Radio berfungsi untuk menerima pilihan dari pengguna, dimana pengguna hanya bisa memilih salah satu. *Tag* HTML yang digunakan

adalah *tag* <input> dengan *attribut* type=”radio” . Untuk radio nilai atau *value* diberikan di kode. Berikut ini contohnya.

```
<input type="radio" value="hadir" name="absensi" checked="" > Hadir
<input type="radio" value="alpa" name="absensi"> Alpa
```

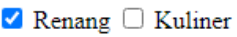


Gambar 1. 28. Radio

8. *CheckBox*

CheckBox berfungsi untuk menerima pilihan dari pengguna, dimana pengguna bisa memilih beberapa pilihan sekaligus. *Tag* HTML yang digunakan adalah *tag* <input> dengan *attribut* type=”checkbox” . Untuk *checkbox* nilai atau *value* diberikan di kode. Hampir sama dengan radio, namun berbeda di *attribut name*. Berikut ini contohnya.

```
<input type="checkbox" value="Renang" name="hobi1" checked="" >
Renang
<input type="checkbox" value="Kuliner" name="hobi2"> Kuliner
```

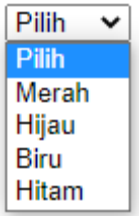


Gambar 1. 29. *Checkbox*

9. *Select*

Select berfungsi untuk menerima pilihan dari pengguna. Pilihan tersusun ke bawah jika diklik. Berikut ini contohnya.

```
<select name="warna">
        <option value="">Pilih</option>
        <option value="Merah">Merah</option>
        <option value="Hijau">Hijau</option>
        <option value="Biru">Biru</option>
        <option value="Hitam">Hitam</option>
</select>
```



Gambar 1. 30. *Select*

10. *DataList*

Datalist berfungsi untuk menerima pilihan dari pengguna. Hampir sama dengan *select*, *datalist* memiliki kelebihan dapat dicari atau diketik. Pilihan tersusun ke bawah jika diklik. Berikut ini contohnya.

```
<input list="id_warna" name="pilihwarna">
<datalist id="id_warna">
        <option value="Merah">Merah</option>
        <option value="Hijau">Hijau</option>
        <option value="Biru">Biru</option>
        <option value="Hitam">Hitam</option>
</datalist>
```

Gambar 1. 31. *Datalist*

11. *Textarea*

Textarea berfungsi untuk menerima input *text* dari pengguna. Hampir sama dengan input *text*, namun ukurannya bisa diperbesar.

Textarea bisa digunakan sebagai editor. *Tag* HTML yang digunakan adalah *tag* <textarea> </textarea> dengan *attribut* cols dan rows . Berikut ini contohnya.

Gambar 1. 32. *Textarea*

12. *Button Submit*

Button submit berfungsi untuk memulai pengiriman data dalam *form*. Bentuk nya sama dengan *button* biasa. *Tag* HTML yang digunakan adalah *tag* <button> </button> dengan *attribut* type=”submit”. Berikut ini contohnya

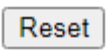
```
<button type="submit" name="ok">OK</button>
```

Gambar 1. 33. *Button Submit*

13. *Button Reset*

Button reset berfungsi untuk mengosongkan atau mengatur ulang data dalam *form*. Bentuk nya sama dengan *button* biasa. *Tag* HTML yang digunakan adalah *tag* <button> </button> dengan attribut type=”reset”. Berikut ini contohnya

```
<button type="reset">Reset</button>
```



Gambar 1. 34. *Reset*

14. *Button* Biasa

Button biasa digunakan untuk *link*. *Tag* HTML yang digunakan adalah *tag* <button> </button> dengan *attribut* type=”button”. Bila tidak diberi *link* atau aksi, *button* ini tidak berfungsi. Berikut ini contohnya.

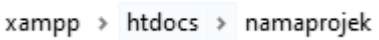
<button type="button" name="ok">Cancel</button>

Gambar 1. 35. *Button*

BAB 2
PENGENALAN PHP

2.1 PENDAHULUAN

Bahasa pemrograman PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada awalnya PHP merupakan singkatan dari *Personal Home Page*. Namun sekarang PHP dikenal singkatan dari *Hypertext Prepocessor*. Perintah di dalam PHP tidak *case sensitive*, artinya dapat dituliskan dengan huruf besar maupun huruf kecil tanpa *error*. Untuk menjalankan kode PHP dibutuhkan *server* misalnya *apache*. Kode program diletakkan dalam folder *htdocs* untuk *server* lokal yang akan digunakan untuk belajar kali ini.



Gambar 2. 1. Folder Proyek

2.2 KODE DASAR PHP

Sama seperti HTML, PHP juga memiliki kode dasar, untuk memulai belajar PHP harus memahami kode dasar berikut ini. Kode tersebut akan sering digunakan setiap membuat program.

Tabel 2. 1. Kode Dasar PHP

No.	Kode	Fungsi
1.	<?php // kode ?>	Untuk pembuka dan penutup PHP. Setiap kode PHP harus berada didalam buka dan tutup PHP.
2.	;(titik koma)	Akhir dari perintah PHP.
3.	“” (petik dua)	Input *string*. Dapat menampilkan isi variabel.
4.	‘’ (petik satu)	Input *string*. Tidak dapat menampilkan isi variabel.
5.	Echo	Mencetak dilayar.

Berikut ini contoh pengenalan kode PHP dalam sebuah file berisi HTML. Setiap kali kode PHP menyisip di kode HTML maka harus dibungkus dengan pembuka dan penutup nya. Jika tidak maka akan ditampilkan kesalahan.

```
<!DOCTYPE html>
<html>
<head>
        <title>Pengenalan PHP</title>
</head>
<body>
<?php
        echo "<h1 align='center'> Halo Dunia</h1>";
?>
<hr>
<?php
        ECHO 'Saya sedang belajar PHP';
?>
</body>
</html>
```

Pada kode program di atas tampak kode PHP berada dalam *tag* buka PHP (<?php) dan *tag* tutup PHP (?>) . Perintah untuk mencetak nilai atau *string* menggunakan PHP adalah *echo*. Perintah *echo* bisa mencetak teks biasa atau *tag* HTML seperti kode program di atas. Perintah *echo* di atas juga mencetak *tag* <h1>.

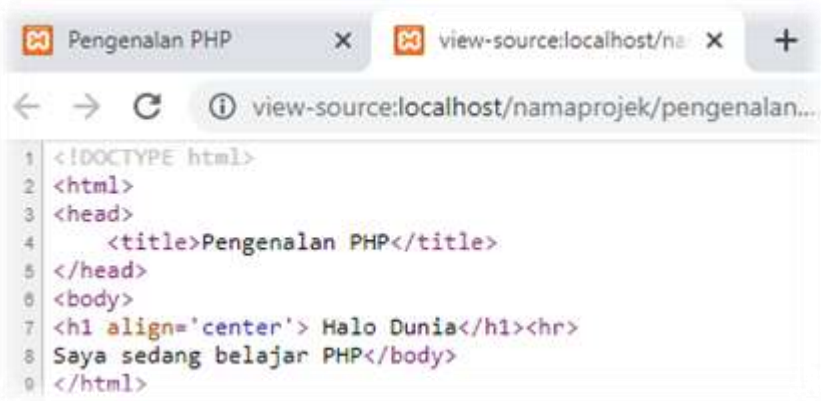
Untuk mencetak *string* perintah *echo* harus diikuti oleh kutip ganda atau kutip tunggal. Pada contoh kode program di atas dibagian atribut *align* menggunakan kutip tunggal. Karena perintah *echo* tersebut sudah menggunakan kutip dua. Jika tidak demikian penulisannya maka akan terjadi kesalahan. Kecuali kondisi di atas dibalik, yaitu perintah *echo* menggunakan kutip tunggal maka nilai atribut menggunakan kutip ganda.

Pada kode PHP yang kedua perintah *echo* menggunakan huruf besar. Hal tersebut membuktikan bahwa perintah PHP tidak *case sensitive*. Penggunaan kutip tunggal mengkhususkan untuk mencetak *string*. Apabila terdapat

variabel dalam perintahnya, maka variabel itu akan dicetak bukan nilainya. Pada Gambar 2.2 dapat dilihat hasil tampilan dari kode program di atas.

Gambar 2. 2. Pengenalan PHP

Perhatikan tampilan halaman web pada Gambar 2.2. Apa yang terlihat pada halaman web tersebut adalah hasil terjemahan dari kode HTML yang dilakukan oleh web *browser*. Kode PHP tidak akan sampai pada web *browser*. Kode PHP hanya diproses oleh *server* dan hasil dikembalikan dalam bentuk kode HTML atau *client script* lain.



Gambar 2. 3. Kode Sumber

Perhatikan Gambar 2.3, pada kode sumber dari halaman tersebut tidak terlihat kode PHP. Kode PHP diterjemahkan di server dan tidak dapat dilihat di web *browser*. Kode PHP diproses ketika permintaan dikirim (*request*) dan hasilnya dikembalikan (*serponse*) kepada web *browser*.

Untuk lebih memahami kapan kode PHP diproses beberapa diantaranya yaitu ketika *request* dikirim pada saat *link* di klik, URL diketikkan atau form di submit. Berbeda dengan pemrograman berbasis *cilent*, pada PHP tidak ada *event* atau kejadian kapan kode program dijalankan.

BAB 3
VARIABEL

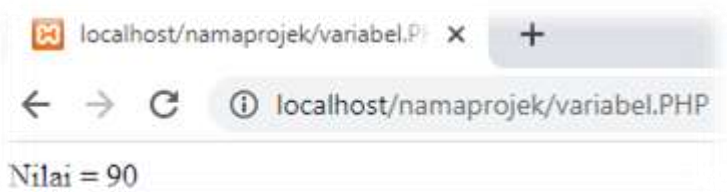
Variabel adalah tempat untuk menyimpan data di memori untuk diproses oleh program. Pada PHP, tipe data untuk variabel tidak didefinisikan. Tipe data otomatis ditetapkan ketika variabel pertama kali diisi. Isi tersebut menentukan tipe data dari variabel PHP. Untuk membuat variabel awali dengan tanda $ diikuti nama variabel. Contoh $nilai. Hal yang tidak diperbolehkan dalam menamai variabel.

1. Menggunakan spasi.
   $nilai akhir=90;
2. Menggunakan angka diawal
   $2nama=”Zen”;

Dalam membuat nama variabel boleh menggunakan *underscore.*
$nilai_akhir=90;
Nama variabel besifat *case sensitive* artinya *a* berbeda dengan *A.* Berikut ini adalah contoh program variabel dengan PHP:

```
<?php
$nilai_akhir=90;
echo "Nilai = $nilai_akhir";
?>
```

Setelah program dijalankan, maka hasil atau tampilan halaman pada *browser* seperti terlihat pada Gambar 3.1:



Gambar 3. 1. Variabel

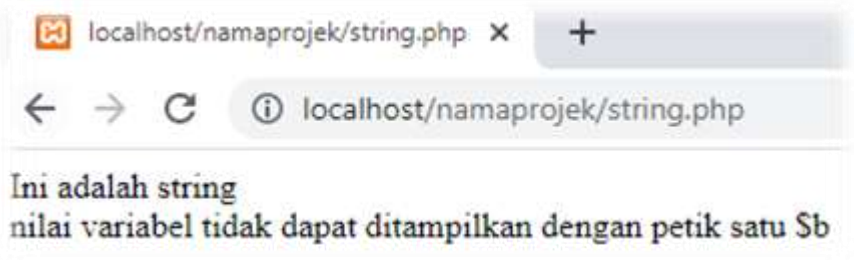
Pada pembahasan PHP selanjutnya tidak disertakan HTML agar lebih fokus. Namun umumnya untuk keperluan tampilan yang baik HTML diperlukan.

3.1 TIPE DATA *STRING*

Tipe data *string* adalah tipe data gabungan beberapa karakter bisa huruf, angka atau karakter lainnya. Berikut ini cara memasukkan data berjenis *string*.

```
<?php
$a="string";
$b='Zen';
echo "Ini adalah $a";
echo "<br>";
echo 'nilai variabel tidak dapat ditampilkan dengan petik satu $b';
?>
```

Pada kode program di atas tampak bagaimana variabel tidak perlu di deklarasikan. Variabel *a* langsung diisi "*string*". Sedangkan variabel *b* langsung diisi nilainya yaitu "zen". Perintah *echo* menggunakan kutip dua dapat mencetak teks dan nilai dari variabel *a*. Selanjutnya perintah *echo* juga dapat mencetak *tag* HTML yaitu "
". Namun jika menggunakan kutip tunggal maka nilai dari variabel tidak akan tampil melainkan variabel itu sendiri. Hasil dari kode program di atas dapat dilihat pada Gambar 3.2.



Gambar 3. 2. Tipe Data *String*

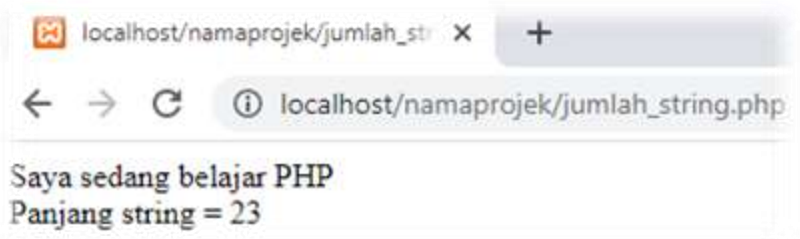
Untuk memasukkan nilai *string* dapat menggunakan petik satu atau petik dua, begitu juga menampilkannya. Namun terdapat perbedaan fungsi dalam menampilkan string menggunakan petik satu dan petik dua. Petik dua dapat menampilkan isi dalam variabel, sedangkan petik satu akan menampilkan apa adanya sesuai dengan yang tertulis, seperti terlihat pada Gambar 3.2. Ada banyak operasi *string* yang dapat memudahkan pemrograman.

3.1.1 Jumlah *String*

Untuk menghitung jumlah *string* dalam variabel fungsi yang digunakan adalah strlen(). Berikut ini contoh programnya.

```
<?php
$a="Saya sedang belajar PHP";
$b=strlen($a);
echo "$a <br>";
echo "Panjang string = $b";
?>
```

Pada kode program di atas tampak fungsi *strlen* atau *string length*. Fungsi tersebut digunakan untuk menghitung jumlah *string* tidak maupun dalam sebuah variabel. Kembalian nilai fungsi bisa langsung ditampilkan atau ditapung lebih dulu. Pada contoh program di atas jumlah *string* ditampung lebih dulu dalam variabel *b*. Hasil dari kode program di atas dapat dilihat pada Gambar 3.3.



Gambar 3. 3. Jumlah *String*

Pada tampilan Gambar 3.3, tampak tulisan Saya sedang belajar PHP. Setelah dihitung menggunakan fungsi strlen() *string* tersebut berjumlah 23. Baik huruf, angka, simbol maupun spasi akan dihitung.

3.1.2 Membalik *String*

Untuk membalik *string* dalam variabel fungsi yang digunakan adalah strrev(). Berikut ini contoh programnya.

```
<?php
$a="Saya sedang belajar PHP";
$b=strrev($a);
echo "$a <br>";
echo "Membalik string = $b";
?>
```

Pada kode program di atas fungsi yang digunakan untuk membalikkan *string* adalah strrev(). *String* yang akan dibalik diletakkan dalam buka tutup kurung sebagai parameter fungsi. Hasil dari membalik *string* tersebut disimpan dalam variabel $b dan kemudian dicetak. Berikut dapat dilihat tampilan hasil kode program di atas pada Gambar 3.4.

Gambar 3. 4. Membalik *String*

Pada Gambar 3.4 tampak hasil membalik *string* bertuliskan Saya sedang belajar PHP. Fungsi strrev() di atas berhasil membalik tulisan tersebut.

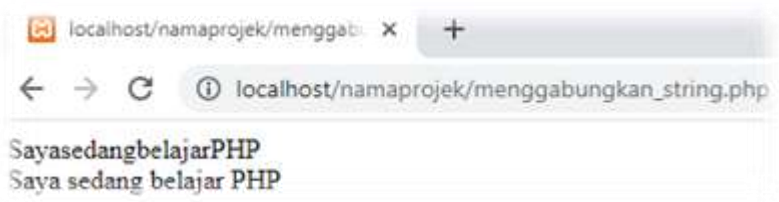
3.1.3 Menggabungkan *String*

Untuk menggabungkan *string* operator yang digunakan adalah ”.” (titik). Berikut ini contoh programnya.

```
<?php
$a="Saya";
$b="sedang";
$c="belajar";
$d="PHP";
echo $a.$b.$c.$d;
echo "<br>";
echo $a." ".$b." ".$c." ".$d;
?>
```

Pada kode program di atas tampak penggunaan titik “.” Sebagai penggabung beberapa *string*. Operator penggabungan ini banyak digunakan untuk menggabungkan label dengan nilai. Walaupun tanpa titik “.” cukup menggunakan petik dua (“) sudah cukup. Namun beberapa kondisi tidak dapat dilakukan. Misalnya terdapat operasi pada variabel.

Pada contoh program di atas operator penggabungan digunakan untuk menambahkan spasi antar *string* dalam variabel $a, $b, $c, dan $d. Penggunaan operator penggabungan ini wajib diketahui pemula, karena sering sekali penggunaannya. Hasil dari kode program di atas dapat dilihat pada Gambar 3.5.



Gambar 3. 5. Menggabungkan *String*

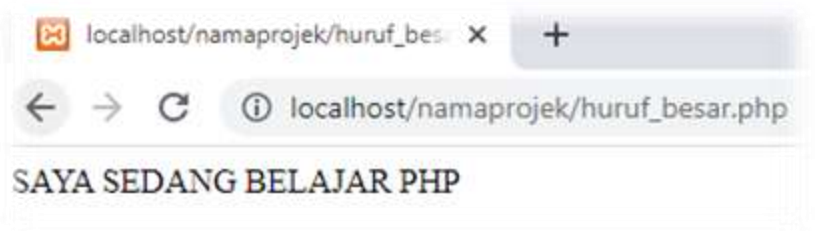
Pada Gambar 3.5 tampak hasil dari penggabungan *string* yang berasal dari variabel $a, $b, $c, dan $d. Pada baris pertama *string* tidak dipisahkan oleh spasi. Sedangkan pada baris kedua *string* dipisahkan oleh spasi menggunakan bantuan operator penggabungan.

3.1.4 Huruf Besar (*Uppercase*)

Fungsi untuk mengubah huruf menjadi huruf besar adalah strtoupper(), berikut ini contoh programnya.

```
<?php
$a="Saya sedang belajar PHP";
$b= strtoupper($a);
echo $b;
?>
```

Pada kode program di atas sebuah contoh kalimat yang disimpan dalam variabel $a. Ternyata kalimat tersebut harus dibuat huruf besar semua. Tanpa perlu repot menulis ulang. Tulisan tersebut dapat dibuah menjadi huruf besar menggunakan fungsi strtoupper(). Hasil dari kode program di atas dapat dilihat pada Gambar 3.6.



Gambar 3. 6. Huruf Besar

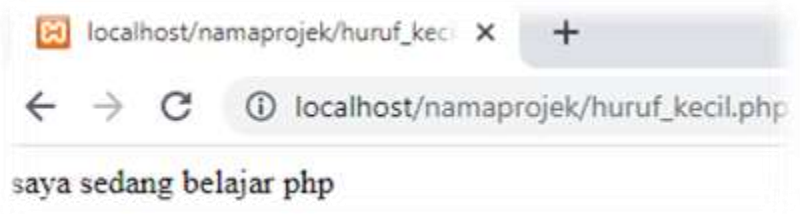
Pada Gambar 3.6 tampak semua huruf adalah huruf besar (kapital) semua. Fungsi yang digunakan untuk hal tersebut adalah strtoupper().

3.1.5 Huruf Kecil (*Lowercase*)

Fungi untuk mengubah huruf menjadi kecil adalah strtolower(), berikut ini contoh programnya.

```
<?php
$a="Saya sedang belajar PHP";
$b= strtolower($a);
echo $b;
?>
```

Pada contoh kode program di atas dapat dilihat kalimat Saya sedang belajar PHP ditampung dalam variabel *$a*. Kemudian nilai tersebut diubah menjadi tulisan kecil semua dengan perintah strtolower(). Hasil dari kode program di atas dapat dilihat pada Gambar 3.7.



Gambar 3. 7. Huruf Kecil

Pada Gambar 3.7 terlihat tulisan saya belajar PHP dengan huruf kecil semua. Dengan fungsi di atas maka dengan mudah mengubah teks menjadi seragam.

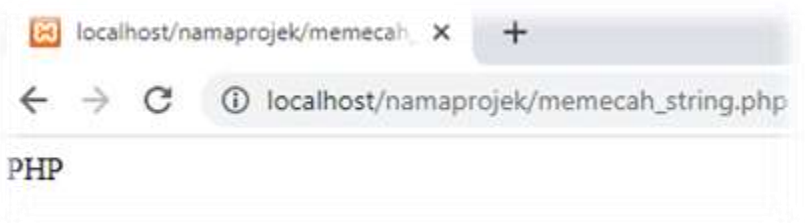
3.1.6 Memecah *String*

Untuk memecah *string* fungsi yang digunakan adalah explode() atau str_plit. Hasilnya berupa *array* dari *string* yang dipecah. Berikut ini contoh program dengan fungsi explode().

```
<?php
$a="Saya sedang belajar PHP";
$b= explode(" ", $a);
echo $b[3];
?>
```

Fungsi satu ini tidak kalah penting. Fungsi ini dibutuhkan *programmer* untuk hal-hal yang sangat penting. Misalnya mengartikan sebuah kode. Kode-kode tertentu perlu dipecahkan agar dapat diartikan.

Fungsi explode(" ", $a) di atas digunakan untuk memecah kata yang dipisahkan oleh spasi. Pada baris pertama sebuah kalimat ditampung ke dalam variabel *$a*. Kemudian fungsi explode akan memecahkan kalimat berdasarkan spasi. Terdapat tiga buah spasi yang ditemukan pada kalimat tersebut dan menghasilkan empat kata yaitu Saya, sedang, belajar dan PHP yang ditampung dalam variabel *$b*. Variabel *$b* di atas berjenis *array*. Setiap *index* nya menyimpan kata sebanyak empat kata. Seperti *array* biasa *index* dimulai dari nol. Sehingga jika dicetak isi dari variabel *$b* dengan index tiga. Maka akan tampil tulisan PHP. Hasil dari kode program di atas dapat dilihat pada Gambar 3.8.



