DASAR PEMROGRAMAN WEB MENGGUNAKAN HTML, CSS, XML, DAN JAVASCRIPT

Penulis: Noor Ifada

Editor: Husni

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA
TAHUN 2018

DASAR PEMROGRAMAN WEB MENGGUNAKAN HTML, CSS, XML, DAN JAVASCRIPT

© Noor Ifada, 2018

Penulis
Noor Ifada

Editor
Husni

Desain Cover & Penata Isi
Tim MNC Publishing

Cetakan I, Desember 2018

Diterbitkan oleh :

Media Nusa Creative
Anggota IKAPI (162/JTI/2015)
Bukit Cemara Tidar H5 No. 34, Malang
Telp. : 0812.3334.0088
E-mail : mncpublishing.layout@gmail.com
Website : www.mncpublishing.com

ISBN : 978-602-462-128-5

Hak Cipta dilindungi undang-undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ke dalam bentuk apapun, secara elektronis maupun mekanis, termasuk fotokopi, merekam, atau dengan teknik perekaman lainnya, tanpa izin tertulis dari Penerbit. Undang-Undang Nomor 19 Tahun 2000 tentang Hak Cipta, Bab XII Ketentuan Pidana, Pasal 72, Ayat (1), (2), dan (6)

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT, karena atas ijin-Nya penulis dapat menyelesaikan Buku Ajar "Dasar Pemrograman Web menggunakan HTML, CSS, XML, dan JavaScript" ini. Tujuan utama penulisan buku ini adalah untuk memberikan pengetahuan dan panduan bagi mahasiswa mengenai dasar pemrograman *web* menggunakan HTML, CSS, XML, dan JavaScript. Buku ini ditulis sebagai bahan ajar untuk mata kuliah Dasar Pemrograman *Web* yang merupakan mata kuliah dasar dan wajib di Prodi Teknik Informatika. Kompetensi yang diharapkan di akhir perkuliahan adalah bahwa mahasiswa memiliki keahlian untuk mengembangkan *web* statis dengan menggunakan kombinasi HTML, CSS dan JavaScript sebagai bahasa standarnya. Sebagai tambahan, mahasiswa juga dapat membuat dokumen XML yang memiliki berkaitan erat dengan HTML dan berperan penting dalam berbagai sistem IT.

Akhir kata, penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang secara langsung ataupun tidak langsung telah membantu dalam proses penulisan buku ini. Penulis juga mengharapkan masukan dan kritik dari pembaca demi perbaikan buku ini ke depannya.

Penulis,

Noor Ifada

DAFTAR ISI

KATA PENGANTAR ..... **iii**
DAFTAR ISI ..... **v**
DAFTAR GAMBAR ..... **ix**
DAFTAR TABEL ..... **xiii**

BAB 1 KONSEP DASAR PEMROGRAMAN WEB **1**
1.1 Web dan Internet 1
1.2 Web Server dan Client 2
1.3 HTTP (Hypertext Transfer Protocol) 3
1.4 Situs Web (Website) 4
1.4.1 Struktur Situs Web (Website) 3
1.4.2 Jenis Situs Web 4
1.4.3 URL Situs Web 5
1.5 Persiapan Pembuatan Web 6
1.5.1 Bahasa 6
1.5.2 Aplikasi 6
1.6 Latihan Soal 8

BAB 2 HTML **9**
2.1 Dasar HTML 9
2.1.1 Struktur dan Sintaks HTML 10
2.1.2 Komentar dalam HTML 13
2.2 Format Teks dalam HTML 13
2.2.1 Heading 13
2.2.2 Paragraph 14
2.2.3 Blockquote 15
2.2.4 Font 15
2.2.5 Font Style 15
2.3 Baris dan Garis dalam HTML 17
2.3.1 Break 17

2.3.2 Garis Horisontal 17
2.4 Image dan Link dalam HTML 17
2.4.1 Image 18
2.4.2 Link 19
2.5 List dalam HTML 19
2.5.1 Unordered list 19
2.5.2 Ordered list 19
2.5.3 Definition List 20
2.6 Tabel dalam HTML 21
2.6.1 Struktur Tabel 21
2.6.2 Judul pada Tabel 22
2.6.3 Penggabungan Sel Kolom 24
2.6.4 Penggabungan Sel Baris 25
2.6.5 Format Header, Body dan Footer 26
2.6.6 Mengatur lebar dan tinggi tabel 27
2.6.7 Membuat warna pada table 27
2.6.8 Cellpading dan cellspacing 28
2.7 Form dalam HTML 29
2.7.1 Struktur Form 29
2.7.2 Pengelompokkan Form 31
2.7.3 Elemen <input> 31
2.7.4 Elemen <select> dan <datalist> 35
2.7.5 Elemen < textarea> 37
2.8 Latihan Soal 37

BAB 3 CSS 41
3.1 Dasar CSS 41
3.1.1 Struktur dan Sintaks CSS 42
3.1.2 Komentar dalam CSS 43
3.2 Selector 43
3.2.1 Selector Elemen HTML 43
3.2.2 CSS di dalam elemen <head> 44
3.2.3 Selector ID 45
3.2.4 Pengelompokan Selector 46

3.3 Lokasi Penempatan CSS 47
3.3.1 CSS di dalam elemen HTML 47
3.3.2 CSS di dalam elemen <head> 48
3.3.3 CSS di dalam file eksternal 48
3.4 The Box model 49
3.5 CSS untuk Layout Halaman Web 51
3.5.1 Properti float dan clear untuk Pengaturan Posisi 52
3.5.2 Contoh Layout Halaman Web 54
3.6 Latihan Soal 57