Gambar 3. 8. Memecah *String* Berdasarkan Karakter

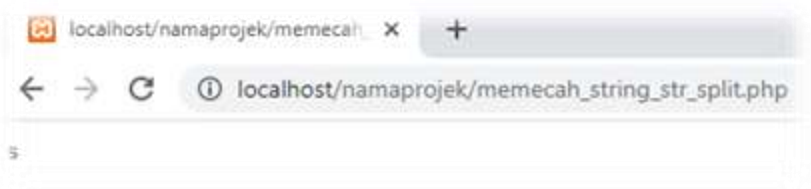
Pada Gambar 3.8, *string* dipecah berdasarkan spasi. Pecahan pertama $b[0] adalah Saya, $b[1] adalah sedang, $b[2] adalah belajar, $b[3] adalah PHP.

Berbeda dengan fungsi explode() yang berfungsi memecah string menggunakan acuan nilai seperti spasi. Fungsi berikut ini akan memecah

string satu persatu. Fungsi tersebut adalah str_split(). Berikut ini contoh program dengan fungsi str_split().

```
<?php
$a="Saya sedang belajar PHP";
$b= str_split($a);
echo $b[5];
?>
```

Pada kode program di atas variabel $b akan menjadi *array* yang menyimpan setiap karakter. Kemudian perintah echo mencetak nilai dari *array* $b dengan indeks ke-5. Hasil dari kode program di atas dapat dilihat pada Gambar 3.9.



Gambar 3. 9. Memecah *String* Satu Persatu

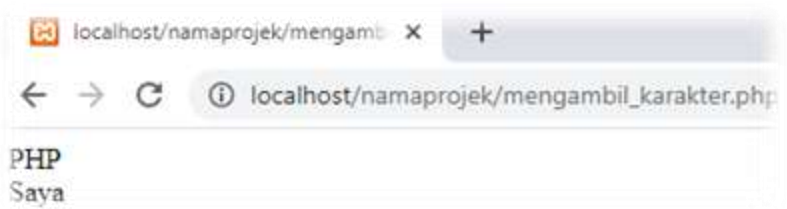
Pada Gambar 3.9 terlihat huruf *s* yang merupakan indeks ke-5 dari *array*. Sedangkan urutan huruf tersebut adalah urutan yang ke-6 dari kalimat : Saya sedang belajar PHP.

3.1.7 Mengambil Beberapa Karakter Dari Kanan atau Kiri

Untuk mengambil beberapa karakter dari kanan atau kiri fungsi yang digunakan adalah substr(), berikut ini contoh programnya.

```
<?php
$a="Saya sedang belajar PHP";
//dari kanan
echo substr($a,-3);
echo "<br>";
//dari kiri
echo substr($a,0,4);
?>
```

Pada kode program di atas, fungsi yang digunakan untuk mengambil beberapa karakter adalah *substr*. Hasil dari kode program di atas dapat dilihat pada Gambar 3.10.



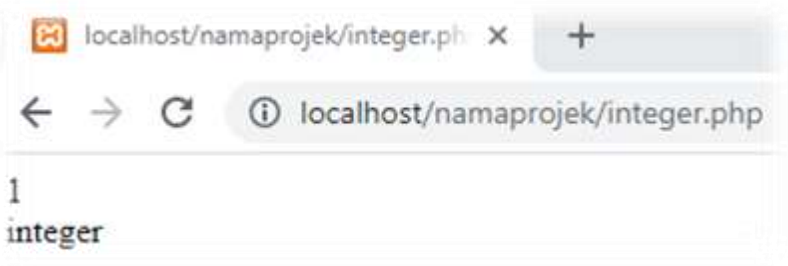
Gambar 3. 10. Mengambil Beberapa Karakter

3.2 TIPE DATA *INTEGER*

Tipe data *integer* digunakan untuk menampung bilangan bulat. Berikut ini contoh programnya.

```
<?php
$a=1;
echo $a;
echo "<br>";
echo gettype($a);
?>
```

Fungsi yang digunakan untuk mengetahui tipe data variabel adalah gettype(). Hasil dari kode program di atas dapat dilihat pada Gambar 3.11.



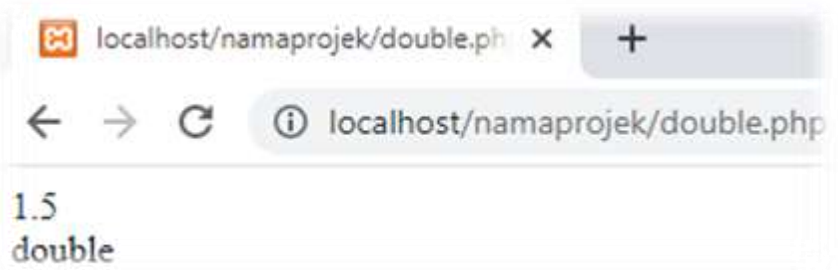
Gambar 3. 11. Tipe Data *Integer*

3.3 TIPE DATA *DOUBLE*

Tipe data *double* digunakan untuk menampung bilangan pecahan. Berikut ini contoh programnya.

```
<?php
$a=1.5;
echo $a;
echo "<br>";
echo gettype($a);
?>
```

Fungsi yang digunakan untuk mengetahui tipe data variabel adalah gettype(). Hasil dari kode program di atas dapat dilihat pada Gambar 3.12.



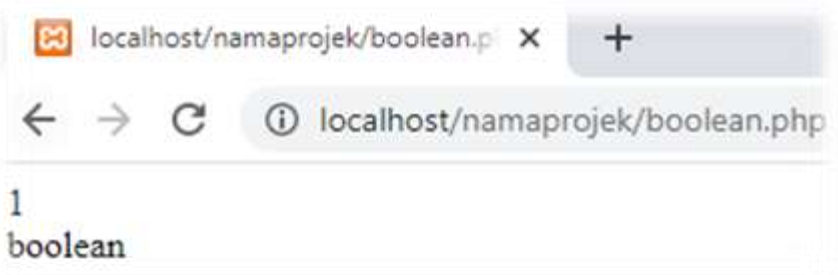
Gambar 3. 12. Tipe Data *Double*

3.4 TIPE DATA *BOOLEAN*

Tipe data *boolean* digunakan untuk menampung nilai *true* atau *false*. Berikut ini contoh programnya.

```
<?php
$a=True;
echo $a;
echo "<br>";
echo gettype($a);
?>
```

Fungsi yang digunakan untuk mengetahui tipe data variabel adalah gettype(). Hasil dari kode program di atas dapat dilihat pada Gambar 3.13.



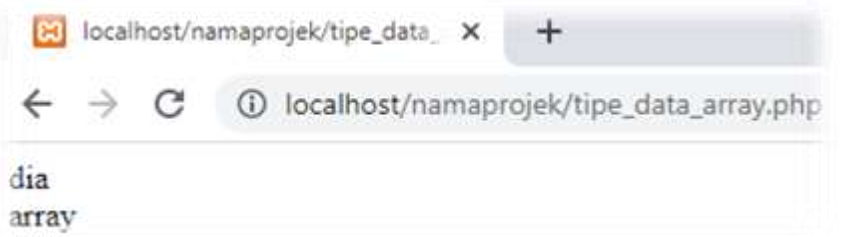
Gambar 3. 13. Tipe Data *Boolean*

3.5 TIPE DATA *ARRAY*

Tipe data *array* adalah variabel yang mampu menampung lebih dari satu data dengan *index*. Berikut ini contoh programnya.

```
<?php
$a=["saya","kamu","dia"];
echo $a[2];
echo "<br>";
echo gettype($a);
?>
```

Fungsi yang digunakan untuk mengetahui tipe data variabel adalah gettype(). Hasil dari kode program di atas dapat dilihat pada Gambar 3.14.



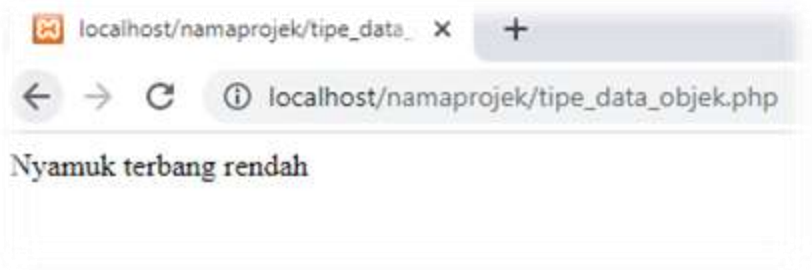
Gambar 3. 14. Tipe Data *Array*

3.6 TIPE DATA OBJEK

Tipe data objek dari sebuah *class* adalah berupa variabel (diawali $) dengan tipe data objek. Berikut ini contoh programnya.

```
<?php
class Nyamuk{
        public function terbang(){
                echo "Nyamuk terbang rendah";
        }
}

$obj_nyamuk= new Nyamuk;
$obj_nyamuk->terbang();
?>
```

Pada kode di atas, variabel objeknya adalah $obj_nyamuk. Fungsi yang digunakan untuk mengetahui tipe data variabel adalah gettype(). Hasil dari kode program di atas dapat dilihat pada Gambar 3.15.



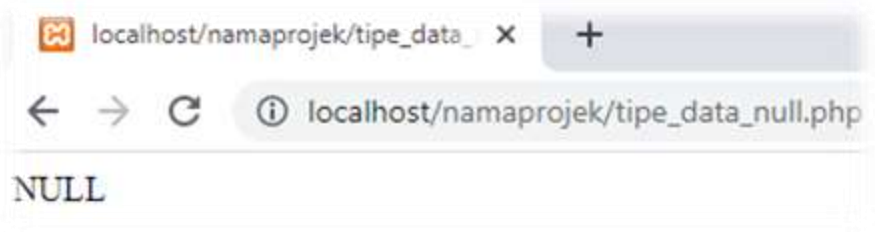
Gambar 3. 15. Tipe Data Obyek

3.7 TIPE DATA *NULL*

Tipe data *null* untuk menunjukkan sebuah variabel belum memiliki isi data. Berikut ini contoh programnya.

```
<?php
$a=NULL;
echo $a;
echo gettype($a);
?>
```

Fungsi yang digunakan untuk mengetahui tipe data variabel adalah gettype(). Hasil dari kode program di atas dapat dilihat pada Gambar 3.16.



Gambar 3. 16. Tipe Data *Null*

3.8 TIPE DATA *RESOURCE*

Tipe data *resource* dalah tipe data yang menampung data sumber. Contohnya ketika membuat koneksi ke *databese* maka koneksi tersebut ditampung ke dalam sebuah variabel. Maka variabel tersebut memiliki tipe data *resource*.

3.9 FUNGSI UNTUK PENGECEKAN VARIABEL

Ada beberapa fungsi pengecekan variabel yang penting untuk diketahui. Sering kali sebuah alur program perlu mengecek sebuah variabel, misalkan apakah isinya kosong (*Null*) atau bahkan ada tidaknya variabel yang dimaksud sampai tipe data variabel tersebut.

Tabel 3. 1. Fungsi Untuk Pengecekan Variabel

No.	Fungsi	Keterangan
1.	isset( )	Apakah variabel diset atau ada?
2.	empty( )	Apakah variabel kosong?
3.	!isset( )	Apakah variabel tidak diset atau tidak ada?
4.	!empty( )	Apakah variabel tidak kosong?

Operator merupakan bagian penting dari pemrograman, mengingat kerja utama dari komputer adalah menghitung. Ada beberapa jenis operator diantaranya adalah:

1. Aritmatika
2. Penugasan
3. *Bitwise*
4. Perbandingan
5. Logika
6. Penggabungan *String*
7. Increment dan *Decrement*

4.1 OPERATOR ARITMATIKA

Operator aritmatika adalah operator matematika yang sangat sering kita gunakan. Mulai dari perhitungan sederhana hingga rumit. Berikut ini operator-operator aritmatika.

Tabel 4. 1. Operator Aritmatika

No.	Operator	Keterangan
1.	+	Penjumlahan
2.	-	Pengurangan
3.	*	Perkalian
4.	/	Bagi
5.	%	Modulus

Operasi dapat dilakukan terhadap nilai ataupun isi dari variabel. Berikut ini contoh program operasi aritmatika.

1. Penjumlahan

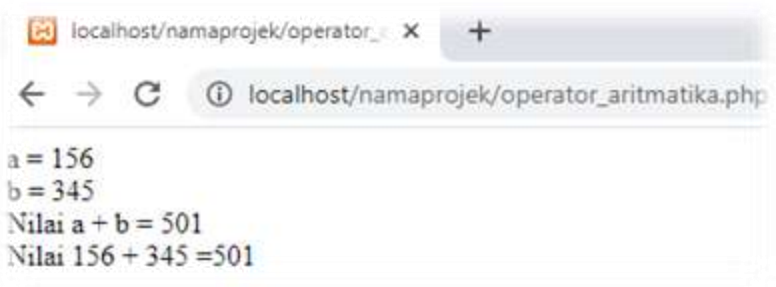
Operator penjumlahan adalah operator dasar yang sering kita gunakan. *Processor* di dalam komputer memproses penjumlahan dalam bentuk

bilangan biner. Kemampuan komputer sebagai mesin penghitung di dalam *Arithmetic and Logic Unit* membuat banyak algoritma program yang tak luput dari perhitungan-perhitungan. Berikut ini contoh program operasi penjumlahan.

```
<?php
$a=156;
$b=345;
$hasil=$a+$b;
$hasil1=156+345;
echo "a = $a <br>";
echo "b = $b <br>";
echo " Nilai a + b = $hasil";
echo "<br>";
echo " Nilai $a + $b =". $hasil1;
?>
```

Pada contoh kode program di atas adalah cara menjumlahkan dua buah nilai (*operand*) menggunakan operator tambah. Kita dapat menjumlahkan dua buah variabel. Bahkan menjumlahkan dua buah bilangan secara langsung.

Pada umumnya dalam menuliskan kode program, hasil dari sebuah proses di tampung ke dalam variable. Kode program di atas hasil penjumlahan di tampung pada variabel $hasil. Setiap yang menampung hasil berada di sebelah kiri tanda sama dengan. Bagi pemula mungkin sedikit sulit memahami hal tersebut. Hasil dari kode program di atas dapat dilihat pada Gambar 4.2.



Gambar 4. 1. Penjumlahan

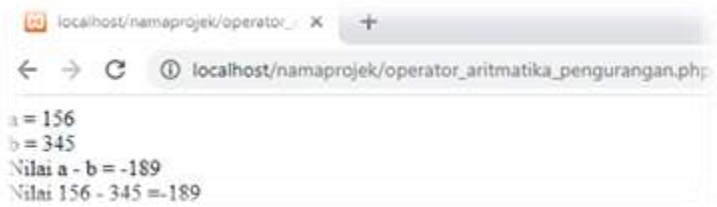
2. Pengurangan

Operator pengurangan adalah operator dasar yang sering kita gunakan setelah operator penjumlahan. *Processor* di dalam komputer memproses pengurangan dalam bentuk bilangan biner. Kemampuan komputer sebagai mesin penghitung di dalam *Arithmetic and Logic Unit* membuat banyak algoritma program yang tak luput dari perhitungan-perhitungan. Berikut ini contoh program operasi pengurangan.

```
<?php
$a=156;
$b=345;
$hasil=$a-$b;
$hasil1=156-345;
echo "a = $a <br>";
echo "b = $b <br>";
echo " Nilai a - b = $hasil";
echo "<br>";
echo " Nilai $a - $b =". $hasil1;
?>
```

Pada contoh kode program di atas adalah cara mengurangkan dua buah nilai (*operand*) menggunakan operator kurang. Kita dapat mengurangkan dua buah variabel. Bahkan mengurangkan dua buah bilangan secara langsung.

Pada umumnya dalam menuliskan kode program, hasil dari sebuah proses di tampung ke dalam variable. Contohnya pada kode program di atas hasil pengurangan di tampung pada variabel *$hasil*. Setiap yang menampung hasil berada di sebelah kiri tanda sama dengan "=". Bagi pemula mungkin sedikit sulit memahami hal tersebut. Hasil dari kode program di atas dapat dilihat pada Gambar 4.3.



Gambar 4. 2. Pengurangan

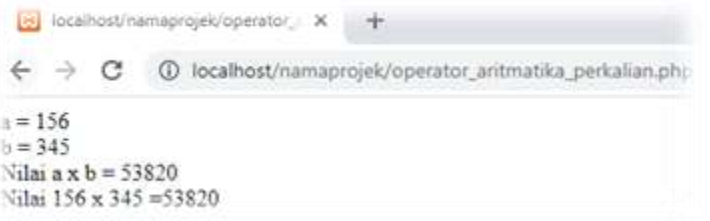
3. Perkalian

Operator perkalian adalah operator aritmatika dasar yang sering kita gunakan setelah operator lainnya. *Processor* di dalam komputer memproses perkalian dalam bentuk bilangan biner. Kemampuan komputer sebagai mesin penghitung di dalam *Arithmetic and Logic Unit* membuat banyak algoritma program yang tak luput dari perhitungan-perhitungan. Berikut ini contoh program operasi perkalian.

```
<?php
$a=156;
$b=345;
$hasil=$a*$b;
$hasil1=156*345;
echo "a = $a <br>";
echo "b = $b <br>";
echo " Nilai a x b = $hasil";
echo "<br>";
echo " Nilai $a x $b =". $hasil1;
?>
```

Pada contoh kode program di atas adalah cara mengalikan dua buah nilai (*operand*) menggunakan operator perkalian. Kita dapat mengalikan dua buah variabel. Bahkan mengalikan dua buah bilangan secara langsung.

Pada umumnya dalam menuliskan kode program, hasil dari sebuah proses di tampung ke dalam variable. Contohnya pada kode program di atas hasil perkalian di tampung pada variabel $hasil. Setiap yang menampung hasil berada di sebelah kiri tanda sama dengan. Bagi pemula mungkin sedikit sulit memahami hal tersebut. Hasil dari kode program di atas dapat dilihat pada Gambar 4.4.



Gambar 4. 3. Perkalian

4. Pembagian

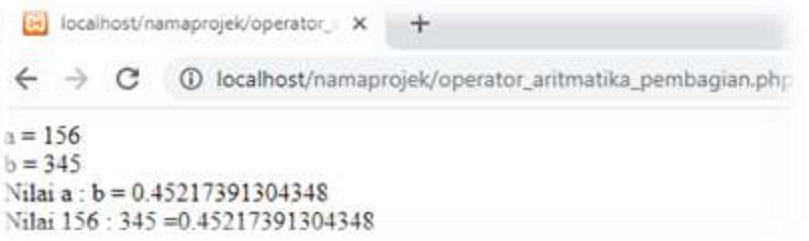
Operator pembagian adalah operator aritmatika dasar yang sering kita gunakan setelah operator lainnya. *Processor* di dalam komputer memproses pembagian dalam bentuk bilangan biner. Kemampuan komputer sebagai mesin penghitung di dalam *Arithmetic and Logic Unit* membuat banyak algoritma program yang tak luput dari perhitungan-perhitungan. Berikut ini contoh program operasi perkalian.

```
<?php
$a=156;
$b=345;
$hasil=$a/$b;
$hasil1=156/345;
echo "a = $a <br>";
echo "b = $b <br>";
echo " Nilai a : b = $hasil";
echo "<br>";
echo " Nilai $a : $b =". $hasil1;
?>
```

Pada contoh kode program di atas adalah cara membagikan dua buah nilai (*operand*) menggunakan operator pembagian. Kita dapat membagikan dua buah variabel. Bahkan membagikan dua buah bilangan secara langsung.

Pada umumnya dalam menuliskan kode program, hasil dari sebuah proses di tampung ke dalam variable. Contohnya pada kode program di atas hasil pembagian di tampung pada variabel $hasil. Setiap yang menampung hasil berada di sebelah kiri tanda sama dengan. Bagi pemula mungkin sedikit

sulit memahami hal tersebut. Hasil dari kode program di atas dapat dilihat pada Gambar 4.5.



Gambar 4. 4. Pembagian

5. Modulus

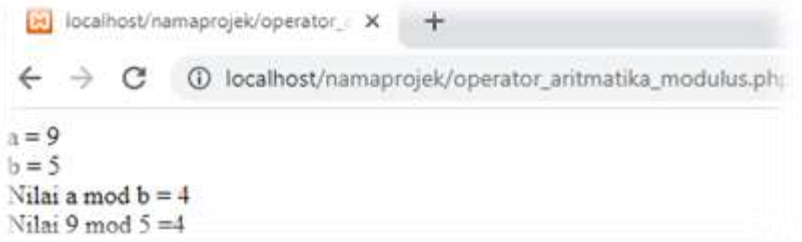
Operator modulus adalah operator aritmatika dasar yang sering kita gunakan setelah operator lainnya. *Processor* di dalam komputer memproses operasi modulus dalam bentuk bilangan biner. Kemampuan komputer sebagai mesin penghitung di dalam *Arithmetic and Logic Unit* membuat banyak algoritma program yang tak luput dari perhitungan-perhitungan. Berikut ini contoh program operasi modulus.

```
<?php
$a=9;
$b=5;
$hasil=$a % $b;
$hasil1=9 % 5;
echo "a = $a <br>";
echo "b = $b <br>";
echo " Nilai a modulus b = $hasil";
echo "<br>";
echo " Nilai $a modulus $b =". $hasil1;
?>
```

Pada contoh kode program di atas adalah cara memoduluskan dua buah nilai (*operand*) menggunakan operator modulus. Kita dapat memoduluskan dua buah variabel. Bahkan memoduluskan dua buah bilangan secara langsung.

Pada umumnya dalam menuliskan kode program, hasil dari sebuah proses di tampung ke dalam variable. Contohnya pada kode program di atas

hasil modulus di tampung pada variabel $hasil. Setiap yang menampung hasil berada di sebelah kiri tanda sama dengan. Bagi pemula mungkin sedikit sulit memahami hal tersebut. Hasil dari kode program di atas dapat dilihat pada Gambar 4.6.



Gambar 4. 5. Modulus

4.2 OPERATOR PENUGASAN

Operator penugasan berfungsi untuk memasukkan suatu nilai ke dalam variabel. Operator penugasan dapat dibedakan menjadi dua jenis. Operator penugasan yang pertama berfungsi menugaskan untuk menampung nilai dengan menimpah apa pun yang ada di dalam variabel tersebut. Sedangkan operator penugasan yang kedua berfungsi menugaskan untuk menampung nilai dengan mempertimbangkan nilai sebelumnya. Berikut ini operator-operator penugasan.

Tabel 4. 2. Operator Penugasan

No.	Operator	Keterangan
1.	=	Memberi nilai
2.	+=	Memberi nilai dan menjumlahkan nilai sebelumnya
3.	-=	Memberi nilai dan mengurangkan nilai sebelumnya
4.	.=	Memberi nilai dan menggabungkan nilai sebelumnya
5.	*=	Memberi nilai dan mengalikan nilai sebelumnya
6.	/=	Memberi nilai dan membagikan nilai sebelumnya
7.	%=	Memberi nilai dan memoduluskan nilai sebelumnya

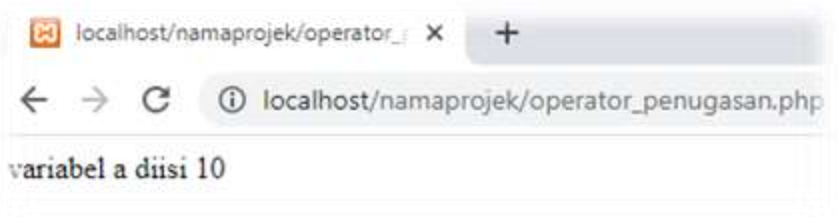
1. Sama Dengan

Operator ini berfungsi untuk memberi nilai ke dalam sebuah variabel. Aturan memasukkan nilai ke dalam variabel yaitu variabel di sebelah kiri

sama dengan, sedangkan nilai yang akan dimasukkan berada di sebelah kanan sama dengan. Berikut ini contoh programnya.

```
<?php
$a=10;
echo "variabel a diisi $a";
?>
```

Pada contoh program di atas adalah contoh kasus operator penugasan sama dengan untuk memasukkan nilai ke dalam variabel. Variabel $a diletakkan disebelah kiri tanda sama dengan. Sedangkan nilai yang akan dimasukkan berada di sebelah kanan tanda sama dengan. Hasil dari kode program di atas dapat dilihat pada Gambar 4.7.



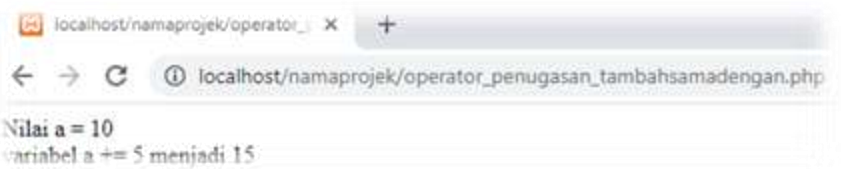
Gambar 4. 6. Sama Dengan

2. Tambah Sama Dengan

Operator ini berfungsi untuk memberi nilai ke dalam sebuah variabel dengan menjumlahkan nilai sebelumnya. Berikut ini contoh programnya.

```
<?php
$a=10;
echo "Nilai a = $a <br>";
$a+=5;
echo "variabel a += 5 menjadi $a";
?>
```

Pada contoh program di atas adalah contoh kasus operator penugasan tambah sama dengan. Jika nilai sebelumnya ada, maka nilai tersebut tidak ditimpah melainkan dijumlahkan dengan nilai yang baru. Sehingga hasil penjumlahan nilai yang lama dengan nilai yang baru adalah lima belas. Hasil dari kode program di atas dapat dilihat pada Gambar 4.8.



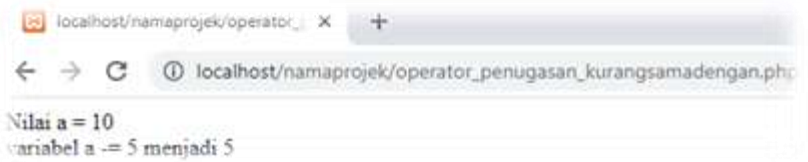
Gambar 4. 7. Tambah Sama Dengan

3. Kurang Sama Dengan

Operator ini berfungsi untuk memberi nilai ke dalam sebuah variabel dengan mengurangkan nilai sebelumnya. Berikut ini contoh programnya.

```
<?php
$a=10;
echo "Nilai a = $a <br>";
$a-=5;
echo "variabel a -= 5 menjadi $a";
?>
```

Pada contoh program di atas adalah contoh kasus operator penugasan kurang sama dengan. Jika nilai sebelumnya ada, maka nilai tersebut tidak ditimpah melainkan dikurangkan dengan nilai yang baru. Sehingga hasil pengurangan nilai yang lama dengan nilai yang baru adalah lima. Hasil dari kode program di atas dapat dilihat pada Gambar 4.9.



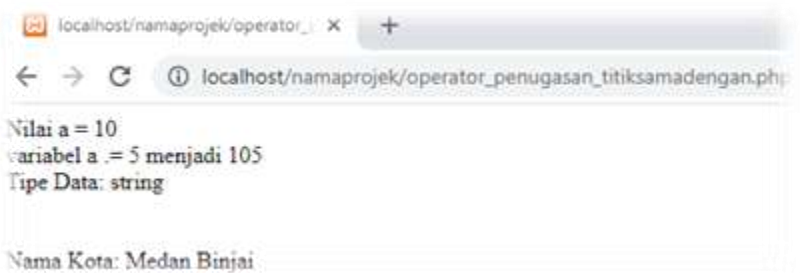
Gambar 4. 8. Kurang Sama Dengan

4. Titik Sama Dengan

Operator ini berfungsi untuk memberi nilai ke dalam sebuah variabel dengan menggabungkan nilai sebelumnya. Berikut ini contoh programnya.

```
<?php
$a=10;
echo "Nilai a = $a <br>";
$a.=5;
echo "variabel a .= 5 menjadi $a";
echo "<br>";
echo "Tipe Data: ".gettype($a);

echo "<br><br><br>";
$kota="Nama Kota: ";
$kota.="Medan ";
$kota.="Binjai ";
echo $kota;

?>
```

Pada contoh program di atas adalah contoh kasus operator penugasan titik sama dengan. Jika nilai sebelumnya ada, maka nilai tersebut tidak ditimpah melainkan digabungkan dengan nilai yang baru. Namun hasil dari operator ini dalah data dengan tipe *string*. Hasil dari kode program di atas dapat dilihat pada Gambar 4.10.



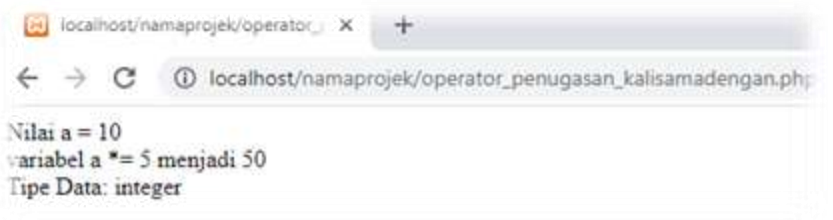
Gambar 4. 9. Titik Sama Dengan

5. Kali Sama Dengan

Operator ini berfungsi untuk memberi nilai ke dalam sebuah variabel dengan mengalikan nilai sebelumnya. Berikut ini contoh programnya.

```
<?php
$a=10;
echo "Nilai a = $a <br>";
$a*=5;
echo "variabel a *= 5 menjadi $a";
echo "<br>";
echo "Tipe Data: ".gettype($a);
?>
```

Pada contoh program di atas adalah contoh kasus operator penugasan kali sama dengan. Jika nilai sebelumnya ada, maka nilai tersebut tidak ditimpah melainkan dikalikan dengan nilai yang baru. Sehingga hasil perkalian nilai yang lama dengan nilai yang baru adalah lima puluh. Hasil dari kode program di atas dapat dilihat pada Gambar 4.11.



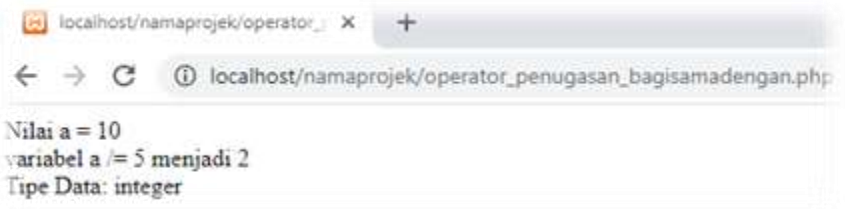
Gambar 4. 10. Kali Sama Dengan

6. Bagi Sama Dengan

Operator ini berfungsi untuk memberi nilai ke dalam sebuah variabel dengan membagikan nilai sebelumnya. Berikut ini contoh programnya.

```
<?php
$a=10;
echo "Nilai a = $a <br>";
$a/=5;
echo "variabel a /= 5 menjadi $a";
echo "<br>";
echo "Tipe Data: ".gettype($a);
?>
```

Pada contoh program di atas adalah contoh kasus operator penugasan bagi sama dengan. Jika nilai sebelumnya ada, maka nilai tersebut tidak ditimpah melainkan dibagikan dengan nilai yang baru. Sehingga hasil pembagian nilai yang lama dengan nilai yang baru adalah dua. Hasil dari kode program di atas dapat dilihat pada Gambar 4.12.



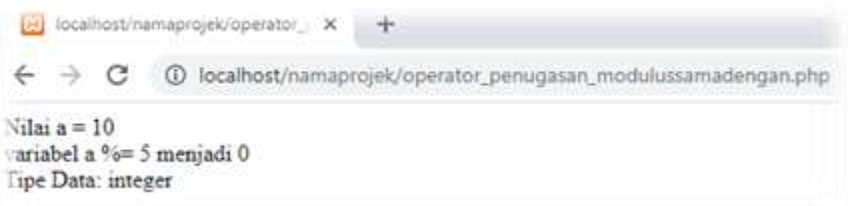
Gambar 4. 11. Bagi Sama Dengan

7. Modulus Sama Dengan

Operator ini berfungsi untuk memberi nilai ke dalam sebuah variabel dengan memoduluskan nilai sebelumnya. Berikut ini contoh programnya.

```
<?php
$a=10;
echo "Nilai a = $a <br>";
$a*=5;
echo "variabel a *= 5 menjadi $a";
echo "<br>";
echo "Tipe Data: ".gettype($a);
?>
```

Pada contoh program di atas adalah contoh kasus operator penugasan modulus sama dengan. Jika nilai sebelumnya ada, maka nilai tersebut tidak ditimpah melainkan dimoduluskan dengan nilai yang baru. Sehingga hasil modulus nilai yang lama dengan nilai yang baru adalah nol. Hasil dari kode program di atas dapat dilihat pada Gambar 4.13.



Gambar 4. 12. Modulus Sama Dengan

4.3 OPERATOR *BITWISE*

Operator *bitwise* adalah operator khusus untuk operasi pada level bilangan biner. Meskipun angka yang akan kita hitung adalah desimal, tapi komputer akan menghitung berdasarkan nilai binernya. Jika diperhatikan, maka hal tersebut membuktikan bahwa komputer menggunakan sistem bilangan biner pada perangkat kerasnya. Berikut ini operator-operator *Bitwise*.

Tabel 4. 3. Operator Bitwise

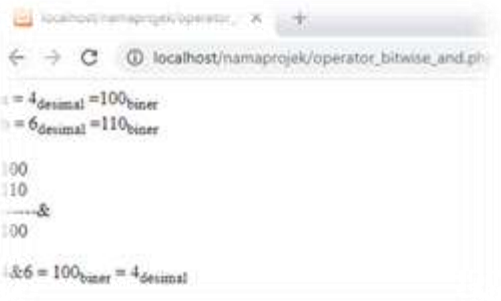
No.	Operator	Keterangan
1.	&	Bitwise AND
2.	|	Bitwise OR
3.	^	Bitwise XOR
4.	<<	Bitwise Shift Left (geser bit ke kiri)
5.	>>	Bitwise Shift Right (geser bit ke kanan)

1. *Bitwise* AND

Operasi AND berlaku untuk nilai biner dari sebuah *operand*. Jadi bilangan dalam bentuk desimal akan diubah dulu menjadi biner sebelum dioperasikan oleh operator AND. Sama seperti operator logika AND, hasil sama dengan 1 jika dan hanya jika semua operandnya 1. Berikut ini contoh programnya.

```
<?php
$a=4;
$b=6;
echo " a = $a<sub>desimal</sub> =".decbin($a)."<sub>biner</sub> <br>";
echo " b = $b<sub>desimal</sub> =".decbin($b)."<sub>biner</sub>";
echo "<br><br>";
echo decbin($a);
echo "<br>";
echo decbin($b);
echo "<br>";
echo "------";
echo "&";
echo "<br>";
$c=$a&$b;
echo decbin($c);
echo "<br><br>";
echo "$a&$b = ".decbin($c). "<sub>biner</sub> = $c<sub>desimal</sub>";

?>
```

Pada contoh program di atas penulis mencoba mendeskripsikan proses operasi menggunakan operator AND. Nilai variabel $a adalah empat desimal. Kemudian penulis memperlihatkan hasil konversi binernya yaitu 100_2. Sedangkan nilai variabel $b adalah enam desimal. Kemudian penulis memperlihatkan hasil konversi binernya yaitu 110_2. Terakhir penulis memperlihatkan proses operasi AND. Hasil operasi AND terhadap 6 dan 4 adalah 4. Hasil dari kode program di atas dapat dilihat pada Gambar 4.14.



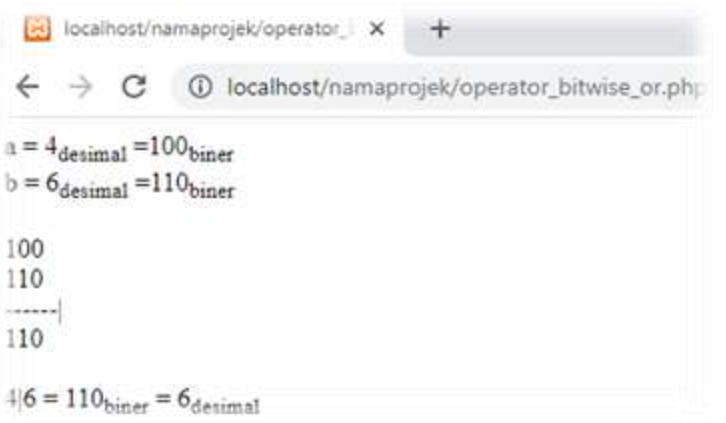
Gambar 4. 13. Operator *Bitwise* AND

2. *Bitwise* OR

Operasi OR berlaku untuk nilai biner dari sebuah *operand*. Jadi bilangan dalam bentuk desimal akan diubah dulu menjadi biner sebelum dioperasikan oleh operator AND. Sama seperti operator logika OR, hasil sama dengan 1 jika salah satu *operand*-nya 1. Berikut ini contoh programnya.

```
<?php
$a=4;
$b=6;
echo " a = $a<sub>desimal</sub> =".decbin($a)."<sub>biner</sub> <br>";
echo " b = $b<sub>desimal</sub> =".decbin($b)."<sub>biner</sub>";
echo "<br><br>";
echo decbin($a);
echo "<br>";
echo decbin($b);
echo "<br>";
echo "------";
echo "|";
echo "<br>";
$c=$a|$b;
echo decbin($c);
echo "<br><br>";
echo "$a|$b = ".decbin($c). "<sub>biner</sub> = $c<sub>desimal</sub>";

?>
```

Pada contoh program di atas penulis mencoba mendeskripsikan proses operasi menggunakan operator OR. Nilai variabel $a adalah empat desimal. Kemudian penulis memperlihatkan hasil konversi binernya yaitu 100_2. Sedangkan nilai variabel $b adalah enam desimal. Kemudian penulis memperlihatkan hasil konversi binernya yaitu 110_2. Terakhir penulis memperlihatkan proses operasi OR. Hasil operasi OR terhadap 6 dan 4 adalah 6. Hasil dari kode program di atas dapat dilihat pada Gambar 4.15.



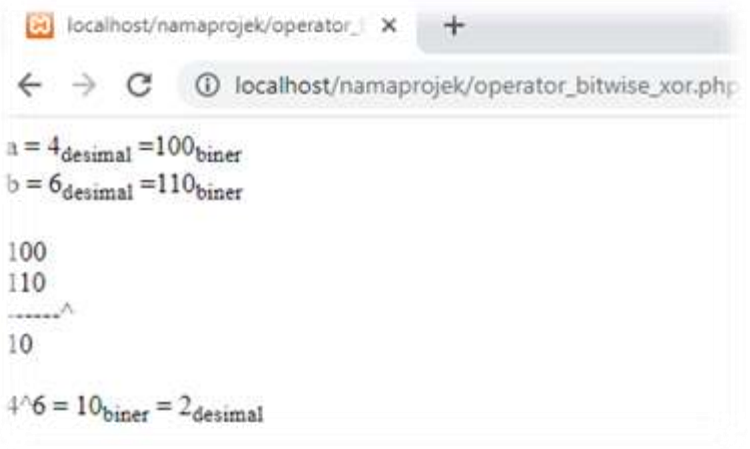
Gambar 4. 14. Operator *Bitwise* OR

3. Operator *Bitwise* XOR

Operasi XOR berlaku untuk nilai biner dari sebuah *operand*. Jadi bilangan dalam bentuk desimal akan diubah dulu menjadi biner sebelum dioperasikan oleh operator XOR. Sama seperti operator logika XOR, hasil sama dengan 1 jika dan hanya jika nilai operandnya berbeda. Berikut ini contoh programnya.

```
<?php
$a=4;
$b=6;
echo " a = $a<sub>desimal</sub> =".decbin($a)."<sub>biner</sub> <br>";
echo " b = $b<sub>desimal</sub> =".decbin($b)."<sub>biner</sub>";
echo "<br><br>";
echo decbin($a);
echo "<br>";
echo decbin($b);
echo "<br>";
echo "------";
echo "^";
echo "<br>";
$c=$a^$b;
echo decbin($c);
echo "<br><br>";
echo "$a^$b = ".decbin($c). "<sub>biner</sub> = $c<sub>desimal</sub>";

?>
```

Pada contoh program di atas penulis mencoba mendeskripsikan proses operasi menggunakan operator XOR. Nilai variabel $a adalah empat desimal. Kemudian penulis memperlihatkan hasil konversi binernya yaitu 100_2. Sedangkan nilai variabel $b adalah enam desimal. Kemudian penulis memperlihatkan hasil konversi binernya yaitu 110_2. Terakhir penulis memperlihatkan proses operasi XOR. Hasil operasi XOR terhadap 6 dan 4 adalah 2. Hasil dari kode program di atas dapat dilihat pada Gambar 4.16.



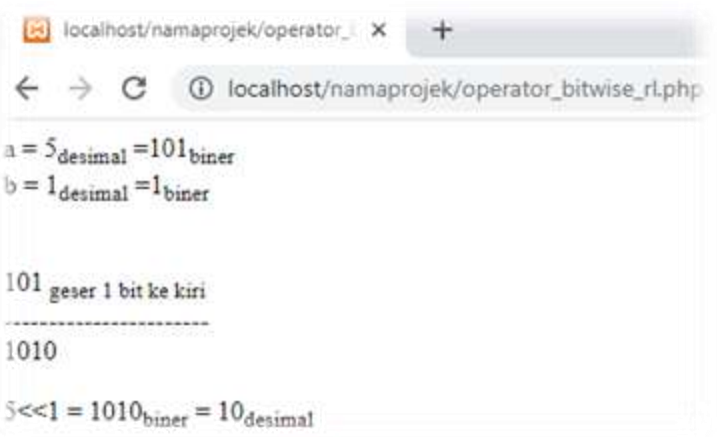
Gambar 4. 15. Operator *Bitwise* XOR

4. Operator *Bitwise Shift Left*

Operasi *Shift Lef*t berlaku untuk nilai biner dari sebuah *operand*. Jadi bilangan dalam bentuk desimal akan diubah dulu menjadi biner sebelum dioperasikan oleh operator *Shift Lef*t. Operator ini berfungsi untuk menggeser bit ke kiri sebanyak faktor penggesernya. Berikut contoh programnya.

```
<?php
$a=5;
$b=1;
echo " a = $a<sub>desimal</sub> =".decbin($a)."<sub>biner</sub> <br>";
echo " b = $b<sub>desimal</sub> =".decbin($b)."<sub>biner</sub> <br>";
echo "<br><br>";
echo decbin($a)."<sub> geser 1 bit ke kiri</sub>";
echo "<br> ----------------------";
echo "<br>";
$c=$a<<$b;
echo decbin($c);
echo "<br><br>";
echo "$a<<$b = ".decbin($c). "<sub>biner</sub> =
$c<sub>desimal</sub>";

?>
```

Pada contoh program di atas penulis mencoba mendeskripsikan proses operasi menggunakan operator *Shift Lef*t. Nilai variabel $a adalah lima desimal. Kemudian penulis memperlihatkan hasil konversi binernya yaitu 101 $_2$. Sedangkan nilai variabel $b adalah 1 desimal. Kemudian penulis memperlihatkan hasil konversi binernya yaitu 1 $_2$. Terakhir penulis memperlihatkan proses operasi *Shift Lef*t. Hasil operasi *Shift Lef*t terhadap 5 dan 1 adalah 10. Hasil dari kode program di atas dapat dilihat pada Gambar 4.17.