BAB 4 XML 59
4.1 Dasar XML 59
4.1.1 Struktur dan Sintaks XML 61
4.1.2 Komentar dalam XML 63
4.2 Memformat Tampilan XML 63
4.2.1 Memformat Tampilan dengan CSS 65
4.2.2 Memformat Tampilan dengan XSLT 66
4.3 Latihan Soal 68

BAB 5 JAVASCRIPT 69
5.1 Dasar JavaScript 70
5.1.1 Struktur dan Sintaks JavaScript 70
5.1.2 Komentar dalam JavaScript 71
5.2 Lokasi Penempatan JavaScript 71
5.2.1 JavaScript di dalam Elemen <body> 71
5.2.2 JavaScript di dalam Elemen <head> 71
5.2.3 Penempatan JavaScript di dalam File Eksternal 72
5.3 Variabel dan Tipe Data 73
5.4 Operator 74
5.4.1 Operator Aritmatika (Arithmetic) 74
5.4.2 Operator Penugasan (Assignment) 75
5.4.3 Operator Logika (Logical) 75

5.4.4 Operator Perbandingan 76
5.4.5 Operator Bitwise 77
5.4.6 Operator String 78
5.5 Struktur Kondisi 79
5.5.1 Kondisi IF 79
5.5.2 Kondisi SWITCH 79
5.6 Struktur Perulangan 80
5.6.1 Perulangan FOR 80
5.6.2 Perulangan WHILE 81
5.6.3 Perulangan Tidak Terhingga (Infinite) 82
5.7 Fungsi (Function) 82
5.7.1 Deklarasi Fungsi 83
5.7.2 Memanggil Fungsi 83
5.8 Built-in Objects 84
5.8.1 String Object 84
5.8.2 Array Object 84
5.8.3 Math Object 85
5.8.4 Date Object 85
5.9 RegExp 86
5.10 HTML DOM 90
5.11 Mengakses Elemen HTML 91
5.12 Kotak Dialog 92
5.13 HTML Event 93
5.14 Validasi Form 94
5.15 Latihan Soal 96

INDEKS 97

DAFTAR PUSTAKA 99

LAMPIRAN

Jawaban Soal Latihan 101

BIOGRAFI PENULIS 115

BIOGRAFI EDITOR 116

DAFTAR GAMBAR

Gambar 1.1. *Web* dan *internet* 2
Gambar 1.2. *Web Server* dan *Client* 3
Gambar 1.3. Struktur *situs web* 4
Gambar 1.4. Ilustrasi URL 5
Gambar 1.5. *Google Chrome Developer Tools* 7
Gambar 2.1. Ilustrasi *tag*, elemen dan atribut dalam HTML 11
Gambar 2.2. Contoh HTML sederhana 12
Gambar 2.3. Contoh *heading* 13
Gambar 2.4. Contoh *paragraph* 14
Gambar 2.5. Contoh *blockquote* 15
Gambar 2.6. Contoh *font* 15
Gambar 2.7. Contoh *font type* 16
Gambar 2.8. Contoh perbandingan *list* 20
Gambar 2.9 . Elemen utama tabel 21
Gambar 2.10. Contoh tabel sederhana 22
Gambar 2.11. Contoh tabel, judul tabel, dan judul kolom 23
Gambar 2.12. Contoh tabel, judul tabel, dan judul baris 24
Gambar 2.13. Contoh penggabungan sel kolom tabel 25
Gambar 2.14. Contoh penggabungan sel baris tabel 25
Gambar 2.15. Contoh format *header*, *body* dan *footer* pada tabel . 26
Gambar 2.16. Contoh pengaturan lebar dan tinggi tabel 27
Gambar 2.17. Contoh membuat warna pada tabel 28
Gambar 2.18. Contoh cellpadding dan cellspacing 29
Gambar 2.19. Struktur *form* 30
Gambar 2.20. Contoh *form* sederhana 30
Gambar 2.21. Contoh *form* dengan elemen <input> 35
Gambar 2.22. Contoh perbandingan elemen <select> dan <datalist> dalam *form* 36
Gambar 2.23. Contoh form dengan elemen <textarea> 37
Gambar 2.24. Tampilan *list* bersarang 37
Gambar 2.25. Tampilan memuat gambar dan penggunaan *link*. 38

Gambar 2.26. Tabel dengan gabungan judul baris dan kolom ... 38
Gambar 2.27. Tampilan tabel yang memiliki gabungan kolom dan baris ... 39
Gambar 2.28. Contoh *layout* tampilan *form* pada Gambar 2.28 dengan menggunakan tabel ... 39
Gambar 3.1. Struktur dalam sintaks CSS ... 42
Gambar 3.2. Contoh CSS sederhana dalam dokumen HTML ... 42
Gambar 3.3. Contoh *selector* elemen HTML dalam CSS ... 44
Gambar 3.4. Contoh *selector* class dalam CSS ... 45
Gambar 3.5. Contoh *selector* ID dalam CSS ... 46
Gambar 3.6. Contoh pengelompokkan *selector* ... 47
Gambar 3.7. Contoh penempatan CSS di dalam elemen HTML 47
Gambar 3.8. Contoh penempatan CSS di dalam elemen <head> ... 48
Gambar 3.9. Contoh penempatan CSS sebagai file eksternal ... 49
Gambar 3.10. Skrip CSS (style.css) yang digunakan dalam Gambar 3.9 ... 49
Gambar 3.11. Komponen dalam The Box model ... 50
Gambar 3.12. Contoh skrip HTML untuk properti Box ... 51
Gambar 3.13. Contoh properti float dalam dokumen HTML ... 53
Gambar 3.14. Skrip CSS (style_div.css) yang digunakan dalam Gambar 3.13 ... 54
Gambar 3.15. Contoh *layout* halaman *web* ... 55
Gambar 3.16. Skrip CSS (style_layout.css) yang digunakan dalam Gambar 3.15 ... 56
Gambar 3.17. *Layout* baru halaman *web* untuk Gambar 3.15 ... 57
Gambar 4.1. Pengembangan bahasa markah berbasis SGML (Krause 2016) ... 60
Gambar 4.2. Struktur *tree* pada XML ... 61
Gambar 4.3. Contoh XML sederhana ... 61
Gambar 4.4. Contoh aturan sintaks dalam dokumen XML ... 62
Gambar 4.5. Contoh aturan penamaan elemen dalam dokumen XML ... 63