Gambar 4. 16. Operator *Bitwise* Shift Left

5. Operator *Bitwise Shift Right*

Operasi *Shift Right* berlaku untuk nilai biner dari sebuah *operand*. Jadi, bilangan dalam bentuk desimal akan diubah dulu menjadi biner sebelum dioperasikan oleh operator *Shift Right*. Operator ini berfungsi untuk menggeser bit ke kanan sebanyak faktor penggesernya. Berikut contoh programnya.

```
<?php
$a=5;
$b=2;
echo " a = $a<sub>desimal</sub> =".decbin($a)."<sub>biner</sub> <br>";
echo " b = $b<sub>desimal</sub> =".decbin($b)."<sub>biner</sub> <br>";
echo "<br><br>";
echo decbin($a)."<sub> geser 2 bit ke kanan</sub>";
echo "<br> ----------------------";
echo "<br>";
$c=$a>>$b;
echo decbin($c);
echo "<br><br>";
echo "$a>>$b = ".decbin($c). "<sub>biner</sub> =
$c<sub>desimal</sub>";

?>
```

Pada contoh program di atas penulis mencoba mendeskripsikan proses operasi menggunakan operator *Shift Right*. Nilai variabel $a adalah lima desimal. Kemudian penulis memperlihatkan hasil konversi binernya yaitu 101_2. Sedangkan nilai variabel $b adalah 2 desimal. Kemudian penulis memperlihatkan hasil konversi binernya yaitu 10_2. Terakhir penulis memperlihatkan proses operasi *Shift Rigt*. Hasil operasi *Shift Rigt* terhadap 5 dan 2 adalah 1. Hasil dari kode program di atas dapat dilihat pada Gambar 4.18.

Gambar 4. 17. Operator *Bitwise Shift Right*

4.4 Operator Perbandingan

Sesuai dengan namanya, operator perbandingan untuk membandingkan nilai. Hasil dari operator perbandingan adalah nilai benar (1) atau salah (0). Sehingga operator ini banyak digunakan pada kasus kondisi. Berikut ini adalah operator-operator tersebut.

Tabel 4. 4. Operator Perbandingan

No.	Operator	Keterangan	Output
1	==	Apakah sama dengan	Benar atau Salah
2	!=	Apakah tidak sama dengan	
3	<>	Apakah tidak sama dengan	
4	<	Apakah lebih kecil	
5	>	Apakah lebih besar	
6	<=	Apakah lebih kecil sama dengan	
7	>=	Apakah lebih besar sama dengan	

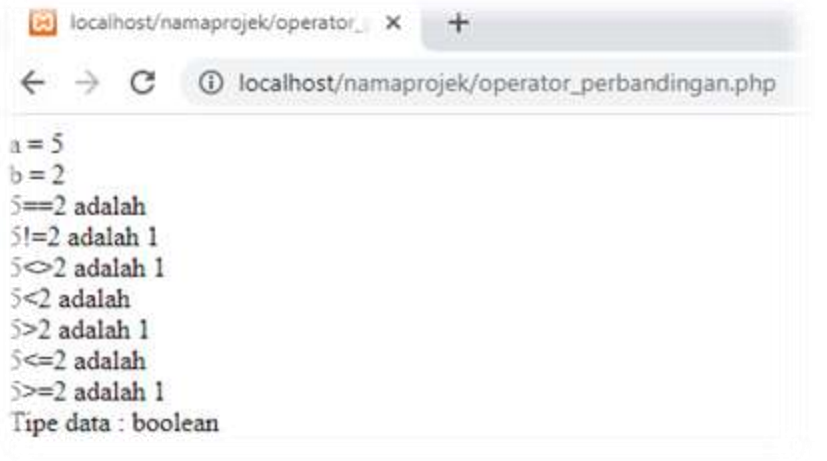
1. Operator perbandingan "==" berfungsi membandingkan dua buah nilai apakah sama. Jika sama maka hasilnya *true* atau satu. Sedangkan jika nilai keduanya tidak sama maka hasilnya *false* atau kosong.

2. Operator perbandingan "!=" berfungsi membandingkan dua buah nilai apakah tidak sama. Jika tidak sama maka hasilnya *true* atau satu. Sedangkan jika nilai keduanya sama maka hasilnya *false* atau kosong.
3. Operator perbandingan "<>" sama dengan operator "!=" berfungsi membandingkan dua buah nilai apakah tidak sama. Jika tidak sama maka hasilnya *true* atau satu. Sedangkan jika nilai keduanya sama maka hasilnya *false* atau kosong.
4. Operator perbandingan "<" berfungsi membandingkan dua buah nilai apakah lebih kecil dari. Jika lebih kecil dari nilai di sebelah kanannya maka hasilnya *true* atau satu. Sedangkan jika nilai keduanya sama atau lebih besar dari nilai di sebelah kanannya maka hasilnya *false* atau kosong.
5. Operator perbandingan ">" berfungsi membandingkan dua buah nilai apakah lebih besar dari. Jika lebih besar dari nilai di sebelah kanannya maka hasilnya *true* atau satu. Sedangkan jika nilai keduanya sama atau lebih kecil dari nilai di sebelah kanannya maka hasilnya *false* atau kosong.
6. Operator perbandingan "<=" berfungsi membandingkan dua buah nilai apakah lebih kecil dari atau sama dengan. Jika lebih kecil atau sama dengan dari nilai di sebelah kanannya maka hasilnya *true* atau satu. Sedangkan jika nilai keduanya lebih besar dari nilai di sebelah kanannya maka hasilnya *false* atau kosong.
7. Operator perbandingan ">=" berfungsi membandingkan dua buah nilai apakah lebih besar atau sama dengan dari. Jika lebih besar atau sama dengan dari nilai di sebelah kanannya maka hasilnya *true* atau satu. Sedangkan jika nilai keduanya lebih kecil dari nilai di sebelah kanannya maka hasilnya *false* atau kosong.

Untuk memahami fungsi operator dalam program PHP. Berikut ini adalah contoh program yang menggunakan operator sebagai prosesnya.

```
<?php
$a=5;
$b=2;
$hasil=$a==$b;
echo "a = $a <br>";
echo "b = $b <br>";
echo "$a==$b adalah $hasil <br>";
$hasil=$a!=$b;
echo "$a!=$b adalah $hasil <br>";
$hasil=$a<>$b;
echo "$a<>$b adalah $hasil <br>";
$hasil=$a<$b;
echo "$a<$b adalah $hasil <br>";
$hasil=$a>$b;
echo "$a>$b adalah $hasil <br>";
$hasil=$a<=$b;
echo "$a<=$b adalah $hasil <br>";
$hasil=$a>=$b;
echo "$a>=$b adalah $hasil <br>";

echo "Tipe data : ".gettype($hasil);
?>
```

Pada contoh program di atas, variabel $a bernilai 5 dan $b bernilai 2. Sedangkan variabel $hasil berfungsi menampung hasil dari operator perbandingan berupa satu atau kosong. Kita dapat melihat hasil dari operator perbandingan pada program di atas melalui tampilan hasil program berikut. Hasil dari kode program di atas dapat dilihat pada Gambar 4.19.



Gambar 4. 18. Operator Perbandingan

4.5 OPERATOR LOGIKA

Operaor logika sering digunakan untuk menggabungkan dua operator perbadingan atau lebih. Nilai keluaran dari operator logikan juga *boolean* sehingga sering digunakan untuk kondisi. Berikut ini operator-operator logika.

Tabel 4. 5. Operator Logika

No.	Operator	Keterangan	Output
1.	&& atau AND	Operator logika AND	Benar atau Salah
2.	|| atau OR	Operator logika OR	
3.	XOR	Operator logika XOR	
4.	!	Operator logika NOT	

1. Operator logika "&& atau AND" berfungsi membandingkan dua buah nilai *boolean*. Hasilnya akan *true* atau benar, jika dan hanya jika kedua *operand* bernilai benar. Selain itu akan bernilai *false* atau salah. Nilai *operand* yang sering digunakan adalah berasal dari operator perbandingan.
2. Operator logika "|| atau OR" berfungsi membandingkan dua buah nilai *boolean*. Hasilnya akan *true* atau benar, jika salah satu atau kedua *operand* bernilai *true*. Selain itu akan bernilai *false* atau salah. Nilai

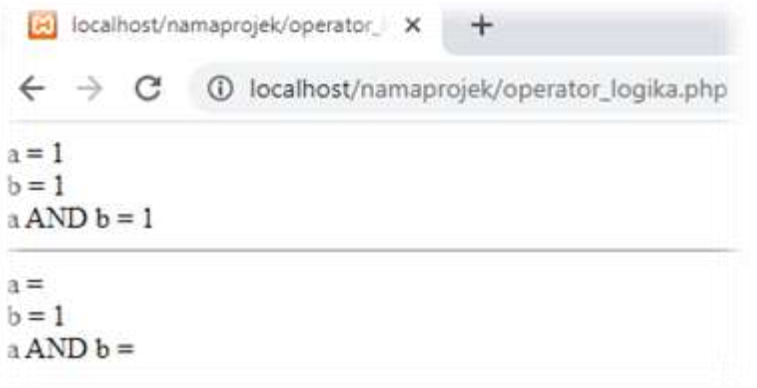
operand yang sering digunakan adalah berasal dari operator perbandingan.

3. Operator logika “XOR” berfungsi membandingkan dua buah nilai *boolean*. Hasilnya akan *true* atau benar, jika *operand* bernilai berbeda. Selain itu akan bernilai *false* atau salah. Nilai *operand* yang sering digunakan adalah berasal dari operator perbandingan.
4. Operator logika “!” atau “NOT” berfungsi membalikkan nilai *boolean*. Jikan nilai *boolean true* atau benar, maka operator logika ”NOT” akan membalikkannya menjadi *false* atau salah, begitu juga berlaku sebaliknya.

Berikut ini adalah contoh program untuk operator logika AND dan OR. Operator logika ini berfungsi mengecek hasil dari operator perbandingan.

```
<?php
$user="admin";
$status="login";

$a=$user=="admin";
$b=$status=="login";
$logika_and=$a AND $b;

echo "a = $a <br>";
echo "b = $b <br>";
echo " a AND b = $logika_and <br>";
echo "<hr>";

$a=$user=="user";
$b=$status=="login";
$logika_and=$a AND $b;

echo "a = $a <br>";
echo "b = $b <br>";
echo " a AND b = $logika_and <br>";
?>
```

Variabel *a* dan *b* yang pertama bernilai benar sehingga jika di lakukan operasi AND hasilnya juga benar atau 1. Sedangkan variabel *a* yang kedua bernilai salah dan variabel *b* kedua bernilai benar, jika di lakukan operasi AND hasilnya salah (tidak ada). Hasil dari kode program di atas dapat dilihat pada Gambar 4.20.



Gambar 4. 19. Operator Logika

4.6 OPERATOR PENGGABUNGAN *STRING*

Operator yang digunakan adalah titik ".". Operator ini berfungsi untuk menggabungkan *string*. Operator ini sudah dibahas pada bagian bab variabel.

4.7 OPERATOR *INCREMENT* DAN *DECREMENT*

Operator ini sering dijumpai pada perulangan. Berikut ini adalah operator-operator *increment* dan *decrement*.

Tabel 4. 6. Operator Increment dan Decrement

No.	Operator	Keterangan	Contoh
1.	++ (di depan)	Naik satu di awal iterasi	++$i
2.	++ (di belakang)	Naik satu setelah iterasi selesai	$i++
3.	-- (di depan)	Turun satu di awal iterasi	--$i
4.	-- (di belakang)	Turun satu setelah iterasi selesai	$i--

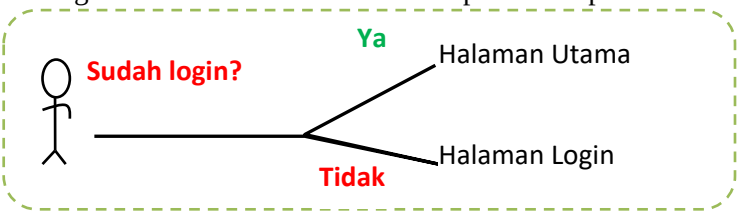
1. Operator “++” terletak di awal nama variabel, biasa berada di dalam perulangan. Contoh ++$i, artinya $i=$i+1. Dimana penambahan dilakukan begitu variabel di defenisikan.
2. Operator “++” terletak setelah nama variabel, biasa berada di dalam perulangan. Contoh $i++, artinya $i=$i+1. Dimana penambahan dilakukan setelah satu perulangan selesai.
3. Operator “--” terletak di awal nama variabel, biasa berada di dalam perulangan. Contoh --$i, artinya $i=$i-1. Dimana pengurangan dilakukan begitu variabel di defenisikan.
4. Operator “--” terletak setelah nama variabel, biasa berada di dalam perulangan. Contoh $i--, artinya $i=$i-1. Dimana pengurangan dilakukan setelah satu perulangan selesai.

BAB 5
KONDISI

Kondisi atau percabangan merupakan bagian penting dari pemrograman yang harus dipahami. Kondisi memungkinkan perlakukan yang berbeda terhadap suatu keadaan yang berbeda. Ada dua cara membuat kondisi yaitu dengan *If-Else* dan *Switch-Case*. Pemahaman yang diperlukan untuk pembahasan ini adalah variabel, operator perbandingan, dan operator logika. Jika belum paham maka kembali pelajari materi tersebut sebelum memulai materi ini.

5.1 IF-ELSE

Untuk menyelesaikan masalah dalam kondisi sederhana sampai rumit, *If-Else* ini mampu mengatasinya. Kondisi dapat digambarkan sebagai percabangan, jika benar maka ia masuk ke dalam cabangnya, sedangkan jika salah maka tidak masuk ke dalam cabang tersebut. Hal tersebut menjelaskan bahwa nilai *true* dan *false* diperlukan oleh kondisi. Sehingga operator yang sering digunakan dalam membuat kondisi adalah operator perbandingan dan atau operator logika. Gambar ilustrasi kondisi dapat dilihat pada Gambar 5.1.



Gambar 5. 1. Kondisi

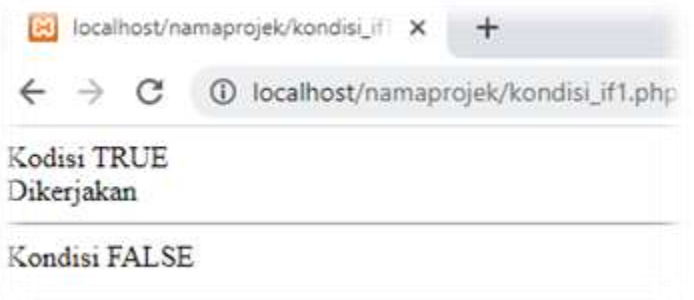
Dari ilustrasi di atas terlihat perlakuan berbeda terhadap orang yang sudah login dengan yang belum login. Berikut ini contoh program untuk kondisi.

```
if (kondisi) {
   # kode perintah...
}
```

Penulisan if dikuti dengan kodisi yang berada di dalam kurung. Sedangkan perintah yang akan dikerjakan jika kondisi benar berada di dalam kurung kurawal. Jika kondisinya salah makan perintah diabaikan. Berikut ini contoh dasar kondisi menggunakan if.

```
<?php
echo "Kodisi TRUE <br>";
if (TRUE) {
   echo "Dikerjakan";
}
echo "<hr>";
echo "Kondisi FALSE";
if (FALSE) {
   echo "Tidak Dikerkajan";
}
?>
```

Pada contoh program di atas, kondisi pertama bernilai *true*, maka perintah di dalamnya dikerjakan. Sedangkan kondisi kedua bernilai *false*, maka perintah di dalamnya tidak dikerjakan. Hasil dari kode program di atas dapat dilihat pada Gambar 5.2.

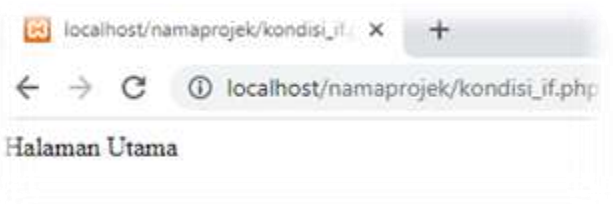


Gambar 5. 2. Kondisi Menggunakan *Boolean*

Setelah paham bahwa kondisi bekerja dengan bilangan *boolean*, jika benar blok perintah dikerjakan dan jika salah blok perintah dilewati saja. Maka selanjutnya mengisi kondisi dengan operator perbandingan sebagai berikut.

```
<?php
$status="login";
if ($status=="login") {
   echo "Halaman Utama";
}else{
   echo "Halaman Login";
}
?>
```

Arti kode di atas, didefinisikan variabel status bernilai login. Kondisinya jika variabel status bernilai login maka cetak “Halaman Utama”, Selain itu (*else*) cetak “Halaman Login”. Hasil dari kode program di atas dapat dilihat pada Gambar 5.3.

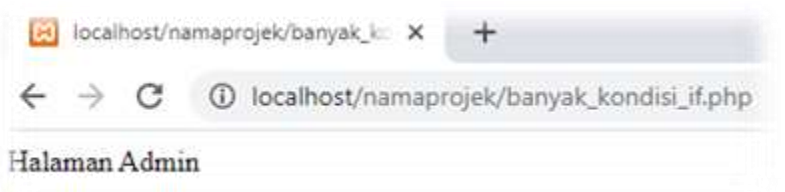


Gambar 5. 3. Kondisi Dengan *If-Else*

Contoh di atas menggunakan satu kondisi, namun bagaimana jika ada banyak kondisi yang harus diproses sekaligus?. Solusinya adalah menggunakan operator logika, sebagaimana yang sudah dijelaskan pada bab sebelumnya. Berikut ini contoh program kondisi menggunakan operator logika.

```
<?php
$status="login";
$user="Admin";
if ($status=="login" AND $user="Admin") {
   echo "Halaman Admin";
}else{
   echo "Halaman User";
}
?>
```

Pada kode program di atas, kondisi menggunakan dua buah nilai *boolean* sebagai *operand* dan menggunakan satu operator logika yaitu AND. Nilai *boolean* di dapat dari operator perbandingan. Variabel $status yang bernilai "login" dibandingkan dengan "login" melalui operator "==" dan hasilnya benar. Kemudian variabel $user yang bernilai "Admin" dibandingkan dengan "Admin" melalui operator "==" dan hasilnya benar. Benar dengan benar jika di AND kan maka hasilnya juga benar. Oleh karena itu perintah di dalam if dijalankan. Hasil dari kode program di atas dapat dilihat pada Gambar 5.4.



Gambar 5. 4. Banyak Kondisi Dengan *If-Else*

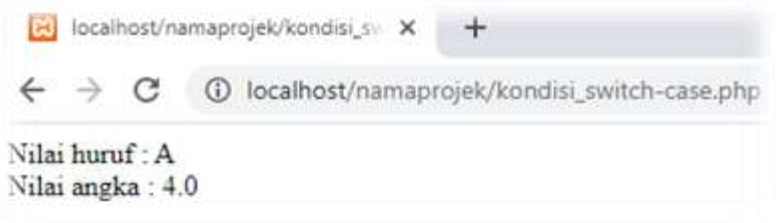
5.2 SWITCH-CASE

Cara lain menyelesaikan kondisi adalah menggunakan *switch-case*. Dengan menggunkakan *switch-case* dapat menghasilkan kode program yang lebih sederhana sehingga mudah dibaca. Umumnya digunakan untuk menyelasaikan kasus pilihan. Untuk lebih jelasnya berikut kode program *switch-case*.

```
<?php
$nilai="A";
echo " Nilai huruf : $nilai <br>";
switch ($nilai) {
   case 'A':
      echo "Nilai angka : 4.0";
      break;
   case 'B':
      echo "Nilai angka : 3.0";
      break;
   case 'C':
      echo "Nilai angka : 2.0";
```

```
        break;
    case 'D':
        echo "Nilai angka : 1.0";
        break;
    case 'E':
        echo "0";
        break;

    default:
        echo "Nilai Salah";
        break;
}
?>
```

Pada program di atas dapat dilihat variabel yang akan di cek berada di dalam *switch* yaitu $nilai. Pengecekan menggunakan perintah *case*. Apakah nilai sama dengan “A”, “B” dan seterusnya. Jika benar maka perintah sebelum *break* dikerjakan. Hasil dari kode program di atas dapat dilihat pada Gambar 5.5.



Gambar 5. 5. Kondisi Dengan *Switch-Case*

Kesimpulanya menggunakan *switch-case* hanya mampu mengecek isi sebuah variabel seperti menggunakan operator “==“ jika pada *If-Else*. Jika isinya sama, maka perintah dijalankan.

Perulangan sering ditemukan dalam pemrograman seperti halnya kondisi. Perulangan berfungsi untuk melakukan pekerjaan atau perintah beberapa kali. Menuliskan perintah dalam perulangan lebih efisien daripada menuliskan perintah secara berulang-ulang, karena pada perulangan perintah dapat dituliskan sekali saja. Ada dua jenis perulangan yaitu: perulangan terhitung dan perulangan tak terhitung.

Perulangan terhitung adalah perulangan yang jumlah perulangannya diketahui. Cotohnya membuat penomoran 1 samapai 10 atau yang diinginkan, jadi perulangan sangat jelas jumlahnya. Perulangan tak terhitung sebaliknya, yaitu jumlah perulangannya belum diketahui pasti. Contohnya ketika mencari sebuah kata pada tabel. Tentunya kata tersebut tidak diketahui secara pasti pada baris berapa. Pencarian itu selesai ketika kata ditemukan, tidak peduli berapa banyak baris yang sudah dibaca.

6.1. FOR

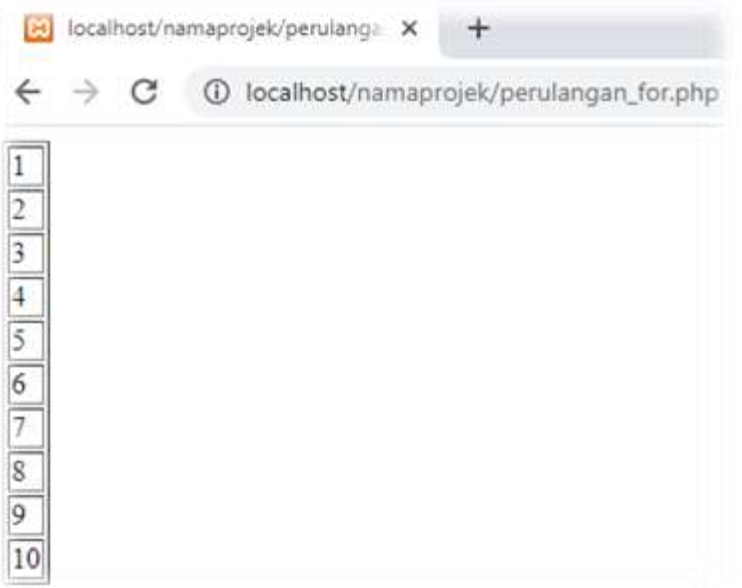
Perulangan menggunakan *for* adalah untuk perulangan terhitung. Berikut ini contoh program perulangan menggunakan for.

```
for ($i=1; $i <= 10 ; $i++) {
   # Perintah
}
```

Di dalam *for* terdiri dari bagian yang dipisahkan dengan titik koma (;). Bagian pertama dapat diartikan sebagai nilai awal yaitu 1 dan bagian kedua batasan akhir yaitu lebih kecil sama dengan 10, sedangkan yang terakhir adalah langkah. Langkah yaitu cara dari batas awal menuju akhir. Dalam hal ini langkahnya bertambah satu setelah iterasi selesai (*$i++*).

```
<?php
echo "<table border=1>";
for ($i=1; $i <= 10; $i++) {
        echo "<tr>";
            echo "<td>";
                echo $i;
            echo "</td>";
        echo "</tr>";
}
echo "</table>";
?>
```

Pada kode program di atas, adalah contoh membuat baris tabel menggunakan perulangan. Hasil dari kode program di atas dapat dilihat pada Gambar 6.1.