Gambar 4.6. Contoh tampilan dokumen XML (Gambar 4.3 tanpa komentar) pada *web browser* 64
Gambar 4.7. Contoh tampilan dokumen XML yang memiliki kesalahan pada *web browser* 64
Gambar 4.8. Contoh penggunaan CSS untuk format tampilan dokumen XML 65
Gambar 4.9. Skrip CSS (xml_style.css) yang digunakan dalam Gambar 4.8 65
Gambar 4.10. Contoh penggunaan XSL untuk format tampilan dokumen XML 66
Gambar 4.11. Skrip XSL (xml_style.xsl) yang digunakan dalam Gambar 4.10 67
Gambar 4.12. Skrip XMLdengan empat kesalahan sintaks 68
Gambar 4.13. Tampilan baru skrip dalam Gambar 4.3 pada *web browser* 68
Gambar 5.1. Contoh JavaScript sederhana 70
Gambar 5.2. Contoh penempatan JavaScript di dalam tag <head> 72
Gambar 5.3. Contoh link penempatan JavaScript sebagai file eksternal 72
Gambar 5.4. Skrip pada *file* eksternal.js yang digunakan dalam Gambar 5.3 72
Gambar 5.5. Contoh kondisi IF 79
Gambar 5.6. Contoh kondisi SWITCH 80
Gambar 5.7. Contoh perulangan *FOR* 80
Gambar 5.8. Contoh perulangan WHILE 81
Gambar 5.9. Contoh perulangan DO-WHILE 81
Gambar 5.10. Contoh perintah *break* 82
Gambar 5.11. Contoh pembuatan dan penggunaan fungsi 83
Gambar 5.12. Contoh implementasi *method* dalam *String Object* 84
Gambar 5.13. Contoh implementasi *method* dalam *Array Object* 85
Gambar 5.14. Contoh implementasi *method* dalam *Math Object* . 85
Gambar 5.15. Contoh implementasi *method* dalam *Date Object* .. 86

Gambar 5.16. Contoh fungsi dengan RegExp untuk validasi tanggal 90
Gambar 5.17. Contoh implementasi *method* dalam RegExp *Object* 90
Gambar 5.18. Skrip HTML untuk contoh pembuatan DOM tree 91
Gambar 5.19. Struktur DOM *tree* dari dokumen HTML pada Gambar 5.18 91
Gambar 5.20. Contoh akses elemen HTML dengan *node* pada DOM *tree* atau *method getElementById* 92
Gambar 5.21. Contoh pembuatan kotak dialog *alert*, *prompt* dan *confirm* 93
Gambar 5.22. Contoh pengecekan isian dan tampilan pesan kesalahan untuk validasi *form* 95
Gambar 5.23. Contoh pengecekan format angka bilangan bulat dan tampilan pesan kesalahan untuk validasi *form* 96

DAFTAR TABEL

Tabel 2.1. Sejarah perkembangan versi HTML (Krause 2016) 10
Tabel 2.2. Daftar *font style* dalam HTML 16
Tabel 2.3. Contoh perbandingan *file path* 18
Tabel 2.4. Penulisan *link* 19
Tabel 2.5. Atribut type dalam elemen <input> 32
Tabel 2.6. Atribut selain type dalam elemen <input> 33
Tabel 3.1. Metode pembuatan *layout* halaman *web* 52
Tabel 3.2. Daftar properti float dan clear untuk pengaturan posisi 52
Tabel 5.1. Contoh deklarasi variabel dalam JavaScript 74
Tabel 5.2. Daftar Operator Aritmatika dalam JavaScript 74
Tabel 5.3. Daftar Operator Penugasan dalam JavaScript 75
Tabel 5.4. Daftar Operator Logika dalam JavaScript 75
Tabel 5.5. Daftar Operator Perbandingan dalam JavaScript 76
Tabel 5.6. Daftar Operator *Bitwise* dalam JavaScript 77
Tabel 5.7. Daftar Operator *String* dalam JavaScript 78
Tabel 5.8. Daftar *Patterns* dalam *RegExp* 86
Tabel 5.9. Daftar *Modifier* dalam *RegExp* 88
Tabel 5.10. Daftar HTML *Event* (Lemay, Colburn dan Kyrnin 2015) 93

1

KONSEP DASAR PEMROGRAMAN WEB

Bab ini merupakan bab yang harus kita pahami sebelum mempelajari bab-bab selanjutnya dalam buku ini. Konsep dasar *web* yang akan dipelajari meliputi:

- perbedaan *web* dan *internet*
- perbedaan *web server* dan *client*
- definisi dan fungsi HTTP untuk *web*
- struktur, jenis, dan alamat URL situs *web*
- persiapan pengembangan *web*

Setelah selesai mempelajari bab ini maka diharapkan kita dapat menjelaskan konsep dasar *web* tersebut di atas.

1.1. *Web* dan *Internet*

Dalam keseharian, seringkali kita menganggap bahwa *web* dan *internet* sebagai hal yang sama. Pada kenyataannya mereka sebenarnya adalah dua hal berbeda namun memang saling berkaitan erat. *Web*, kependekan dari istilah *World Wide Web* (WWW), merupakan sekumpulan dokumen, gambar-gambar, dan bentuk digital lainnya yang dihubungkan melalui URL dan

hyperlink. Sedangkan *internet* adalah kumpulan dari berbagai jaringan komputer yang saling terkoneksi, dan dapat mencakup seluruh dunia, dengan melalui jalur telekomunikasi seperti telepon, *fiber-optic*, *wireless* dan lainnya. Dengan kata lain dapat dikatakan bahwa *internet* merupakan sarana yang digunakan untuk mengakses layanan *web* (Gambar 1.1).

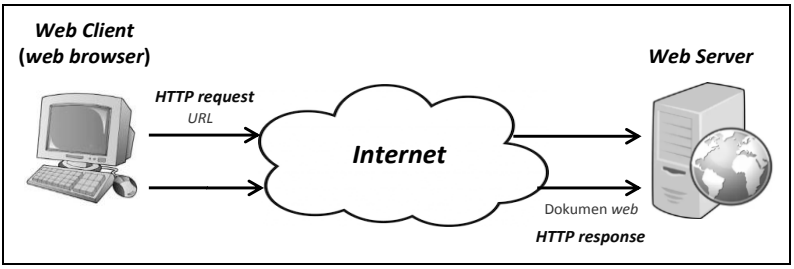
1.2. *Web Server* dan *Client*

Di dalam dunia *internet* selalu terdapat dua sisi yang saling mendukung (Gambar 1.2), yaitu:

a) *Server*, yang merupakan penyedia berbagai layanan. Mesin yang menangani layanan *web* disebut sebagai *web server*. Contoh *web server*: Apache, Microsoft IIS (*Internet Information Server*);
b) *Client*, yang bertugas mengakses layanan yang disediakan oleh *server*. Dalam hal ini, mesin *client* yang mengakses layanan *web* adalah mesin dengan aplikasi yang disebut sebagai *web browser*. Contoh *web browser*: Google Chrome, Mozilla Firefox, Internet Explorer, Safari, Opera.

Gambar 1.1. *Web* dan *internet* [1]

[1] Sumber: https://www.informationq.com/internet-and-www/



Gambar 1.2. ***Web Server*** **dan** ***Client***

1.3. HTTP (*Hypertext Transfer Protocol*)

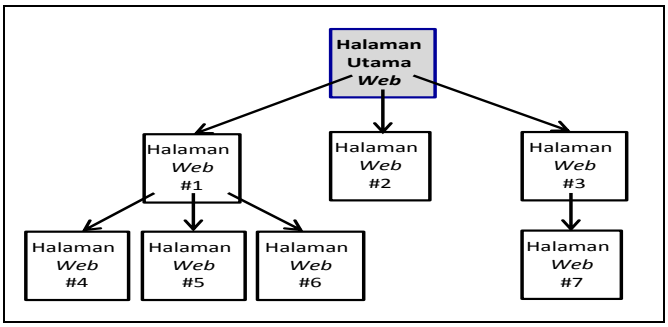
Protokol merupakan bahasa standar untuk mengatur komunikasi jaringan komputer. Pengiriman informasi di *internet* menggunakan suatu protokol yang disebut sebagai protokol transfer. HTTP (*Hypertext Transfer Protocol*) adalah jenis protokol transfer yang menjadi protokol standar untuk akses dokumen *web*. Perhatikan Gambar 1.2 untuk memahami cara kerja protokol ini. HTTP menentukan aturan yang perlu diikuti oleh *web browser* dalam meminta suatu dokumen (*request*), dan oleh *web server* dalam menyediakan dokumen (*response*) yang diminta *web browser*.

Catatan: Selain HTTP, terdapat jenis protokol transfer yang lain seperti: FTP (*File Transfer Protocol*), Gopher, NNTP (*Network News Transfer Protocol*), dan Telnet.

1.4. Situs *Web* (*Website*)

1.4.1. Struktur Situs *Web* (*Website*)

Ilustrasi dalam Gambar 1.3 memperlihatkan bahwa situs *web* umumnya terdiri dari sebuah halaman utama (*homepage*) dan beberapa halaman *web* (*webpages*).



Gambar 1.3. Struktur *situs web*

Deskripsi istilah-istilah yang berkaitan dengan situs *web* adalah sebagai berikut:

- Situs *web* (*website*), merupakan kumpulan dari satu atau lebih halaman *web* yang berisi tentang topik tertentu, yang saling terkoneksi sedemikian rupa (dengan menggunakan *hyperlink*) dan menjadi satu kesatuan
- Halaman *web* (*webpage*), merupakan halaman tertentu dari suatu situs *web*. Halaman *web* dapat juga disebut sebagai dokumen *web*, dan pada umumnya berbentuk dokumen HTML beserta segala komponen digital (seperti gambar dan media yang lain) yang ditampilkan dalam dokumen tersebut
- Halaman Utama (*homepage*), merupakan halaman yang pertama kali akan dilihat oleh *user* yang mengunjungi suatu situs *web*. Halaman ini umumnya digunakan untuk memberikan akses ke keseluruhan halaman *web* yang ada dalam situs tersebut, serta secara garis besar memberikan informasi tentang keseluruhan isi situs.