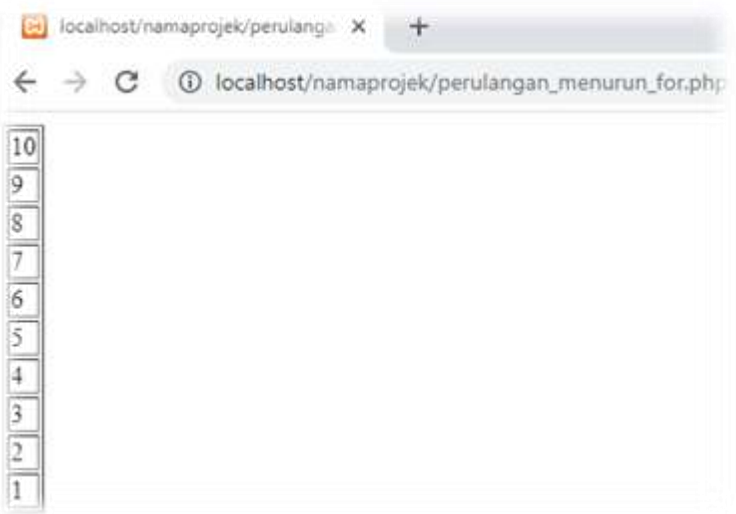


Gambar 6. 1. Perulangan Menggunakan *For*

Menggunakan perulangan perintah dapat dituliskan sekali saja, seperti membuat baris dan sel pada tabel. Berikut ini contoh perulangan menurun dengan *for*.

```
<?php
echo "<table border=1>";
for ($i=10; $i >= 1; $i--) {
      echo "<tr>";
         echo "<td>";
            echo $i;
         echo "</td>";
      echo "</tr>";
}
echo "</table>";
?>
```

Dari kode di atas nilai awal di berikan 10, batasan akhir lebih besar sama dengan 1, terakhir langkahnya menurun satu setelah iterasi selesai. Hasil dari kode program di atas dapat dilihat pada Gambar 6.2.

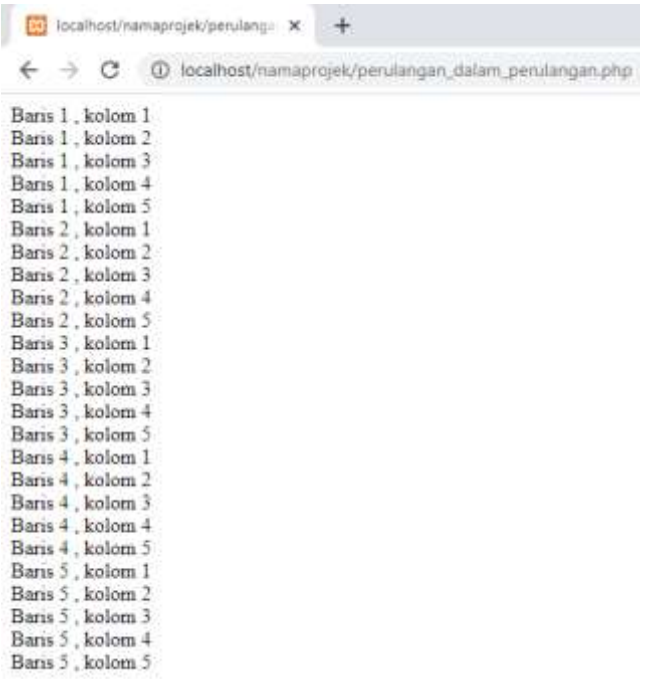


Gambar 6. 2. Perulangan Menurun Menggunakan *For*

Untuk perulangan dalam perulangan, maka perulangan paling dalam dikerjakan terlebih dahulu. Untuk contoh kasus kali ini hanya akan dibahas satu perulangan dalam perulangan.

```
<?php
   for ($baris=1; $baris <=5 ; $baris++) {
      for ($kolom=1; $kolom <=5 ; $kolom++) {
         echo "Baris $baris , kolom $kolom <br>";
      }
   }
?>
```

Pada kode program di atas, adalah contoh membuat baris dan kolom menggunakan perulangan. Hasil dari kode program di atas dapat dilihat pada Gambar 6.3.



Gambar 6. 3. Perulangan Dalam Perulangan

Kolom adalah perulangan yang berada di dalam, sedangkan baris adalah perulangan diluarnya. Jadi, perulangan yang berada di dalam dahulu diselesaikan, lalu perulangan yang berada di luar sampai keduanya selesai.

6.2. WHILE

Perulangan menggunakan *while* dapat digunakan untuk perulangan terhitung maupun tak terhitung. *While* sendiri dapat diartikan saat atau selagi. Artinya kode perintah akan terus dijalankan selagi kondisi masih terpenuhi. Kondisi yang dimaksud berada di dalam buka dan tutup kurung *while*.

```
<?php
$a=10000;
$total=0;
while ($a >=1) {
   $b=$a;
   $a= $a / 2;
   echo $b." / 2 = $a <br>";
   $total++;
}
echo "Banyak perulangan $total perulangan dengan batas nilai lebih besar sama dengan 1";
?>
```

Pada kode program di atas, adalah contoh perulangan tidak terhitung. Hasil dari kode program di atas dapat dilihat pada Gambar 6.4.

Gambar 6. 4. Perulangan Dengan *While*

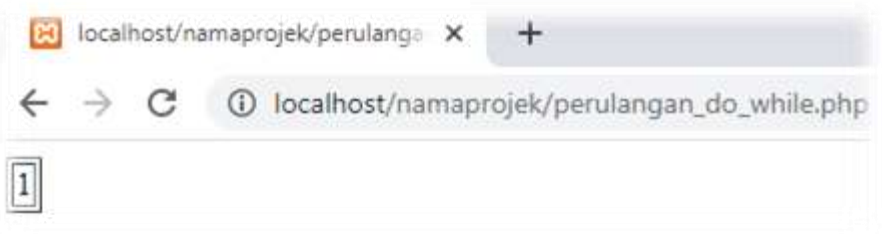
Pada gambar di atas adalah contoh perulangan tak terhitung. Sebenarnya masih banyak lagi contoh kasus perulangan tak terhitung lainnya. Perulagan tak terhitung umumnya digunakan untuk mencari suatu nilai. Sehingga pada kondisi tertentu pencarian tersebut selesai. Pada contoh program di atas perulangan akan selesai jika nilai variabel $a lebih kecil atau sama dengan satu.

6.3. DO-WHILE

Perulangan menggunakan *do-while* hampir sama dengan perulangan *while*. Perbedaannya perintah minimal sekali dijalankan pada blok perulangan walaupun kondisi dalam *while* bernilai *false*, sedangkan pada *while* mungkin saja perintah tidak pernah dijalankan sama sekali karena kondisi *false*.

```
<?php
$a=1;
echo "<table border=1>";
do {
   echo "<tr>";
      echo "<td>";
         echo $a;
      echo "</td>";
   echo "</tr>";
$a++;
} while ($a == 0);
echo "</table>";
?
```

Pada kode di atas, kondisi pada *while* bernilai *false*, namun pada *do-while* perintah pada perulangan dijalankan dulu baru kondisi dicek. Sehingga kode pada blok perulangan minimal dijalankan sekali. Hasil dari kode program di atas dapat dilihat pada Gambar 6.5.



Gambar 6. 5. Perulangan Dengan *Do-While*

Array adalah tipe data yang mampu menyimpan lebih dari satu nilai yang sejenis dalam sebuah variabel menggunakan *index*. Nilai yang disimpan dapat berupa angka maupun teks, namun yang terpenting sejinis. Pemilihan tipe data yang tepat dalam membuat program akan membuat kode program menjadi efisien. Contohnya ketika akan menyimpan banyak nilai sejenis akan lebih efisien menggunakan *array* dari pada variabel biasa.

Array	
Index	***Value***
0/nama1	83
1/nama2	84
2/nama3	98
3/nama4	96
4/nama5	98

Gambar 7. 1. *Array*

Index yang digunakan dapat berupa angka ataupun teks, yang terpenting setiap index tidak sama.

7.1 DEKLARASI DAN PENGISIAN *ARRAY*

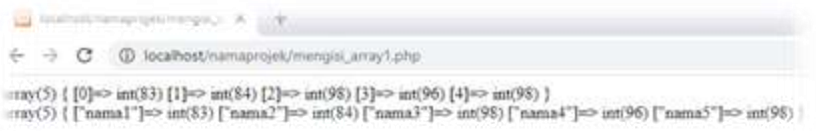
Seperti mendeklarasikan variabel umumnya pada PHP, deklarasi *array* juga langsung diisi nilai. Ada beberapa cara pengisian nilai dalam *array*.

7.1.1 Cara Mengisi *Array* Pertama

Cara ini dapat mengisi nilai sekaligus dengan bungkus buka tutup kurung biasa "( )". Brikut ini contoh program mengisi nilai *array* menggunakan *index* angka maupun teks.

```
<?php
$nilai_index_angka = array(83,84,98,96,98);//index angka
//atau
$nilai_index_teks = array('nama1' => 83, 'nama2' =>84, 'nama3' =>98, 'nama4' =>96, 'nama5' =>98);//index huruf
var_dump($nilai_index_angka);
echo "<br>";
var_dump($nilai_index_teks);
?>
```

Pada PHP deklarasi jumlah aggota *array* tidak diperlukan. Sehingga lebih mudah dalam memasukkan data ke dalam *array*. Hasil dari kode program di atas dapat dilihat pada Gambar 7.2.



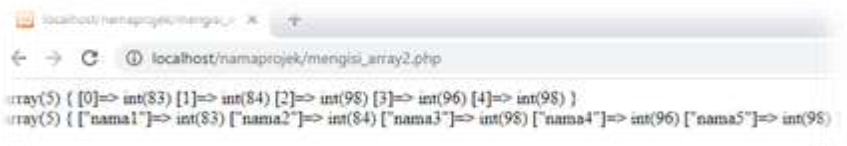
Gambar 7. 2. Cara Mengisi *Array* Pertama

7.1.2 Cara Mengisi *Array* Kedua

Dengan cara ini kita dapat mengisi nilai sekaligus dengan bungkus buka tutup kurung siku "[ ]". Brikut ini contoh program mengisi nilai *array* menggunakan *index* angka maupun teks.

```
<?php
$nilai_index_angka = [83,84,98,96,98];//index angka
//atau
$nilai_index_teks = ['nama1' => 83, 'nama2' =>84, 'nama3' =>98, 'nama4' =>96, 'nama5' =>98];//index huruf
var_dump($nilai_index_angka);
echo "<br>";
var_dump($nilai_index_teks);
?>
```

Pada contoh kode program di atas terlihat perbedaan dengan cara pertama. Perbedaan tersebut adalah pemberian *index* dalam mengisi *array*. Hasil dari kode program di atas dapat dilihat pada Gambar 7.3.



Gambar 7. 3. Cara Mengisi *Array* Kedua

7.1.3 Cara Mengisi *Array* Ketiga

Berbeda dari cara sebelumnya yaitu mengisi nilai sekaligus, cara ketiga ini mengisi nilai satu persatu.

```
<?php
$nilai_index_angka [0] = 83;
$nilai_index_angka [1] = 84;
$nilai_index_angka [2] = 98;
$nilai_index_angka [3] = 96;
$nilai_index_angka [4] = 98;
//atau
$nilai_index_teks ['nama1'] = 83;
$nilai_index_teks ['nama2'] = 84;
$nilai_index_teks ['nama3'] = 98;
$nilai_index_teks ['nama4'] = 96;
$nilai_index_teks ['nama5'] = 98;

var_dump($nilai_index_angka);
echo "<br>";
var_dump($nilai_index_teks);
?>
```

Pada contoh kode program di atas terlihat perbedaan dengan cara kedua. Perbedaan tersebut adalah pemberian nilai dilakukan satu persatu. Hasil dari kode program di atas dapat dilihat pada Gambar 7.4.

Gambar 7. 4. Cara Mengisi *Array* Ketiga

Cara ketiga ini banyak digunakan dalam prakteknya. Cara ini digunakan untuk mengisi data dalam *array* menggunakan perulangan.

7.2 MENCETAK NILAI *ARRAY*

Untuk mencetak nilai *array* sama seperti variabel biasanya, yaitu menuliskan variabel dengan *index*-nya. Ada beberapa cara untuk mencetak nilai dalam *array* baik secara langsung maupun menggunakan perulangan.

7.2.1 Cara Mencetak Nilai *Array* Pertama

Cara pertama mencetak nilai *array* dengan menuliskan nama variabel berserta *index*-nya satu per satu.

```
<?php
$nilai_index_angka = array(83,84,98,96,98);//index angka
//atau
$nilai_index_teks = array('nama1' => 83, 'nama2' =>84, 'nama3' =>98, 'nama4' =>96, 'nama5' =>98);//index huruf
echo "Index Angka <br><br>";
echo $nilai_index_angka[0];
echo "<hr>";
echo $nilai_index_angka[1];
echo "<hr>";
echo $nilai_index_angka[2];
echo "<hr>";
echo $nilai_index_angka[3];
```

```
echo "<hr>";
echo $nilai_index_angka[4];
echo "<hr>";
echo "<br><br><br>";
echo "Index Teks <br><br>";
echo $nilai_index_teks['nama1'];
echo "<hr>";
echo $nilai_index_teks['nama2'];
echo "<hr>";
echo $nilai_index_teks['nama3'];
echo "<hr>";
echo $nilai_index_teks['nama4'];
echo "<hr>";
echo $nilai_index_teks['nama5'];
echo "<hr>";
?>
```

Pada contoh kode program di atas terlihat cara untuk mencetak nilai dalam *array*. Hasil dari kode program di atas dapat dilihat pada Gambar 7.5.

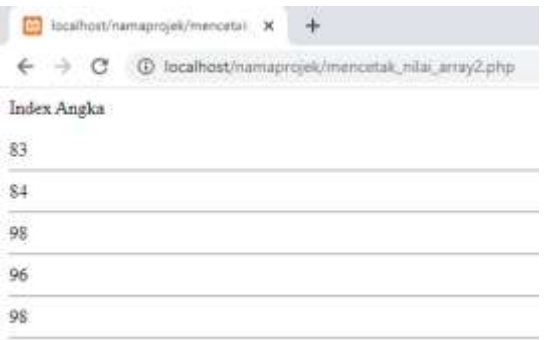
Gambar 7. 5. Cara Mencetak Nilai *Array* Pertama

7.2.2 Cara Mencetak Nilai *Array* Kedua

Cara mencetak nilai array kedua menggunakan *for*. *For* hanya mampu melakukan perulangan terhitung. Artinya batas perulangan atau kondisi harus diketahui. Terdapat fungsi untuk mengetahui jumlah isi dari sebuah *array* yaitu count(). Namun Cara ini hanya cocok untuk *index* angka, berikut contoh programnya.

```
<?php
$nilai_index_angka = array(83,84,98,96,98);//index angka
echo "Index Angka <br><br>";
for ($i=0; $i <count($nilai_index_angka); $i++) {
   echo $nilai_index_angka[$i]."<hr>";
}
?>
```

Pada contoh kode program di atas terlihat cara untuk mencetak nilai dalam *array* menggunakan *for*. Hasil dari kode program di atas dapat dilihat pada Gambar 7.6.



Gambar 7. 6. Cara Mencetak Nilai *Array* Kedua

7.2.3 Cara Mencetak Nilai *Array* Ketiga

Cara ketiga ini menggunakan perulangan khusus untuk *array* yang disediakan PHP. Cara ini cocok untuk kedua jenis *index*, baik angka maupun teks. Berikut ini contoh programnya.

```
<?php
$nilai_index_angka = array(83,84,98,96,98);//index angka
echo "Index Angka <br><br>";
$nilai_index_teks = array('nama1' => 83, 'nama2' =>84, 'nama3' =>98, 'nama4' =>96, 'nama5' =>98);//index huruf
foreach ($nilai_index_angka as $key => $value) {
   echo "Index $key value : ".$value."<hr>";
}
echo "<br><br><br>Index Teks <br><br>";
foreach ($nilai_index_teks as $key => $value) {
   echo "Index $key value : ".$value."<hr>";
}
?>
```

Pada contoh kode program di atas terlihat cara untuk mencetak nilai dalam *array* menggunakan *foreach* . Hasil dari kode program di atas dapat dilihat pada Gambar 7.7.

Gambar 7. 7. Cara Mencetak Nilai *Array* Ketiga

7.3 FUNGSI-FUNGSI PADA *ARRAY*

Ada banyak fungsi yang ada pada *array* PHP, salah satunya mencari jumlah data seperti pada pembahasan sebelumnya. Berikut ini adalah beberapa fungsi pada *array* PHP yang penting untuk diketahui.

Tabel 7. 1. Fungsi-Fungsi Dalam Array

No.	Fungsi	Keterangan
1.	count( )	Jumlah anggota *array*
2.	current( )	Anggota *array* pertama
3.	end( )	Anggota *array* terakhir
4.	array_search( )	Mencari nilai dalam *array*
5.	sort( )	Mengurutkan nilai *array* secara *ascending*
6.	rsort( )	Mengurutkan nilai *array* secara *descending*

Metode *get* digunakan untuk mengirim nilai ke dalam sebuah program. Hal tersebut umumnya diinput oleh pengguna menggunakan formulir. Akan tetapi nilai dapat dikirim langsung pada URL dengan mengikuti aturan-aturan tertentu. Sedangkankan mengambil nilai yang dikrim tersebut menggunakan variabel $_GET. Pada PHP perintanya sebagai berikut.

```
$_GET[‘nama_variabel’];
```

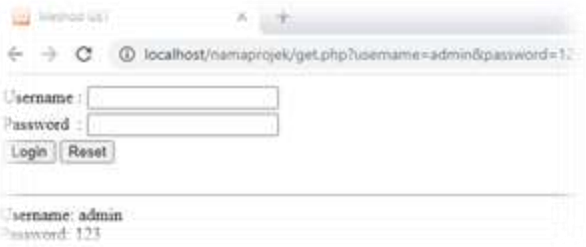
8.1 METODE *GET* PADA *FORM*

Untuk mengirimkan nilai ke suatu halaman salah satu metode yang dapat digunakan adalah metode *get*. Berikut ini contoh implementasi metode *get* pada sebuah *form* HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Method GET</title>
</head>
<body>
   <form action="" method="GET">
      <table>
         <tr>
            <td>Username</td>
            <td>:</td>
            <td><input type="text" name="username"></td>
         </tr>
         <tr>
            <td>Password</td>
            <td>:</td>
```

```
            <td><input type="password" name="password"></td>
        </tr>
        <tr>
            <td colspan="3">
            <button type="submit">Login</button>
            <input type="reset">
            </td>
        </tr>
    </table>
</form>
<br>
<hr>
<?php
if (!empty($_GET['username']) AND !empty($_GET['password'])) {
    echo "Username: $_GET[username] <br>";
    echo "Password: $_GET[password] <br>";
}

?>
</body>
</html>
```

Pada kode program di atas, metode *get* diatur melalui atribut method pada *tag* <form>. Data tersebut tersimpan dalam variabel $_GET. Variabel $_GET merupakan variabel *array* yang berisi data-data *username* dan *password*. Untuk mengambil data *username* misalnya, maka *index* pada variabel $_GET diisi *username* $_GET[username]. Hasil dari kode program di atas dapat dilihat pada Gambar 8.1.



Gambar 8. 1. Metode *GET* Pada *Form*

Dapat dilihat pada Gambar 8.1, di bagian URL terlihat username=admin&password=123. Informasi yang dikirim menggunakan metode *get* akan tampil pada URL. Sehingga metode ini tidak cocok untuk data sensitif. Selain itu isi dari variabel *username* maupun *password* dapat diubah, maupun diinput langsung melalui URL. Namun ada keunggulan dari metode *get* ini yaitu sebagai tempat meletakkan informasi saat halaman berpindah dengan cara yang lebih mudah tanpa harus menggunakan *form*.

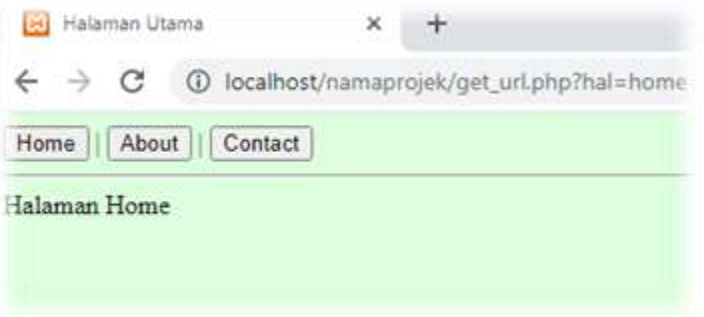
8.2 METODE *GET* PADA URL

Untuk mengirim data dari satu halaman ke halaman lain maupun halaman itu sendiri dengan mudah salah satu solusinya dengan mengirimnya langsung melalui URL. Aturan penulisan variabel pada URL yaitu diawali tanda tanya "?". Kemudian diikuti nama variabel dan isinya (menggunakan sama dengan) tanpa spasi. Apabila variabelnya lebih dari satu maka dipisahkan dengan "&".