1.4.2. Jenis Situs *Web*

Situs *web* dapat dikategorikan menjadi menjadi dua jenis, yaitu:

- *Web* statis, merupakan jenis situs *web* yang menampilkan informasi yang sifatnya statis (tetap). Teknologi yang digunakan untuk membangun sistem ini berbasis *client-side*

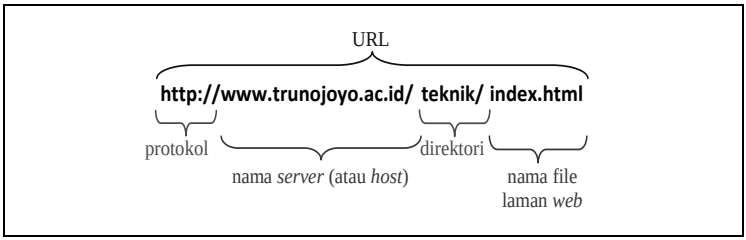
dan fokus pada pengembangan *web* di sisi *front-end*, seperti: HTML, CSS, JavaScript

- *Web* dinamis, merupakan jenis situs *web* yang menampilkan informasi yang sifatnya dinamis dan/atau memproses data yang dikirimkan oleh *user,* serta seringkali memerlukan akses basisdata. Teknologi yang digunakan merupakan gabungan dari teknologi berbasis *client-side* dan *server-side.* Teknologi berbasis *server-side* fokus pada pengembangan *web* di sisi *back-end*, seperti: CGI, ASP, JSP, ASP.NET, PHP.

Ketentuan: Buku ini fokus pada pengembangan *web* statis. Seluruh *file* yang dicontohkan di buku ini disimpan dalam direktori komputer pribadi pembuat skrip dan dapat secara langsung dibuka melalui *web browser* tanpa membutuhkan penggunaan *web server*. Namun perlu untuk diketahui bahwa, dalam dunia internet, situs *web* umumnya selalu disimpan dalam *server*.

1.4.3. URL Situs *Web*

URL (*Uniform Resource Locator*) digunakan untuk menentukan lokasi setiap dokumen atau informasi yang ada di dalam situs *web* pada *server*. URL (lihat Gambar 1.4) dapat diibaratkan sebagai suatu alamat yang terdiri dari empat komponen: (a) protokol yang digunakan, (b) nama *server* (atau *host*), (c) direktori penyimpanan *file*, dan (d) nama *file* (halaman *web*) yang akan diakses.



Gambar 1.4. Ilustrasi URL

1.5. Persiapan Pembuatan *Web*

Persiapan yang dibahas di sini meliputi bahasa dan aplikasi yang akan digunakan untuk pembuatan *web* statis.

1.5.1. Bahasa

Bahasa yang akan digunakan dalam buku ini adalah tiga bahasa standar yang digunakan untuk pengembangan *web* statis, yaitu:

1. HTML, yaitu bahasa markah yang digunakan untuk mendeskripsikan konten atau struktur suatu halaman *web* (pembahasan ada di dalam Bab 2)
2. CSS, yaitu bahasa *stylesheet* yang digunakan untuk memformat konten atau membuat *layout* halaman *web* (pembahasan ada di dalam Bab 3)
3. JavaScript yaitu bahasa pemrograman yang digunakan untuk mengatur perilaku halaman *web* pada *web browser* (pembahasan ada di dalam Bab 5).

Sebagai tambahan, buku ini juga membahas XML (dalam Bab 4) karena kaitan eratnya dengan HTML dan peran pentingnya dalam berbagai sistem IT.

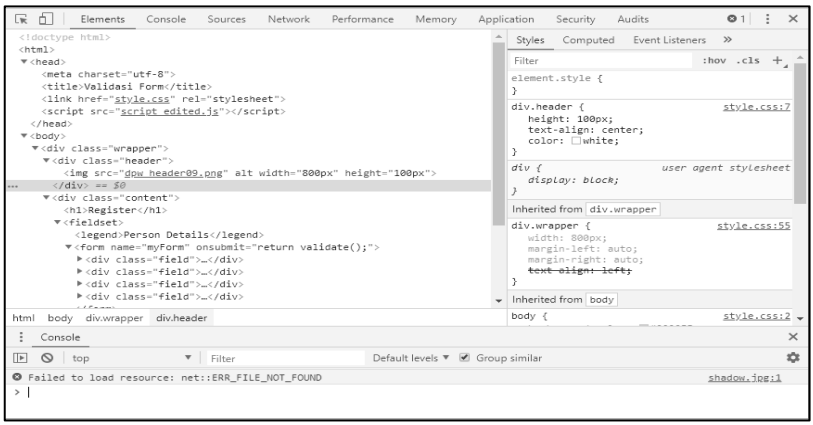
1.5.2. Aplikasi

Aplikasi yang diperlukan untuk pengembangan *web* statis adalah tiga aplikasi berikut:

- Penyunting Teks, yaitu aplikasi untuk menuliskan skrip dengan menggunakan komputer. Penyunting teks yang digunakan dalam buku ini adalah Notepad++ (https://notepad-plus-plus.org/)[2]

[2] Alternatif penyunting teks: Sublime Text (http://www.sublimetext.com/), Brackets (http://brackets.io/)

- *Web Browser*, yaitu aplikasi untuk membuka dokumen *web*. *Web browser* yang digunakan dalam buku ini adalah Google Chrome[3]
- *Developer Tools*, yaitu sebuah panel yang ada dalam *web browser* untuk menginspeksi skrip HTML, CSS, dan JavaScript. *Developer tools* yang digunakan dalam buku ini adalah *Google Chrome Developer Tools*. Pilihan cara mengaksesnya, dalam sistem operasi Windows atau Linux, adalah (a) dengan menekan tombol **F12** pada *keyboard*, (b) menekan kombinasi tombol **Ctrl+Shift+I** pada *keyboard*, atau (c) klik kanan tombol *mouse* dan kemudian pilih opsi ***Inspect***. Tampilan *Google Chrome Developer Tools* dapat dilihat dalam Gambar 1.5. Tiga *tab* utama dalam *Developer Tools* yang digunakan untuk melakukan inspeksi adalah:
  a) *tab* **Elements**, digunakan untuk inspeksi skrip HTML dan CSS yang ada dalam elemen HTML
  b) *tab* **Styles**, digunakan untuk inspeksi skrip CSS yang ada dalam elemen **<head>** dan *file* eksternal
  c) *tab* **Console**, digunakan untuk inspeksi skrip JavaScript.