```
<!DOCTYPE html>
<html>
<head>
        <title>Halaman Utama</title>
</head>
<body bgcolor="#ddffdd">
<a href="get_url.php?hal=home"><button>Home</button></a> | <a
href="get_url.php?hal=about"><button>About</button></a> | <a
href="get_url.php?hal=contact"><button>Contact</button></a>
<?php
if (isset($_GET['hal'])) {
```

```
    if ($_GET['hal']=='home') {
        echo "<hr>Halaman Home";
    }elseif ($_GET['hal']=='about') {
        echo "<hr>Halaman About";
    }elseif ($_GET['hal']=='contact') {
        echo "<hr>Halaman Contact";
    }else{
        echo "<hr>Halaman Tidak Ditamukan";
    }
}else{
    echo "<hr>Halaman Home";
}
?>
</body>
</html>
```

Pada contoh kode program di atas, nilai dikirim langsung lewat *link*. Nilai tersebut dimanfaatkan untuk kondisi. Sedangkan kondisi menentukan halaman yang akan tampil. Hasil dari kode program di atas dapat dilihat pada Gambar 8.2.



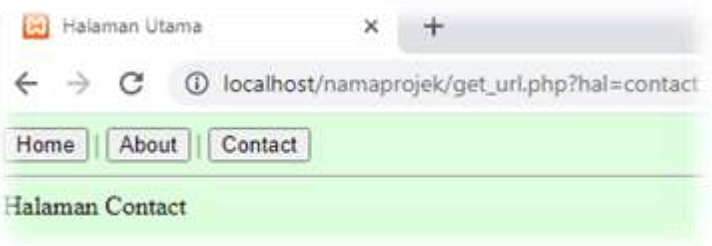
Gambar 8. 2. Metode *GET* Pada URL

8.3 BERPINDAH HALAMAN OTOMATIS

Hal yang umum dilakukan setelah *form* di *submit* adalah berpindah halaman otomatis dari URL tujuan submit ke halaman lain menggunakan fungsi *header( )*.

```
<?php
//redirect.php
header("location:get_url.php?hal=contact");
?>
```

Ketika file tersebut diakses maka akan langsung berpindah halaman. Hasil dari kode program di atas dapat dilihat pada Gambar 8.3.



Gambar 8. 3. Berpindah Halaman Otomatis

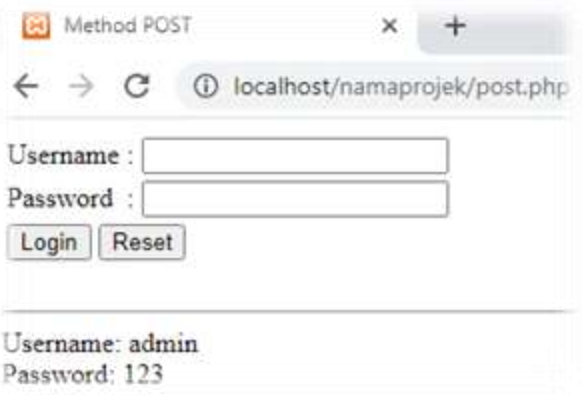
8.4 METODE *POST* PADA *FORM*

Metode *post* hanya dapat mengirim nilai dari sebuah *form*. Berbeda dengan metode *get*, nilai yang dikirim tidak tampil di URL. Metode *post* cocok untuk mengirim data sensitif seperti *password* agar tidak dapat dilihat.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Method POST</title>
</head>
<body>
   <form action="" method="POST">
      <table>
         <tr>
            <td>Username</td>
            <td>:</td>
            <td><input type="text" name="username"></td>
         </tr>
```

```
        <tr>
            <td>Password</td>
            <td>:</td>
            <td><input type="password" name="password"></td>
        </tr>
        <tr>
            <td colspan="3">
            <button type="submit">Login</button>
            <input type="reset">
            </td>
        </tr>
    </table>
</form>
<br>
<hr>
<?php
if (!empty($_POST['username']) AND !empty($_POST['password'])) {
    echo "Username: $_POST[username] <br>";
    echo "Password: $_POST[password] <br>";
}

?>
</body>
</html>
```

Untuk mengambil data yang dikirimkan formulir, perintah yang digunakan adalah $_POST[‘name_elemen_form’];. Data yang dikririm menggunakan metode POST tidak terlihat di URL. Hasil dari kode program di atas dapat dilihat pada Gambar 8.4.



Gambar 8. 4. Metode *POST* Pada *Form*

BAB 9
SESSION DAN COOKIES

9.1 SESSION

Pada dasarnya sebuah *server* tidak perduli siapa yang mengirimkan permintaan. Ketika URL diketikkan dan valid, maka *server* akan melayani dengan merespon permintaan tersebut. Terkadang informasi yang akan diberikan disebuah website berbeda untuk setiap orang, terutama pada sebuah sistem. Untuk itu sistem perlu mengenali siapa yang mengirim permintaan tersebut menggunakan *session*. *Session* di buat pada halaman login. Selain sebagai gerbang untuk masuk ke dalam sistem. *Session* mencatat informasi pada variabel *session* mengenai siapa orang yang masuk ke dalam sistem tersebut.

9.1.1 Cara Membuat *Session*

Cara membuat *session* hampir sama dengan membuat variabel biasa atau *array* yaitu ketika dibuat nilainya langsung diisi. Berikut ini cara membuat session.

```
session_start();
$_SESSION[‘status_login_apl_crud’]=’login’;
```

Fungsi session_start()harus dipanggil ketika hendak membuat *session*. *Session* di atas diberi nama status_login_apl_crud dan diisi login. Nama *session* harus unik agar tidak ada nama *session* yang sama dalam sebuah server.

9.1.2 Cara Menggunakan *Session*

Menggunakan *session* artinya mengecek nilai yang ada di dalam *session*. Cara yang digunakan untuk mengambil nilaipun sama dengan cara mengambil nilai sebuah variabel yaitu hanya menyebutkan saja

session yang dimaksud diawali dengan memanggil fungsi session_start(). Berikut ini cara menggunakan session.

```
session_start();
$_SESSION[‘status_login_apl_crud’];
```

9.1.3 Cara Menghapus *Session*

Menghapus *session* adalah hal yang biasa dilakukan ketika pengguna aplikasi *logout*. Berikut ini cara menghapus *session* yang terlah dibuat.

```
session_start();
unset($_SESSION['nama']); // untuk menghapus variabel session nama.
session_destroy(); // untuk menghapus semua variabel session.
```

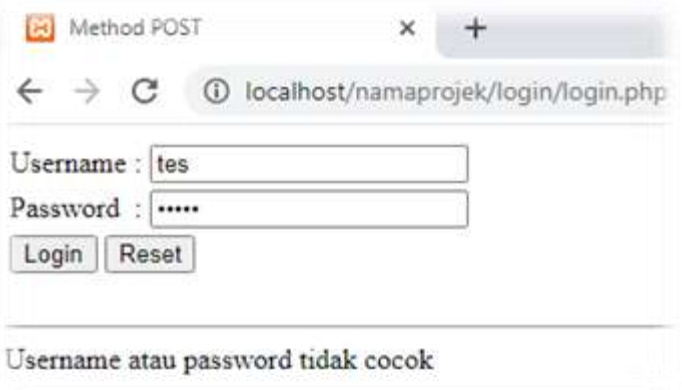
9.1.4 Contoh Kasus *Session*

Contoh kasus kali ini adalah pada halaman login untuk tempat membuat *session*. Sedangkan halaman *home* mengecek atau menggunakan *session*. Kemudian menu *logout* untuk menghapus *session*. Berikut ini contoh programnya.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Method POST</title>
</head>
<body>
   <form action="" method="POST">
      <table>
         <tr>
            <td>Username</td>
            <td>:</td>
            <td><input type="text" name="username"></td>
         </tr>
```

```
        <tr>
            <td>Password</td>
            <td>:</td>
            <td><input type="password" name="password"></td>
        </tr>
        <tr>
            <td colspan="3">
            <button type="submit">Login</button>
            <input type="reset">
            </td>
        </tr>
    </table>
</form>
<br>
<hr>
<?php
//daftar username dan password
$list_user=['user1'=>'admin-123','user2'=>'user-222','user3'=>'coba-234'];
if (!empty($_POST['username']) AND !empty($_POST['password'])) {
    $cari=$_POST['username']."-".$_POST['password'];
    $hasil=array_search($cari,$list_user);
    if (!empty($hasil)) {
        session_start();
        $_SESSION['status_login_apl_crud']="login";
        $_SESSION['username_apl_crud']=$_POST['username'];
        header("Location:home.php");
    }else{
        echo "Username atau password tidak cocok";
    }
}

?>
</body>
</html>
```

Karena kita belum membahas *database*, jadi daftar *user* disimpan di *array*. Ketika datanya ada maka *session* dibuat dan diarahkan ke halaman *home*. Hasil dari kode program di atas ketika login gagal dapat dilihat pada Gambar 9.1.

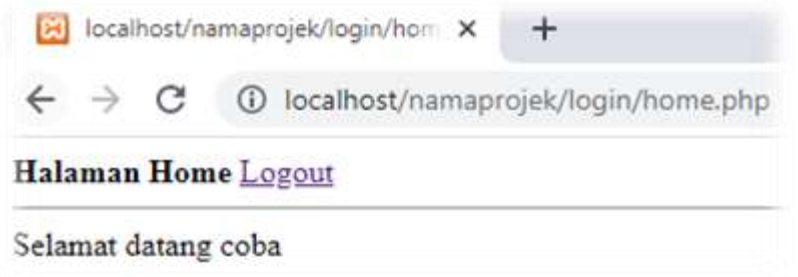


Gambar 9. 1. Login Gagal

Jika login berhasil maka session status_login_apl_crud dan username_apl_crud dibuat dan di arahkan otomatis ke halaman home.

```
<?php
session_start();
if (isset($_SESSION['status_login_apl_crud']) AND
$_SESSION['status_login_apl_crud']=='login') {
   echo "<b>Halaman Home</b> <a href='?aksi=logout'>Logout</a><hr>";
   echo "Selamat datang $_SESSION[username_apl_crud]";
}else {
   header("Location:login.php");
}

if (isset($_GET['aksi']) AND $_GET['aksi']=='logout') {
   session_destroy();
   header("Location:login.php");
}
?>
```

Pada kode program di atas, dilakukan pengecekan *session* dengan kondisi. Jika nilai *session* ada dan sama dengan "login" maka akan tampil halaman *home*. Sedangkan jika *session* kosong, maka halaman akan otomatis berpindah ke halaman login. Kemudian, jika menu logout di klik maka *session* akan di hapus. Lalu halaman akan otomatis berpindah ke halaman login. Hasil dari kode program di atas ketika login berhasil dapat dilihat pada Gambar 9.2.



Gambar 9. 2. Login Berhasil

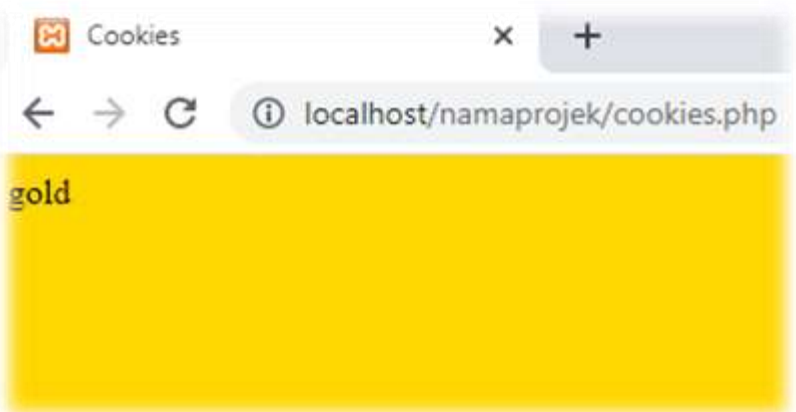
Di halaman home ini *session* di cek, jika halam ini di akses tanpa melalui login maka otomatis kembali ke halaman login. Ketika menu *logout* di klik maka *session* dihapus.

9.2 COOKIES

Berbeda dengan *session*, *cookies* menyimpan data atau informasi di komputer pengguna yang di lakukan *web browser*. Informasi yang disimpan lebih tahan lama dan dikenali di setiap halaman web, beda halnya dengan variabel biasa yang sementara.

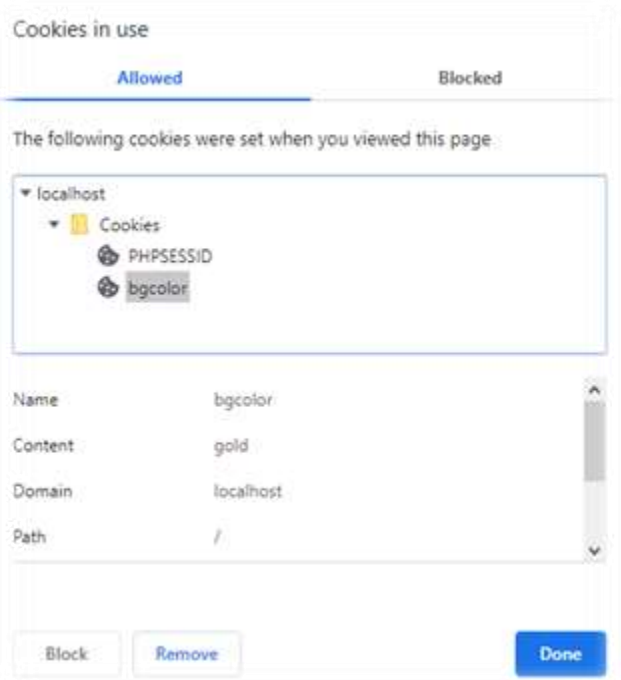
```
<?php
$nama='bgcolor';
$nilai="gold";
setcookie($nama, $nilai, time() + (10), "/");
?>
<html>
<head>
   <title>Cookies</title>
</head>
<body bgcolor="<?php echo $_COOKIE['bgcolor']?>">
<?php echo $_COOKIE['bgcolor'];?>
</body>
</html>
```

Pada kode program di atas, *cookies* di input nilai menggunakan perintah *setcookie*. *Cookies* bekerja setelah halaman jalankan kedua kalinya. Hasil dari kode program di atas dapat dilihat pada Gambar 9.3.



Gambar 9. 3. *Cookies*

Cookies di atas akan menyimpan informasi selama 10 detik. Berikut ini nilai *session* yang dapat dilihat melalui *web browser* seperti pada Gambar 9.4.



Gambar 9. 4. *Cookies* di *Web Browser*

BAB 10
APLIKASI CRUD

10.1 MYSQL

MySQL adalah *database management system* (DBMS) *open soure* yang gratis. Ketika *menginstall* paket xampp pada komputer paket mysql sudah ada didalamnya. Perintah atau bahasa yang digunakan pada *mysql* adalah *Structure Query Language* (SQL). Pada bab ini akan dijelaskan cara melihat, menambahkah, mengubah, dan menghapus data pada *database mysql*.

10.1.1 Membuat *Database*

Database seperti map (folder) tempat meletakkan file-file untuk disimpan. Sedangkan file-file tersebut adalah isi dari *database* yaitu tabel-tabel. Berikut ini akan dicontohkan pembuatan *database* bernama db_apl_crud.

1. Jalankan mysql dari *control pane*l xampp
2. Buka folder bin

```
C:\xampp\mysql\bin>
```

3. Terhubung ke mysql

```
mysql -u root
```

4. Membuat Database

```
create database db_apl_crud;
```

5. Menggunakan database

```
use db_apl_crud;
```

10.1.2 Membuat Tabel

Tabel seperti sebuah file yang berisi data berbentuk tabel. Dalam satu *database* bisa terdiri dari satu atau lebih tabel. Setelah langkah sebelumnya dilakukan berikut cara membuat sebuah tabel pada *mysql*.

```
MariaDB [db_apl_crud]> CREATE TABLE tbl_user(
   -> id_user int(11) AUTO_INCREMENT,
   -> username varchar(20),
   -> password varchar(60),
   -> PRIMARY KEY (id_user)
   -> );
```

10.1.3 Menambahkan Data (*Insert*)

Untuk menambahkan data baru ke dalam *database* perintah yang digunakan adalah INSERT. Berikut perintah lengkapnya untuk menambahkan data baru ke dalam tbl_user.

```
INSERT INTO tbl_user values('1','admin','123');
```

10.2 MEMBACA DATA (*READ*)

Untuk membaca atau menampilkan data dari sebuah tabel perintah yang digunakan adalah SELECT. Berikut ini perintah lengkapnya untuk membaca data dari tbl_user.

```
SELECT * FROM tbl_user;
```

Hasil query di atas sebagai berikut.

```
+---------+----------+----------+
| id_user | username | password |
+---------+----------+----------+
|       1 | admin    | 123      |
+---------+----------+----------+
```

10.3 MENGUBAH DATA (*UPDATE*)

Untuk mengubah data pada sebuah tabel perintah yang digunakan adalah UPDATE. Berikut ini perintah lengkapnya untuk mengubah data dari tbl_user.

```
UPDATE tbl_user SET username='pemilik', password='234' where id_user='1';
```

Setelah data berhasil diubah. Maka kita dapat melihat kembali hasilnya dengan perintah SELECT. Berikut ini adalah perintah tersebut.

```
SELECT * FROM tbl_user;
```

Hasil query di atas sebagai berikut.

```
+---------+----------+----------+
| id_user | username | password |
+---------+----------+----------+
|       1 | pemilik  | 234      |
+---------+----------+----------+
```

10.4 MENGHAPUS DATA

Untuk menghapus sebuah data pada sebuah tabel perintah yang digunakan adalah DELETE. Berikut ini perintah lengkapnya untuk menghapus data dari tbl_user.

```
DELETE FROM tbl_user WHERE id_user='1';
```

Selanjutnya tampilkan kembali isi *tbl_user*

```
SELECT * FROM tbl_user;
```

Hasil query di atas sebagai berikut.

```
Empty set (0.00 sec)
```

10.5 FUNGSI-FUNGSI MYSQL DALAM PHP

Ada banyak fungsi dalam PHP yang berkaitan dengan *mysql* yang sering digunakan. Berikut ini adalah beberapa fungsi tersebut.

Tabel 10. 1. Fungsi-Fungsi Mysql Dalam PHP

No	Fungsi	Keterangan	Contoh
1.	mysqli_connect()	Untuk koneksi ke server	**mysqli_connect("localhost","root","","db_crud")** Contoh penggunaan: $con = mysqli_connect ("localhost", "root", "", "db_crud");
2.	mysqli_query()	Untuk mengirimkan perintah query. Terdiri dari dua parameter yaitu: koneksi, dan SQL	**mysqli_query($con,$sql)** Contoh penggunaan: $sql="Select * from tbl_admin"; $hasil = mysqli_query($con, $sql);
3.	mysqli_fetch_array()	Mengambil hasil baris sebagai asosiatif , array numerik , atau keduanya. Singkatnya untuk menampung baris tabel menjadi array	**mysqli_fetch_array($hasil)** Contoh penggunaan: $row=mysqli_fetch_array($hasil); echo $row['nama_admin];
4.	mysqli_num_rows()	Mengambil jumlah baris di dalam tabel.	**mysqli_num_rows($hasil)** Contoh penggunaan: $hasil=mysqli_query($con, $sql); $jlh=mysqli_num_rows($hasil); echo $jlh;

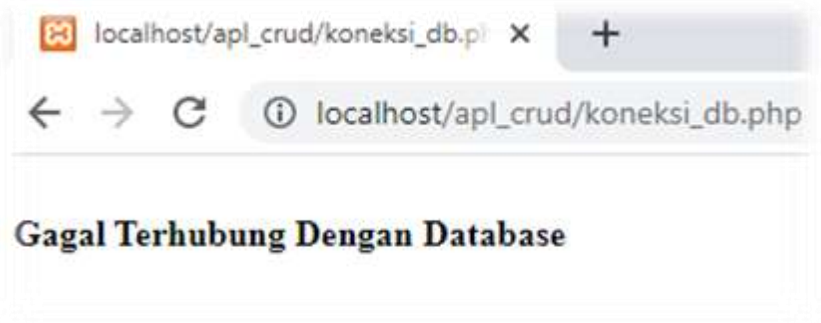
10.6 FILE KONEKSI

File koneksi diperlukan disetiap file yang membutuhkan koneksi dengan *database*, namun file ini cukup satu. Jika diperlukan, file ini di *include*

ke dalam file yang membutuhkan koneksi dengan *database*. Berikut ini contoh program koneksi ke *database* mysql menggunakan PHP.

```
<?php
error_reporting(0);
$koneksi_db=mysqli_connect("localhost","root","","db_apl_crud") or
die("<br> <b>Gagal Terhubung Dengan Database</b>");
?>
```

Pada kode program di atas, perintah untuk terhubung dengan *database* adalah mysqli_connect. Parameter yang digunakan adalah nama *server*, kemudian nama *user database*, *password*, dan terakhir nama dari *database*. Hasil dari kode program di atas dapat dilihat pada Gambar 10.1.



Gambar 10. 1. Gagal Terhung Dengan *Database*

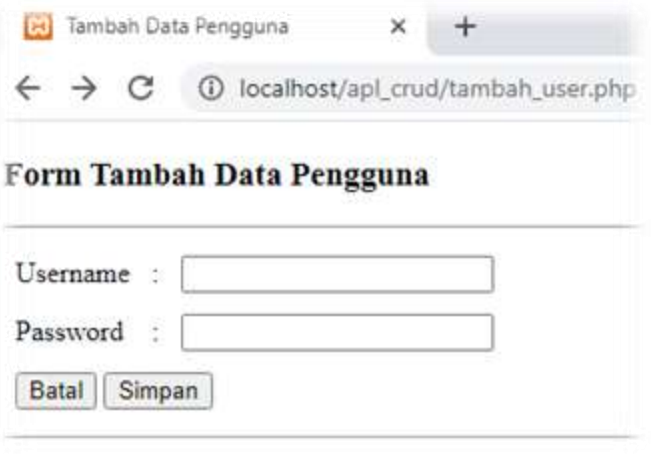
Ketika koneksi berhasil maka tidak ada pesan yang ditampilkan di halaman. Sedangkan jika gagal maka muncul pesan peringatan.

10.7 FILE TAMBAH DATA (*CREATE*)

Setelah mempelajari perintah SQL untuk menambahkan data ke dalam *database* menggunakan *command line*. Sekarang saatnya untuk menambahkan *database* melalui aplikasi.

```
<!DOCTYPE html>
<head>
   <title>Tambah Data Pengguna</title>
</head>
<body>
<h3>Form Tambah Data Pengguna</b></h3><hr>
   <form method="POST" action="aksi_tambah_user.php">
      <table cellpadding="5">
      <tr>
         <td>Username</td>
         <td>:</td>
         <td><input type="text" name="username"></td>
      </tr>
      <tr>
         <td>Password</td>
         <td>:</td>
         <td><input type="password" name="password"></td>
      </tr>
      <tr>
         <td colspan="3">
            <input type="button" value="Batal">
            <input type="submit" value="Simpan">
         </td>
      </tr>
      </table>
   </form>
<hr>
</body>
</html>
```

Pada kode program di atas, terlihat sebuat tag formulir lengkap dengan elemennya. *Method* yang digunakan adalah *POST*. Formulir tersebut akan dikirimkan kepada file aksi_tambah_user.php. Hasil dari kode program di atas dapat dilihat pada Gambar 10.2.



Gambar 10. 2. Halaman Tambah Data

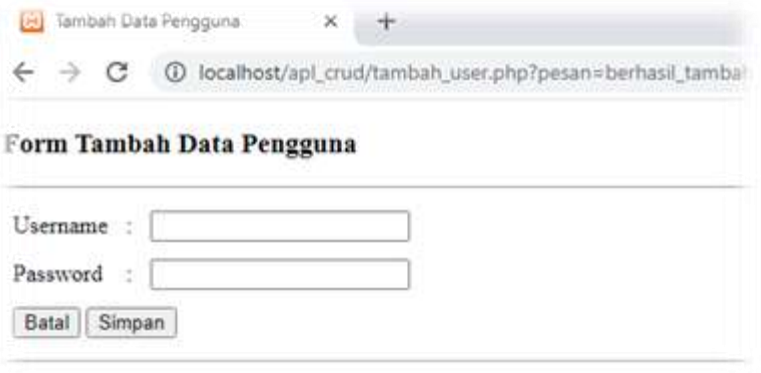
Seletah tampilan form tambah data pengguna selesai dibuat. Selanjutnya selesaikan aksi dari form tersebut.

```
<?php
include "koneksi_db.php";
$username=$_POST['username'];
$password=$_POST['password'];
$sql="INSERT INTO tbl_user (username,password)
VALUES('$username','$password')";
mysqli_query($koneksi_db,$sql) or die (mysqli_error($koneksi_db));
header("Location:tambah_user.php?pesan=berhasil_tambah");
?>
```

Pada kode program di atas, terlihat data-data elemen formulir diambil menggunakan variabel $_POST. Variabel tersebut merupakan *array* yang membawa banyak data. Data-data tersebut tersimpan berdasarkan *index* variabel tersebut. *Index* tersebut berasal dari *name* elemen form pengirim.

Setelah data-data ditampung dalam variabel $username dan $password. Selanjutnya data tersebut disisipkan ke dalam perintah SQL INSERT. Setelah selesai memasukkan data ke dalam database. Maka halaman otomatis

berpindah ke halaman tambah_user.php dengan membawa pesan berhasil ditambah. Pesan berhasil dari kode program di atas dapat dilihat pada Gambar 10.3.



Gambar 10. 3. Aksi Tambah Data

Setelah data berhasil ditambah silahkan lihat hasilnya di *database* menggunakan *comman line* seperti berikut.

```
MariaDB [db_apl_crud]> select * from tbl_user;
+---------+----------+----------+
| id_user | username | password |
+---------+----------+----------+
|       1 | tes      | 234      |
+---------+----------+----------+
```

10.8 MEMBACA DAN MENAMPILKAN DATA (*READ*)

Setelah menambahkan data ke dalam tabel, selajutnya yaitu membaca dan menampilkan isi data tersebut dari dalam tabel. Berikut ini file untuk membaca data dan menampilkan data dari *database* bernama index.php.

```
<!DOCTYPE html>
<head>
   <title>Data User</title>
</head>
<body>
<?php
include "koneksi_db.php";
$sql="SELECT * from tbl_user";
$hasil=mysqli_query($koneksi_db,$sql);
?>
<h3>DATA USER</h3> <a href="tambah_user.php"><input type="button"
value="TAMBAH DATA"></a><hr>
   <table border="1">
      <tr>
         <th>NO.</th>
         <th>ID USER</th>
         <th>USERNAME</th>
         <th>PASSWORD</th>
         <th>AKSI</th>
      </tr>
   <?php
   $no=1;
   while ($baris=mysqli_fetch_array($hasil)) {
      echo "
      <tr>
         <td>$no</td>
         <td>$baris[id_user]</td>
         <td>$baris[username]</td>
         <td>$baris[password]</td>
         <td><a href='edit_user.php?id_user=$baris[id_user]'>EDIT</a> | <a
href='aksi_hapus_user.php?id_user=$baris[id_user]' onclick='return
confirm_del()'>HAPUS</a></td>
      </tr>
      ";
      $no++;
```

```
    }
    ?>
    </table>
</body>
</html>
<script>
function confirm_del(){
    return confirm('are you sure?');
}
</script>
```

Pada kode program di atas, perintah SQL yang dikirimkan oleh PHP ada SELECT. Nilai balik dari *query* di atas di tampung di dalam variabel $hasil. Kemudian $hasil dijadikan *array* bernama $row yang kemudian ditampilkan dalam perulangan while. Hasil dari kode program di atas dapat dilihat pada Gambar 10.4.

Gambar 10. 4. Membaca dan Menampilkan Data (*Read*)

Pada Gambar 10.4 terdapat dua buah aksi yaitu edit dan hapus. Aksi tersebut berupa *link*. *Link* tersebut membawa *primary key* dari data yang akan diedit atau dihapus.

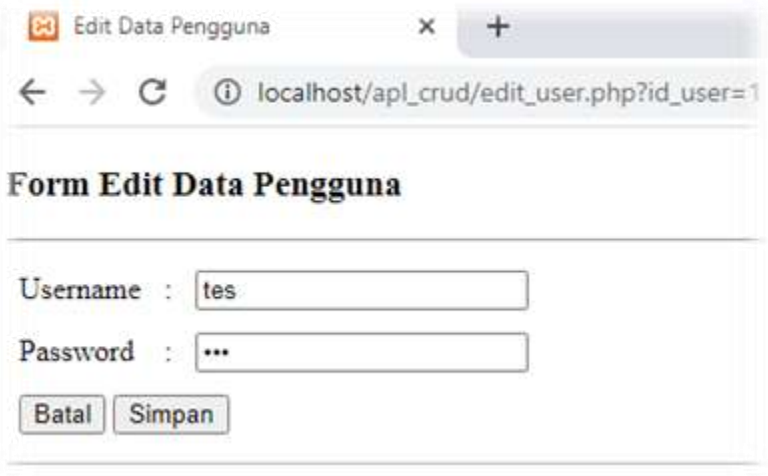
10.9 MENGUBAH DATA (*UPDATE*)

Ada kalanya sebuah data perlu untuk diubah untuk keperluan tertentu. Mengubah data dalam sebuah baris tabel memerlukan kunci utama (*primary key*) untuk mengubahnya. Kunci utama yang unik akan membuat data yang diubah benar. Tanpa menggunakan *primary key* bisa saja data yang berubah lebih dari satu baris.

```
<!DOCTYPE html>
<head>
   <title>Edit Data Pengguna</title>
</head>
<body>
<h3>Form Edit Data Pengguna</b></h3><hr>
<?php
error_reporting(0);
include "koneksi_db.php";
$sql="SELECT *  FROM tbl_user WHERE id_user=$_GET[id_user]";
$hasil=mysqli_query($koneksi_db,$sql);
$baris=mysqli_fetch_array($hasil);
?>
   <form method="POST" action="aksi_edit_user.php">
      <table cellpadding="5">
      <tr>
         <td>Username</td>
         <td>:</td>
         <td><input type="text" name="username" value="<?php echo
$baris['username'] ?>"></td>
      </tr>
      <tr>
         <td>Password</td>
         <td>:</td>
         <td><input type="password" name="password" value="<?php echo
$baris['username'] ?>"></td>
      </tr>
      <tr>
```

```
        <td colspan="3">
            <input type="button" value="Batal">
            <input type="submit" value="Simpan">
        </td>
    </tr>
    </table>
  </form>
<hr>
</body>
</html>
```

Pada kode program di atas, terlihat perintah SQL yang dikirim adalah SELECT dengan akhiran WHERE. Sehingga data yang tampil hanya sebaris saja. Kemudian data tersebut di tampilkan kembali dalam formulir untuk diubah. Hasil dari kode program di atas dapat dilihat pada Gambar 10.5.



Gambar 10. 5. Mengubah Data (*Update*)

Dapat dilihat *primary key* dari data yang akan diubah dikirim melalui URL. Dengan *primary key* tersebut data ditampilkan sebelum diubah.

```
<?php
include "koneksi_db.php";
$id_user=$_POST['id_user'];
$username=$_POST['username'];
$password=$_POST['password'];
$sql="UPDATE tbl_user SET username='$username',password='$password'
WHERE id_user='$id_user'";
mysqli_query($koneksi_db,$sql) or die (mysqli_error($koneksi_db));
header("Location:edit_user.php?pesan=berhasil_ubah");
?>
```

Kode di atas adalah aksi mengubah data dari inputan formulir sebelumnya. Data-data tersebut ditampung ke dalam variabel sebelum disisipkan ke perintah SQL UPDATE. Setelah data diubah, maka halaman otomatis kembali ke edit_user.php dengan membawa pesan berhasil.

10.10 MENGHAPUS DATA (*DELETE*)

Ketika sebuah data tidak diperlukan lagi maka data tersebut akan dihapus. Pada bagian sebelumnya sudah dibahas mengenai cara menghapus data menggunakan perintah SQL. Selanjutnya bagaimana menghapus data menggunakan aplikasi. Proses penghapusan data sendiri setengahnya sudah diselesaikan pada file index.php (membaca dan menampilkan data dari *database*. Perhatikan penggalan kode berikut ini.

```
<a href='aksi_hapus_user.php?id_user=$baris[id_user]' onclick='return
confirm_del()'>HAPUS</a>
```

Pada penggalan kode tersebut terlihat sebuah *link* dengan tulisan "HAPUS" yang diarahkan pada file aksi_hapus_user.php dengan aksi "*onclick*" yang berguna untuk konfirmasi sebelum dihapus. Data yang dibawa pada *link* menuju aksi_hapus_user.php adalah id_user. Dalam menghapus data id_user diperlukan sebagai *primary key* agar data yang terhapus tidak salah. Selanjutnya penyelesaian dari proses penghapusan data pada file aksi_hapus_user.php.

```
<?php
include "koneksi_db.php";
$sql="DELETE FROM tbl_user WHERE id_user='$_GET[id_user]'";
$hasil=mysqli_query($koneksi_db,$sql);
header("Location:index.php?pesan=berhasil_hapus");
?>
```

Pada kode program di atas, terlihat perintah SQL *DELETE* dengan akhiran *WHERE*. Berfungsi menghapus sebuah baris data berdasarkan *primary key* yaitu id_user.

11.1 INCLUDE

Include berfungsi untuk memasukkan isi sebuah file ke dalam bagian file lain. *Include* banyak di gunakan untuk menyatukan bagian-bagian *template* yang sengaja dibuat dalam file yang terpisah. Teknik ini dikenal dengan istilah teknik modularisasi. Pada kasus lain *include* digunakan untuk menggunakan sebuah file yang berisi pengaturan agar dapat digunakan pada banyak halaman, misalnya file koneksi *database*. Untuk contoh file koneksi yang selalu di *include* ke dalam sebuah file untuk berhubungan dengan database sudah dicontohkan pada bab sebelumnya.

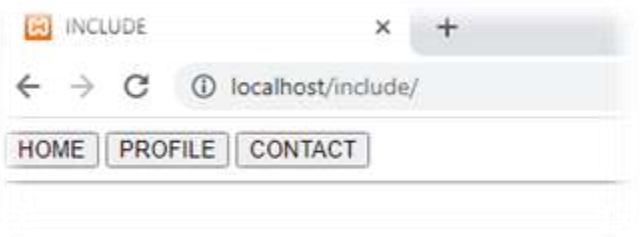
Berikut ini contoh file bernama index.php yang melakukan *include* file header.php dengan ini menu di dalamnya.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>INCLUDE</title>
</head>
<body>
   <?php
   include "header.php";
   ?>
</body>
</html>
```

Sebuah file bernama header.php adalah file yang menyimpan data menu aplikasi. Menu tersebut sengaja dipisahkan dari *index* agar modular. Berikut ini isi kode pada file header.php.

```
<button>HOME</button>
<button>PROFILE</button>
<button>CONTACT</button>
<hr>
```

Pada kode program di atas, terlihat isi file header.php. File tersebut hanya berisi tiga buah tombol dan sebuah garis. Tombol tersebut yakni *home*, *profile*, dan *contact*. Sedangkan garis yang dimaksud adalah garis horizontal. Hasil dari kode program di atas dapat dilihat pada Gambar 11.1.



Gambar 11. 1. *Include*

Pada tampilan *web browser* di atas terlihat sebuah menu dengan tombol *home*, *profile*, dan *contact* dan sebuah garis horizontal. Artinya file index.php berhasil meng-*include* file header.php. Sedangkan jika file yang di-*include* tidak ditemuakan, maka hanya muncul pesan *warning* dan kode lainnya tetap dibaca. Berikut ini contoh ketika *include* gagal dilakukan.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>INCLUDE</title>
</head>
<body>
   <?php
   include "headers.php";
   ?>
   <hr>
```

```
   <b>&copy Zen 2021</b>
</body>
</html>
```

Pada kode program di atas, terlihat nama file yang di-*include* salah. File tersebut seharusnya bernama header.php. Di bawah baris kode tersebut masih ada kode html yang akan tetap dijalankan walaupun terjadi kesalahan. Hasil dari kode program di atas dapat dilihat pada Gambar 11.2.

Gambar 11. 2. *Include* File Tidak Ditemukan

11.2 INCLUDE_ONCE

Hampir sama dengan *include*, *include_onces* hanya memanggil *file* yang pertama saja jika *file* yang di-*include*-kan sama dan lebih dari sekali.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>INCLUDE</title>
</head>
<body>
   <?php
   include_once "header.php";
   include_once "header.php";
   ?>
</body>
</html>
```

Pada contoh program di atas, terlihat kita mencoba meng-*include* sebuah *file* sebanyak dua kali. Jika menggunakan perintah *include* biasa maka *file* tersebut pasti berulang. Sengkan jika menggunakan *include_once*, *file* pertama saja yang di-*include*-kan. Hasil dari kode program di atas dapat dilihat pada Gambar 11.3.

Gambar 11. 3. Menggunakan *Include_Once*

Untuk memastikan perbedaan antara *include* dengan *include_one*. Perhatikanlah contoh kode program berikut ini.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>INCLUDE</title>
</head>
<body>
   <?php
   include "header.php";
   include "header.php";
   ?>
</body>
</html>
```

Pada kode program di atas, sebuah file di-*include* sebanyak dua kali. Maka file tersebut akan muncul dua kali. Hasil dari kode program di atas dapat dilihat pada Gambar 11.4.

Gambar 11. 4. Menggunakan *Include*

11.3 REQUIRE

Fungsi *require* sama dengan *include*, hanya saja jika file yang dimaksud tidak ditemukan maka program dihentikan. Artinya baris program selanjutnya tidak akan dibaca lagi.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>INCLUDE</title>
</head>
<body>
   <?php
   require "headers.php";
   ?>
   <hr>
   <b>&copy Zen 2021</b>
</body>
</html>
```

Pada kode program di atas, sebuah file di-*include* menggunakan *require*. Hasil dari kode program di atas dapat dilihat pada Gambar 11.4.

Gambar 11. 5. *Require* File Tidak Ditemukan

11.4 REQUIRE_ONCE

Require_one memiliki fungsi seperti perpaduan antara *include_one* dengan *require*. *Require_one* hanya mengijinkan satu file yang di panggil apabila banyak file yang sama dicoba untuk dipanggil. Jika file yang dimaksud tidak ditemukan maka akan terjadi fatal *erorr* dan program dihentikan.

12.1 *UPLOAD* FILE

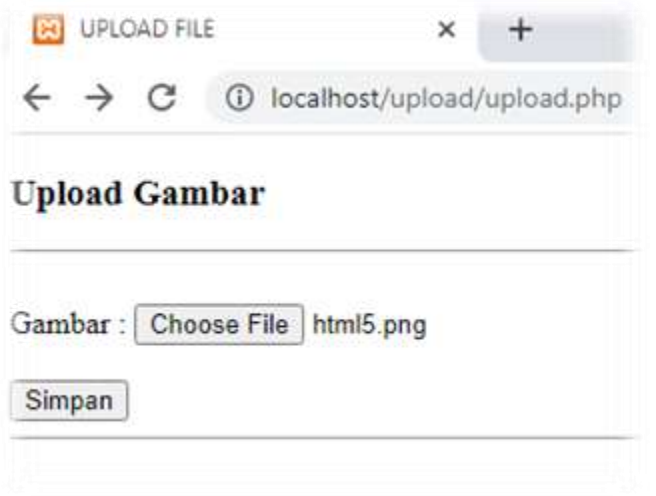
Upload file adalah memindahkan file dari komputer lokal ke dalam aplikasi web pada folder tertentu. Contoh yang sering di *upload* adalah gambar. Umumnya gambar tidak disimpan ke dalam *database* melainkan hanya namanya saja. Sedangkan file gambarnya ditempatkan pada suatu folder di aplikasi. Berikut ini adalah sebuah file berisi formulir untuk meng-*upload* sebuah gambar barnama upload.php.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>UPLOAD FILE</title>
</head>
<body>
   <h3>Upload Gambar</h3><hr><br>
   <form enctype="multipart/form-data" method="POST"
action="aksi_upload.php">
      Gambar : <input type="file" name="file">
      <br><br>
      <input type="submit" name="simpan" value="Simpan">
   </form>
   <hr>
</body>
</html>
```

Ada sedikit perbedaan dengan formulir yang berisi data teks. Formulir yang berisi data gambar harus memiliki atribut enctype. Untuk mencoba meng-*upload* sebuah file, sediakan sebuah folder bernama img_upload. Selanjutnya buat file aksi_upload.php dan ketik kode berikut ini.

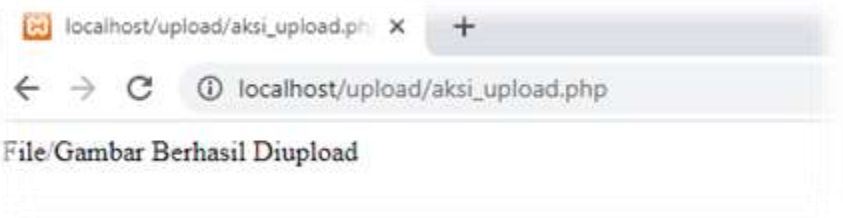
```
<?php
//cek ekstensi file
$nama_file=$_FILES['file']['name'];
$ijinkan = array('gif', 'png', 'jpg','jpeg');
$ekstensi = pathinfo($nama_file, PATHINFO_EXTENSION);
$status = 1;
if (!in_array($ekstensi, $ijinkan)) {
   $status = 0;
}
// cek ukuran file
if ($_FILES["file"]["size"] > 500000) {
   echo "Maaf, Ukuran file terlalu besar";
   $status = 0;
}

//upload file
if ($status==0) {
   echo "File gagal di upload";
}else{
   $target_file="img_upload";
   if (move_uploaded_file($_FILES["file"]["tmp_name"],
$target_file."/".$nama_file)) {
       echo "File/Gambar Berhasil Diupload";
     } else {
       echo "Terjadi kesalahan saat upload";
     }
}
```

Pada kode program di atas, dilakukan pengecekan format file dan ukuran file. Jika sesuai barulah file tersebut di-*upload*. Membatasi format file yang boleh di-*upload* adalah satu pencegahan terjadinya serangan keamanan aplikasi. Sedangkan membatasi ukuran file dapat menghemat penyimpanan *website*. Pada kode program di atas, sebuah file di-*include* sebanyak dua kali. Maka file tersebut akan muncul dua kali. Hasil dari kode program di atas dapat dilihat pada Gambar 12.1.



Gambar 12. 1. *Upload* File

Pada gambar di atas, dapat dilihat formulir sederhana untuk meng-*upload* gambar. Gambar yang akan di *upload* bernama html5.png. Berikut ini adalah respon dari aplikasi ketika berhasil di-*upload*.



Gambar 12. 2. *Upload* File Berhasil

12.2 PENGECEKAN FILE SEBELUM DI-*UPLOAD*

Pengecekan file sebelum diupload diantaranya adalah pengecekan format file dan pengecekan ukuran file.

12.2.1 Pengecekan Format File

Pengecekan format atau ekstensi file sangat penting dilakukan agar pengguna tidak sembarang mengunggah file. Jika tidak dibatasi format file apa saja yang di ijinkan, maka file berbahaya bisa masuk ke dalam sistem dan mungkin saja file tersebut bisa berjalan dan mempengaruhi sistem. Sehingga ijin beberapa batasan format file untuk di-*upload* perlu dilakukan.

12.2.2 Pengecekan Ukuran File

Jika ukuran file bebas untuk diupload, maka resiko yang timbul adalah kehabisan ruang penyimpanan yang kosong. Membatasi ukuran file yang dibolehkan untuk diupload adalah salah satu langkah yang tepat untuk menghemat ruang penyimpanan.

BAB 13
PENGENALAN CSS

13.1 PENGENALAN CSS

CSS adalah singkatan dari *Cascading Style Sheet* yang digunakan untuk mengatur tampilan halaman web agar lebih baik lagi dari sekedar menggunakan HTML. CSS bukanlah bahasa pemrograman yang dapat memproses logika. CSS digunakan untuk membantu memperindah tampilan sebuah halaman HTML dengan fitur-fitur yang disediakannya. CSS mampu mengubah jenis huruf, warna huruf maupun *background*, membentuk tampilan, sampai memberikan efek layaknya animasi.

CSS juga mampu membuat tampilan halamen web fleksibel terhadap layar pengguna. Dengan CSS tampilan dapat menyesuaikan dengan ukuran layar seperti, laptop, tablet, *smart phone* dan lain-lain. Dengan demikian mempelajari CSS dalam pemrograman web sangat penting untuk menghasilkan aplikasi yang baik secara keseluruhan.

13.2 BAGIAN-BAGIAN CSS

Sebelum menuliskan CSS ada baiknya mengetahui bagian-bagian dari CSS dan fungsinya. Bagian-bagian CSS yang umum yaitu *selector*, *property*, dan *value*.

Gambar 13. 1. Bagian-bagian CSS

13.2.1 Selector

Seperti artinya, *selector* berfungsi sebagai pemilih bagian HTML mana yang akan diberi *style* atau gaya. Beberapa selector yang sering digunakan antara lain selector nama tag, selector *class* dan selector *id*.

1. *Selector* Nama Tag

   Jika menamai selector dengan nama tag maka seluruh tag dengan nama tersebut dikenai gaya. Berikut ini contohnya:

```
h1{
color:red;
}
```

Semua elemen HTML yang menggunakan tag h1 akan berwarna merah hurufnya.

2. *Selector Class*

   Selector yang satu ini paling banyak dijumpai dan digunakan orang. Pemberian gaya tergantung pada pemanggilan atribut *class*. Artinya jika *class* tersebut dipanggil maka gaya diberikan. Penulisan *selector class* ditandai dengan awalan titik ".". Berikut ini contoh kode CSS nya.