Gambar 1.5. ***Google Chrome Developer Tools***

[3] Alternatif *web browser*: Mozilla Firefox, Internet Explorer, Opera

1.6. Latihan Soal

Latihan 1.1

Jelaskan persamaan dan perbedaan antara halaman web (*webpage*) dan halaman utama (*homepage*)!

Latihan 1.2

Jelaskan apa yang dimaksud dengan URL!

Latihan 1.3

Sebutkan dan jelaskan tiga bahasa standar yang digunakan dalam *web* statis!

2

HTML

Bab ini membahas HTML sebagai bahasa markah yang digunakan untuk membuat halaman *web*. Selain mempelajari dasar HTML, di sini kita juga akan mempelajari perintah-perintah HTML berikut:

- format teks
- pembuatan baris dan garis
- pemuatan *image* dan pembuatan *link*
- pembuatan *list*
- pembuatan tabel dan *form*

Setelah selesai mempelajari bab ini maka diharapkan kita dapat mendemonstrasikan aplikasi perintah-perintah HTML tersebut di atas untuk membuat halaman *web*.

2.1. Dasar HTML

HTML (*HyperText Markup Language*), pertama kali dikembangkan tahun 1991, adalah bahasa markah yang digunakan untuk mendeskripsikan konten atau struktur suatu halaman *web*. Standar HTML merupakan hasil kerja sama antara W3C *(World Wide Web Consortium)* dan WHATWG *(Web Hypertext*

Application Technology Working Group) (Krause 2016). Tabel 2.1 memperlihatkan sejarah perkembangan versi HTML.

Tabel 2.1. Sejarah perkembangan versi HTML (Krause 2016)

Tahun	Versi
1991	HTML
1993	HTML+
1995	HTML2
1997	HTML3.2
1999	HTML4.01
2001	XHTML1.1
2004	WHATWG
2006	Kerjasama WHATWG dengan W3C
2012	HTML5
2013	XHTML5
2015 - 2016	HTML5.2

Ketentuan: Buku ini mengimplementasikan HTML5. Akan tetapi, karena pembelajaran materi CSS (yang diperlukan untuk memformat konten atau membuat *layout* laman web) baru diberikan pada Bab 3, maka sebagian besar ilustrasi dalam Bab 2 masih tetap menggunakan versi HTML sebelumnya.

Tips: Selalu validasikan skrip HTML yang dibuat melalui *validator website* (https://validator.w3.org) untuk memastikan bahwa skrip yang dibuat sudah sesuai dengan sintaks HTML5.

2.1.1. Struktur dan Sintaks HTML

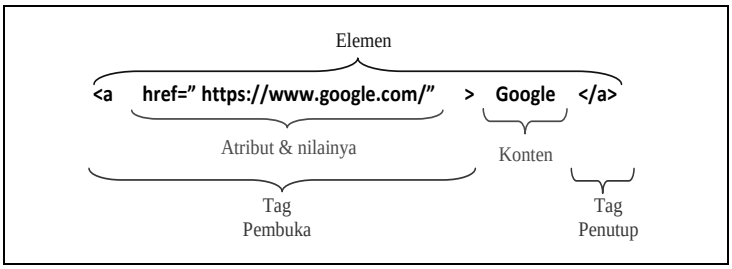
Sebelum memulai pembahasan struktur dokumen HTML, perhatikan Gambar 2.1 untuk memahami tiga istilah penting yang ada dalam HTML:

a) *Tag*, digunakan untuk menentukan tingkah laku dalam *web browser*

- *tag* pembuka, sintaks penulisan "**<namaTag>**". Contoh: **<a>**
- *tag* penutup, sintaks penulisan "**</namaTag>**". Contoh: **</a>**

b) Elemen (*element*), terdiri dari pasangan *tag* pembuka dan penutup, serta konten yang ada di dalamnya. Contoh: **<a>Google</a>**

c) Atribut (*attribute*), digunakan untuk memodifikasi nilai dari elemen HTML. Nilai dari atribut umumnya diletakkan di dalam tanda petik (" "). Contoh: **href="http://www.google.com/"**.



Gambar 2.1. Ilustrasi *tag*, elemen dan atribut dalam HTML

Catatan: HTML juga memiliki beberapa elemen yang tidak memiliki konten. Dalam hal ini, maka elemen HTML tersebut dapat dituliskan secara lebih singkat tanpa menggunakan *tag* penutup: "<**namaTag**>" atau "<**namaTag /**>".

Catatan: HTML juga memiliki beberapa atribut yang tidak memiliki nilai, misalnya atribut: **disabled**, **selected**, **checked**, **required**, dan lain-lain.

Struktur dasar suatu dokumen HTML terdiri dari empat elemen utama, yaitu:

a. **<!DOCTYPE>** untuk deklarasi tipe dokumen HTML
b. **<html>** untuk kontainer atau *root* dokumen HTML
c. **<head>** untuk kontainer informasi dokumen HTML yang tidak ditampilkan dalam *web browser*

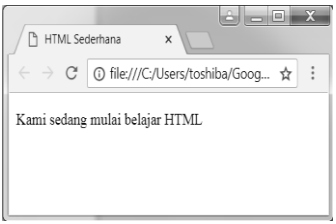
d. **<body>** untuk kontainer informasi yang akan ditampilkan dalam *web browser*.

Gambar 2.2 memperlihatkan contoh HTML sederhana yang memenuhi kriteria struktur dasar suatu dokumen HTML. Dokumen tersebut menggunakan elemen **<title>** di dalam elemen **<head>** untuk menambahkan informasi judul dokumen HTML, dan elemen **<p>** di dalam elemen **<body>** untuk menampilkan sebuah paragraf dalam *web browser*. Perhatikan bahwa deklarasi tipe dokumen HTML5 dituliskan sebagai **<!DEKLARASI html>**.

Skrip HTML:

```
1.  <!DOCTYPE html> <!-- untuk mendeklarasikan tipe dokumen HTML -->
2.  <html> <!-- untuk menyatakan bahwa dokumen ini merupakan dokumen HTML -->
3.      <head>
4.          <!-- memberikan informasi mengenai dokumen HTML -->
5.          <!-- tag-tag: TITLE, BASE, ISINDEX, LINK, SCRIPT, STYLE dan META -->
6.          <title> HTML Sederhana </title>
7.      </head>
8.      <body> <!-- menyimpan informasi atau data yang akan ditampilkan dalam  dokumen HTML -->
9.          <p>Kami sedang mulai belajar HTML</p>
10.     </body>
11. </html>
```

Hasil Eksekusi:



Gambar 2.2. Contoh HTML sederhana

Ketentuan: Untuk kemudahan ilustrasi, contoh-contoh yang ditunjukkan dalam sub-bab dan bab setelah ini (sebagian besar) tidak akan menampilkan penggunaan elemen **<!DOCTYPE html>**, **<html>**, **<head>** dan **<body>** dalam skrip HTML. Dengan demikian, mereka harus selalu dipahami sebagai (potongan) skrip yang diletakkan di dalam elemen **<body>**.

Tips: Gunakan aplikasi *Developer Tools* dalam *web browser* untuk menginspeksi skrip HTML (cek Sub-bab 1.5.2).

2.1.2. Komentar dalam HTML

Penambahan komentar di dalam skrip HTML dapat dilakukan dengan meletakkan baris-baris komentar di dalam tanda "**<!--**" dan "**-->**". Pemberian komentar adalah hal yang umum dilakukan dalam pemrograman. Fungsi komentar antara lain adalah untuk memberi nama aplikasi, mendeskripsikan tujuan penulisan suatu blok skrip tertentu di dalam *file*, memberi nama pengarang, tanggal pembuatan, nomer versi, ataupun informasi hak cipta.

2.2. Format Teks dalam HTML

Format teks dalam HTML yang dimaksud di sini adalah membuat struktur teks dari suatu halaman *web*. Format yang dibahas meliputi *heading*, *paragraph*, *blockquote*, *font* dan *font style*.

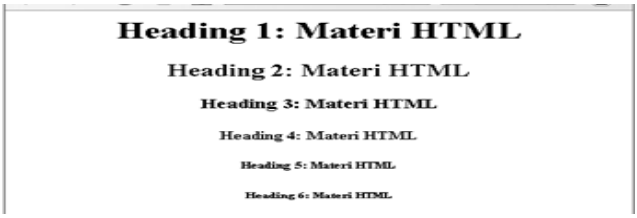
2.2.1. *Heading*

Format *heading* adalah untuk membuat judul atau sub judul dalam dokumen HTML. *Heading* memberikan pilihan tingkatan judul dari **<h1>** sebagai tingkatan terbesar hingga **<h6>** untuk tingkatan terkecil. Gambar 2.3 memperlihatkan contoh implementasi keenam tingkatan *heading* dalam dokumen HTML.

Skrip HTML:

```
1.  <h1 align="center">Heading 1: Materi HTML</h1>
2.  <h2 align="center">Heading 2: Materi HTML</h2>
3.  <h3 align="center">Heading 3: Materi HTML</h3>
4.  <h4 align="center">Heading 4: Materi HTML</h4>
5.  <h5 align="center">Heading 5: Materi HTML</h5>
6.  <h6 align="center">Heading 6: Materi HTML</h6>
```

Hasil Eksekusi:



Gambar 2.3. Contoh *heading*

Catatan: Penggunaan atribut *align* dalam elemen **<heading>** sudah tidak diimplementasikan lagi dalam HTML5. HTML5 merekomendasikan penggunaan CSS untuk memformat perataan *heading*.

2.2.2. *Paragraph*

Format *paragraph* adalah untuk membuat paragraf dalam dokumen HTML dengan menggunakan elemen **<p>**. Gambar 2.4 memperlihatkan contoh pembuatan satu *heading* dengan dua *paragraph*.

Skrip HTML:

```
1.  <h2 align="center">Materi HTML</h2>
2.  <p align="right">Kami sedang mulai belajar Dasar Pemrograman Web. Pada awal materi kami diperkenal
    kan pada konsep dasar Web </p>
3.  <p align="left">Saat ini masuk ke materi Dasar-dasar HTML </p>
```

Hasil Eksekusi:

Gambar 2.4. Contoh *paragraph*

Catatan: Penggunaan atribut *align* dalam elemen **<paragraph>** sudah tidak diimplementasikan lagi dalam HTML5. HTML5 merekomendasikan penggunaan CSS untuk memformat paragraf.

Catatan: Penambahan tanda spasi atau baris tidak akan mempengaruhi hasil eksekusi skrip HTML karena *web browser* akan mengabaikan tambahan spasi dan baris tersebut ketika laman *web* ditampilkan. Gunakan elemen **<pre>** jika ingin mendapatkan tampilan yang sama persis dengan skrip yang dibuat.

2.2.3. *Blockquote*

Format *blockquote* adalah untuk menuliskan kutipan teks dalam dokumen HTML dengan menggunakan elemen

<blockquote>. Gambar 2.5 memperlihatkan contoh pembuatan satu *blockquote*.