```
.tebal{
        Font-weight:bold;
}
```

Jika pada *selector* nama *tag* tidak perlu lagi memanggil untuk mendapatkan gayanya, lain halnya dengan selector *class* ini. Elemen HTML perlu memanggil dengan menggunakan atribut *class* di *tag* buka HTML. Kemudian menyebutkan *value* yaitu nama *class* pada CSS, namun tidak diikuti tanda titik ".". Berikut ini contoh kode HTML yang memanggil atau menggunakan class

```
<span class=”tebal”> Penting <span>
```

Span adalah *tag* HTML yang tidak memberi efek jika tidak diberi gaya. Jika *tag* HTML memanggil *class* tebal dibagian mana saja tidak perduli *tag* apa saja, maka akan diberi efek tulisan tebal.

3. *Selector Id*

Cara kerja *selector* ini hampir sama dengan selector *class*, bedanya menggunakan penanda pagar "#" di awal nama *selector*. Sedangkan atribut untuk memanggilnya adalah *id* dengan *value* nama *selector id* tanpa diikuti tanda pagar. Berikut ini contoh penulisan kode CSS dengan *selector id*.

```
#merah{
        color:red;
}
```

Selanjutnya gaya tersebut dipanggil atau digunakan pada elemen HTML. Berikut contoh kode HTMLnya.

```
<span id="merah"> Sangat Penting </span>
```

Setiap elemen HTML yang memanggil style ini akan diberi gaya sesuai dengan isi *style*-nya. Namun atribut ini sering digunakan oleh *javascript*.

13.2.2 *Propery* dan *Value*

Property bisa dikatakan sebagai jenis gaya yang akan digunakan. Gaya atau *property* dalam sebuah *selector* dapat berjumlah satu atau lebih. Setiap *property* harus di isi *value* atau nilainya dengan pemisah titik dua antara *propery* dan value ":". Setiap gaya atau *property* dan *value* hendaknya diakhiri titik koma ";". Berikut ini contoh kode CSS nya menggunakan *selector class*.

```
.info {
        background-color:cyan;
        color:white;
}
```

Contoh kode CSS di atas terdiri dari dua properti beserta nilainya. Kedua *property* tersebut harus diakhiri titik koma (;).

13.3 CARA PENULISAN CSS

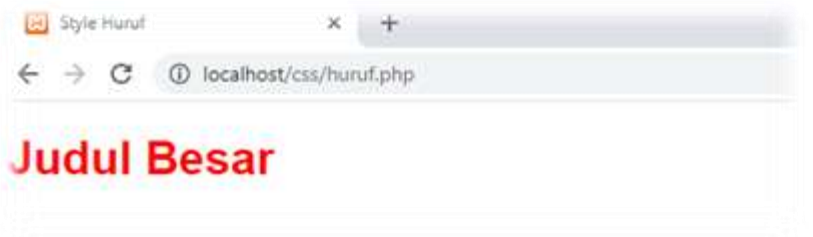
Ada tiga cara penulisan CSS yaitu: *inline* CSS, internal CSS, eksternal CSS. Masing masing cara penulisan ini memiliki kelebihan dan kekurangan.

13.3.1 *Inline* CSS

Penulisan *inline* CSS ini langsung pada kode HTML (*inline*) tepatnya pada *tag* buka HTML menggunakan atribut bernama *style*. Nilai dari atribut *style* tersebut langsung diisi *property* dan *value* sesuai dengan gaya yang diinginkan. Untuk lebih jelasnya perhatikan contoh kode berikut ini.

```
<html>
<head>
   <title>Style Huruf</title>
   <link rel="stylesheet" href="style1.css">
</head>
<body>
   <h1 style="color:red; font-family:Arial, Helvetica, sans-serif;">Judul Besar</h1>
</body>
</html>
```

Pada contoh kode CSS di atas, *tag* <h1> diberi atribut bernama *style* yang berisi kode CSS. Setiap *property* dan *value* CSS diakhiri tanda titik koma. Hasil dari kode program di atas dapat dilihat pada Gambar 13.2.



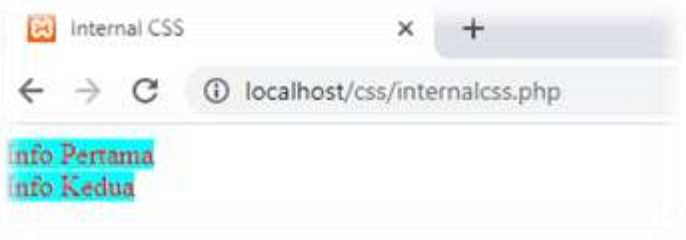
Gambar 13. 2. *Inline* CSS

13.3.2 Internal CSS

Hampir mirip dengan *inline* CSS yaitu dituliskan pada file yang sama dengan kode HTML, bedanya pada internal CSS sudah digunkan *selector*. Artinya kode CSS dapat dipakai berkali-kali di elemen HTML dalam satu file tersebut, sehingga kode CSS menjadi lebih efisien.

```
<html>
<head>
   <title>Internal CSS</title>
</head>
<style>
   .info{
      background-color: aqua;
      color: red;
   }
</style>
<body>
   <span class="info">Info Pertama</span><br>
   <span class="info">Info Kedua</span>
</body>
</html>
```

Pada contoh kode di atas, terlihat CSS didefenisikan di dalam *tag* <head>. Selanjutnya, untuk menggunakan gaya dari CSS tersebut, kita perlu menggunakan atribut *class* dengan *value* sesuai dengan gaya yang diinginkan. Hasil dari kode di atas dapat dilihat pada Gambar 13.3.



Gambar 13. 3. Internal CSS

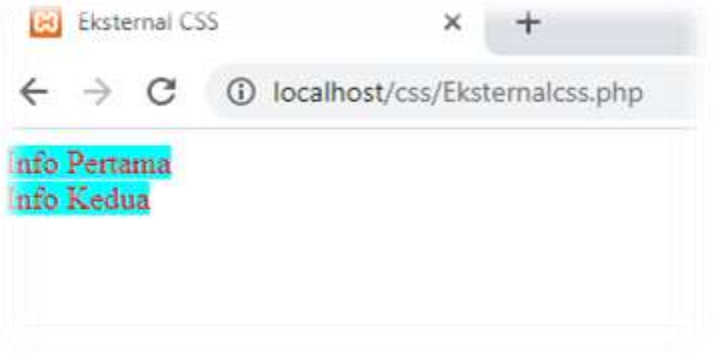
13.3.3 Eksternal CSS

Berbeda dengan *inline* CSS dan internal CSS yang kode HTML nya satu file dengan CSS. Eksternal css menempatkan kode CSS berbeda file dengan kode HTML nya. Manfaatnya kode menjadi bersih karna tidak bercampur dan file CSS dapat digunakan pada file manapun.

```
/* nama file: style1.css*/
.info{
   background-color: aqua;
   color: red;
}
```

```
<!-- nama file : eksternal.css !-->
<html>
<head>
   <title>Eksternal CSS</title>
</head>
<link rel="stylesheet" href="style1.css">
<body>
   <span class="info">Info Pertama</span><br>
   <span class="info">Info Kedua</span>
</body>
</html>
```

Tag HTML yang digunakan untuk menghubungkan CSS adalah tag <link>. Jika terhubung maka gaya-gaya (*style*) yang ada di dalam file tersebut dapat digunakan. Kelebihan dari penulisan CSS secara eksternal ini adalah file CSS dapat dipanggil lagi oleh file-file HTML lain. Hasil dari kode di atas dapat dilihat pada Gambar 13.4.



Gambar 13. 4. Eksternal CSS

13.4 *PARENT CHILD* DAN *DECENDANT*

Di dalam penulisan *selector* pada CSS dikenal dengan istilah *parent*, *child* dan *decendant*. *Parent* artinya orang tua, *child* artinya anak dan *decendant* artinya keturunan. Memberi gaya pada elemen HTML cukup memanggil *class* nya atau *id* nya. Namun bagaimana jika yang kita maksudkan adalah elemen yang lebih spesifik berdasarkan strukturnya. Maka penulisan *selector* yang menggunakan konsep *parent*, *child* dan *decendant* ini yang bisa jadi solusinya.

13.4.1 Parent Child

Konsep orang tua ke anank yaitu memberikan *style* tepat pada *child* nya saja. Cara penulisan kodenya menggunakan tanda lebih besar ">". Untuk lebih jelasnya perhatikan contoh kode CSS dan HTML berikut.

```
.ortu>.anak{
   background-color: aqua;
}

.anak{
   color: red;
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Parent Child CSS</title>
   <link rel="stylesheet" href="parent_child.css">
</head>
<body>
   <div class="ortu">
      <div class="anak">Dikenakan Style Parent Child</div>
   </div>
   <div class="ortu">
      <div class="ayah">
         <div class="anak"> Tidak Dikenakan Style </div>
      </div>
   </div>
</body>
</html>
```

Pada kode di atas, setiap elemen HTML yang memeiliki atribut kelas anak akan diberi warna teks merah. Sedangkan *background* berwarna *cyan* hanya diberikan pada kelas anak yang kelas *parent* bernama ortu. Kelas anak yang kedua tidak dikenakan *background* berwarna *cyan* karena kelas *parent* miliknya adalah ayah. Hasil dari kode di atas dapat dilihat pada Gambar 13.5.

Gambar 13. 5. *Parent Child* CSS

Untuk *class* anak yang kedua tidak dikenakan *style* karena *parent* nya tidak langsung ke *class* ortu.

13.5 PARENT DECENDANT

Hampir sama dengan *parent child*, namun pada *parent decendant* beraku *style* untuk semua keturuan dari *parent* nya. Cara penulisan kodenya menggunakan spasi " ". Untuk lebih jelasnya perhatikan kode CSS dan HTML berikut.

```
.ortu>.anak{
   background-color: aqua;
}

.anak{
   color: red;
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Parent Decendant CSS</title>
   <link rel="stylesheet" href="parent_decendant.css">
</head>
<body>
   <div class="ortu">
      <div class="anak">Dikenakan Style Parent Decendant
```

```
            </div>
    </div>
    <div class="ortu">
        <div class="ayah">
            <div class="anak">Dikenakan Style Parent Decendant
                </div>
        </div>
    </div>
</body>
</html>
```

Pada kode di atas dapat dilihat kelas anak dan ayah merupakan keturunan kelas ortu. Hasil dari kode di atas dapat dilihat pada Gambar 13.6.

Gambar 13. 6. *Parent Decendant* CSS

Dapat dilihat hasilnya pada gambar di atas, pada konsep *parent decendant*, *class* anak yang ke dua mendapat *style* karena merupakan *class* yang berada di dalam turunan *class* ortu.

14.1 MEWARNAI TEKS

Mengatur warna teks sangat diperlukan untuk menyelaraskan suatu tampilan halaman web. Mewarnai teks umumya menggunakan CSS, walaupun demikinan menggunakan *tag* dan atribut tertentu pada HTML juga bisa. *Tag* yang dapat mengubah warna teks adalah tag <font>. Namun menggunakan CSS menjadikan kode lebih efisien. Ada beberapa jenis pewarnaan pada teks yaitu *solid* dan *gradient*.

```
/* nama file: mewarnai_teks.css*/
.solid{
   color: gold;
}
.gradien {
   font-size: 72px;
   background: -webkit-linear-gradient(rgb(25, 146, 226), rgb(194, 18, 18));
   -webkit-background-clip: text;
   -webkit-text-fill-color: transparent;
  }
```

```
<!DOCTYPE html>
<head>
   <title>Mewarnai Teks</title>
   <link rel="stylesheet" href="mewarnai_teks.css">
</head>
<body>
   <div class="solid">Warna Solid</div>
   <div class="gradien">Warna Gradien</h1>
</body>
</html>
```

Untuk mewainai teks dengan warna solid sangat mudah, cukup menggunakan properti "*color*". Namun untuk membuat warna gradien membutuhkan trik khusus. Hasil dari kode di atas dapat dilihat pada Gambar 14.1.

Gambar 14. 1. Mewarnai Teks

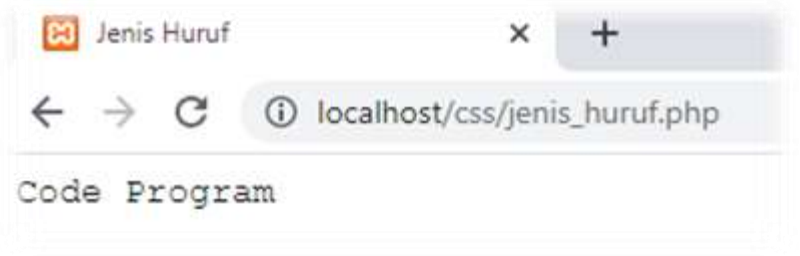
14.2 JENIS HURUF

Memilih jenis huruf yang tepat dapat membuat tampilan desain halaman web menjadi bagus. Setiap jenis huruf memiliki arti tersendiri yang dapat menyampaikan pesan tertentu kepada pembacanya. *Property* yang digunakan adalah "*font-family*". Berikut ini cara mengubah jenis huruf menggunakan CSS.

```
<!DOCTYPE html>
<head>
   <title>Jenis Huruf</title>
   <link rel="stylesheet" href="jenis_huruf.css">
</head>
<body>
   <div class="jenis_huruf1">Code Program</div>
</body>
</html>
```

```
.jenis_huruf1{
   font-family:'Courier New', Courier, monospace;
}
```

Pada kode di atas dapat dilihat *property* yang digunakan adalah "*font-family*" dan nilai yang diberikan adalah Courier New. Hasil dari kode program di atas dapat dilihat pada Gambar 14.2.



Gambar 14. 2. Jenis Huruf

Pada contoh kode di atas, jenis huruf yang dipilih adalah *Courier New*, *Courier*, atau *Monospace*. Sistem akan memilihkan jenis huruf berdasarkan ketersediaan huruf tersebut dikomputer.

DAFTAR PUSTAKA

Cara Memotong atau Mengambil Sebagian String (fungsi substr). (n.d.). Retrieved November 10, 2022, from duniailkom.com/tutorial-php-cara-memotong-atau-mengambil-sebagian-string-fungsi-substr/

CSS Gradient Text. (n.d.). Retrieved November 10, 2022, from https://cssgradient.io/blog/css-gradient-text/

PHP. (n.d.). Retrieved November 10, 2022, from https://www.php.net/

PHP Data Types. (n.d.). Retrieved November 10, 2022, from https://www.w3schools.com/php/php_datatypes.asp

Solichin, A., & Kom, S. (2005). *Pemrograman Web dengan PHP dan MySQL*.

Tagger, B., & Mulia. (2014). *Modul Pemrograman Web. Dimulai dari penggunaan HTML sampai Javascript* (p. 117).

BIORGRAFI PENULIS

Muhammad Zen, S.T., M. Kom merupakan Dosen di Universitas Pembangunan Panca Budi Medan. Mengajar di Program Studi Sistem Komputer. Sejak karirnya sebagai dosen, Zen tertarik mendalami bidang pemrograman web, *text processing* dan keamanan data. Muhammad Zen lahir di Bukit Gantung, Kec. Wampu, Kab. Langkat pada tanggal 07 Mei 1992.

Dr. Sayuti Rahman merupakan Dosen di Universitas Harapan Medan. Studi S3 di Universitas Syiah Kuala, Banda Aceh. Sejak karirnya sebagai dosen, Sayuti tertarik dengan mendalami bidang citra digital. Sehingga saat ini Sayuti fokus dalam penelitian yang berkaitan dengan mengekstraksi informasi didalam citra digital yaitu Visi komputer. Penelitian yang sedang dikerjakan saat ini adalah membangun sistem parkir cerdas dengan mongoptimalkan CNN dalam mengklasifikasi ruang parkir. Sayuti Rahman lahir di Bukit gantung, Kec. Wampu, Kab. Langkat pada tanggal 18 juni 1987.

Haida Dafitri, S.T., M.Kom lahir di Medan pada tanggal 17 Mei 1988. Penulis memperoleh gelar Sarjana Teknik di Sekolah Tinggi Teknik Harapan pada Program Studi Teknik Informatika, Maret 2010. Kemudian Juli 2011, Penulis melanjutkan studi Magister di Universitas Sumatera Utara, Program Studi Teknik Informatika Fakultas Ilmu Komputer dan Teknologi Informasi Hingga lulus pada September 2013. Dari Tahun 2011 hingga saat ini, Penulis adalah Dosen di Program Studi Teknik Informatika Fakultas Teknik dan Komputer di Universitas Harapan Medan. Menjadi Dosen Pembimbing Lapangan pada Program Kampus Mengajar Angkatan 3 di Tahun 2021 menjadi suatu pengalaman menarik bagi penulis dalam mendampingi dan membimbing koordinasi mahasiswa dengan Dinas Pendidikan Kabupaten/Kota dan Satuan Pendidikan atau sekolah sasaran, selain itu penulis dapat meningkatkan kemampuan dalam mengembangkan ide-ide yang inovatif dalam pembelajaran. Penulis memiliki ketertarikan penelitian di bidang Pemrosesan Informasi Multimedia, Augmented dan Virtual Reality, Pemrosesan Informasi Dalam Belajar, Kecerdasan Buatan, Data Mining, Sistem Pendukung Keputusan.

Risko Liza, S.T., M.Kom lahir di Medan, 09 November 1978 adalah Dosen yang mengajar di salah satu Universitas di Medan. Saat ini Penulis mengajar di Program Studi Teknik Informatika Fakultas Teknik dan Komputer, Universitas Harapan Medan. Mata Kuliah yang diampu termasuk pada peminatan Jaringan, Multimedia dan Mikrokontroller (Robotics). Penulis lebih suka pada hal hal baru pada teknologi baru apalagi berbau tantangan. Penulis juga sedang melakukan riset mandiri yang berhubungan dengan teknologi terbarukan dengan kolaborasi antar Program studi.

Rachmat Aulia, S.Kom., M.Sc.IT. adalah seorang dosen di Perguruan Tinggi Swasta bernama Universitas Harapan Medan. Lahir di Medan tanggal 27 Desember 1983. Beliau menyelesaikan pendidikan S1 tahun 2006 di Universitas Budi Luhur dan pendidikan S2 tahun 2008 di University Utara Malaysia. Karir dosennya dimulai tahun 2008 dengan menjadi dosen tetap di Program Studi Teknik Informatika Sekolah Tinggi Teknik Harapan (saat ini berubah menjadi Universitas Harapan Medan). Beliau telah menghasilkan publikasi karya ilmiah dalam bentuk jurnal (nasional terakreditasi, ber-ISSN, dan internasional) dan conference (seminar nasional dan internasional) hingga saat ini. Beberapa jurnal dan conference yang dihasilkan berkaitan dengan database, data mining, kriptografi, dan robotics. Saat ini, beliau sedang mendalami bidang data mining yang mengarah ke pendidikan selanjutnya. Selain itu, beliau mahir membuat program menggunakan bahasa program web baik secara manual maupun dengan bantuan tools.

Nurjamiyah, S.Kom., M.Cs. merupakan dosen tetap pada Program Studi Sistem Informasi Universitas Harapan Medan sejak tahun 2014, seorang wanita kelahiran kota Pematang Siantar pada tanggal 2 Agustus 1982. Beliau meraih gelar Masters of Computer Science (M.Cs.) dari Universitas Gadjah Mada Yogyakarta pada tahun 2012. Selain sebagai dosen, sejak tahun 2018 Beliau menjadi seorang Entrepreneur di bidang jasa Laundry, sehingga sampai saat ini diberikan kewenangan untuk mengajar mata kuliah yang berhubungan dengan Kewirausahaan dan E-Business, selain mata kuliah yang berhubungan dengan Sistem Informasi.

Pemrograman web merupakan salah satu pemrograman dalam bidang komputer yang berkembang pesat seiring kemajuan internet. Sehingga banyak dari masyarakat umum maupun pelajar atau mahasiswa mempelajarinya. Buku ini berisi materi berkaitan dengan algoritma seperti kondisi, perulangan dan array. Buku ini juga memperkenalkan beberapa bahasa seperti HTML, CSS, PHP dan SQL kepada pemula.

Banyak kode yang perlu dipelajari untuk membangun sebuah aplikasi berbasis web, namun semua itu biasanya tidak dikerjaan seorang diri. Apalagi banyak kerangka kerja yang memudahkan kita mengembangkan aplikasi. Dalam pembuatan alikasi berbasis web, proses pekerjaan minimal terbagi dua yaitu tampilan dan logika atau program. Untuk yang ingin belajar membuat tampilan dapat memperdalam keahlian desain tampilan halaman web yaitu mempelajari HTML dan CSS. Sedangkan yang ingin mempelajari logika atau program dapat memperdalam PHP dan basis data.

CV. Tahta Media Group
Surakarta, Jawa Tengah
Web : www.tahtamedia.com
Ig : tahtamediagroup
Telp/WA : +62 813 5346 4169

ISBN 978-623-8070-19-0