Skrip HTML:

```
1.  <p>Contoh quote: <blockquote>No gain without pain</blockquote></p>
```

Hasil Eksekusi:

Contoh quote:

No gain without pain

Gambar 2.5. Contoh *blockquote*

2.2.4. *Font*

Format *font* adalah untuk mendefinisikan ukuran, jenis dan warna dari *font* dengan menggunakan elemen **<font>** dan atribut **size**, **face**, serta **color** untuk memodifikasi ukuran, jenis dan warna *font*. Gambar 2.6 memperlihatkan contoh pembuatan *font* dengan variasi tiga atribut di atas.

Skrip HTML:

```
1.  <p>Teks berukuran 3 (ukuran normal) dibanding <font size="7">teks  berukuran 7</font></p>
2.  <p><font face="courier">Teks ini menggunakan font-face Courier</font></p>
3.  <p><font color="blue">Teks ini berwarna biru</font></p>
```

Hasil Eksekusi:

Teks berukuran 3 (ukuran normal) dibanding teks berukuran 7

Teks ini menggunakan font-face Courier

Teks ini berwarna biru

Gambar 2.6. Contoh *font*

Catatan: Elemen **<font>** dan atribut-atributnya sudah tidak diimplementasikan lagi dalam HTML5. HTML5 merekomendasikan penggunaan CSS untuk memformat *font*.

2.2.5. *Font Style*

Format *font style* adalah mendefinisikan bagaimana teks ditampilkan. Tabel 2.2 memperlihatkan daftar *font style* dalam HTML dan ilustrasi contoh penggunaannya dapat dilihat dalam Gambar 2.7.

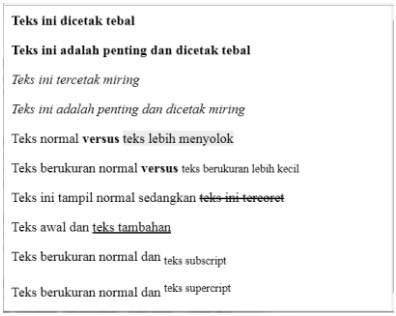
Tabel 2.2. Daftar *font style* dalam HTML

Elemen	Fungsi
<b>	Membuat teks menjadi tercetak tebal (*bold*)
<string>	Membuat teks menjadi penting dan tercetak tebal
<i>	Membuat teks menjadi tercetak miring (*italic*)
****	Membuat teks menjadi penting dan tercetak miring
<mark>	Membuat teks menjadi lebih menyolok
<small>	Membuat teks menjadi berukuran lebih kecil
<del>	Membuat teks menjadi tercoret (*deleted*)
<ins>	Membuat teks menjadi teks tambahan (*inserted*)
<sub>	Membuat teks menjadi *subscript*
<sup>	Membuat teks menjadi *superscript*

Skrip HTML:

```
1.  <p><b>Teks ini dicetak tebal</b></p>
2.  <p><strong>Teks ini adalah penting dan dicetak tebal</strong></p>
3.  <p><i>Teks ini tercetak miring</i></p>
4.  <p><em>Teks ini adalah penting dan dicetak miring</em></p>
5.  <p>Teks normal <b>versus</b> <mark>teks lebih menyolok</mark></p>
6.  <p>Teks berukuran normal <b>versus</b> <small>teks berukuran lebih kecil</small></p>
7.  <p>Teks ini tampil normal sedangkan <del>teks ini tercoret</del></p>
8.  <p>Teks awal dan <ins>teks tambahan</ins></p>
9.  <p>Teks berukuran normal dan <sub>teks  subscript</sub></p>
10. <p>Teks berukuran normal dan <sup>teks supercript</sup></p>
```

Hasil Eksekusi:



Gambar 2.7. Contoh *font type*

2.3. Baris dan Garis dalam HTML

2.3.1. *Break*

Break dalam HTML adalah untuk mengganti baris atau menuliskan teks pada baris berikutnya. *Break* dibuat dengan menggunakan elemen **
**.

2.3.2. Garis Horisontal

Garis horisontal dalam HTML adalah untuk memberikan sekat baris antara (kelompok) teks yang satu dengan (kelompok) teks lain. Garis horisontal dibuat dengan menggunakan elemen **<hr>**.

Catatan: Elemen **
** dan **<hr>** merupakan elemen HTML yang tidak memiliki konten.

2.4. *Image* dan *Link* dalam HTML

Terdapat satu konsep yang harus kita pahami sebelum memulai pembahasan *image* dan *link* dalam HTML. Konsep yang dimaksud di sini disebut sebagai *file path*, yang digunakan untuk mendeskripsikan lokasi suatu *file* di dalam struktur direktori suatu *situs web*. HTML memiliki dua macam *file path*, yaitu:

- *Absolute path*, yang mendeskripsikan alamat URL atau lokasi suatu *file* di dalam mesin/komputer
- *Relative path*, yang mendeskripsikan lokasi suatu *file* (yang relatif) terhadap lokasi *file* (skrip HTML) halaman *web*.

Cara yang paling baik untuk mendeskripsikan lokasi suatu *file* adalah dengan menggunakan *relative path*, karena dapat dipastikan bahwa halaman *web* akan selalu dapat dibuka dengan lengkap dan sempurna meskipun direktori *situs web* dipindahkan lokasinya.

Catatan: Elemen **<img>** merupakan elemen HTML yang tidak memiliki konten.

Catatan: Penulisan deskripsi lokasi file (dan juga nama *file*) dalam *file path* adalah *case-sensitive*.

2.4.1. *Image*

Image dalam HTML adalah memuat gambar dengan menggunakan elemen **<img>** dan menambahkan atribut **src** untuk menentukan lokasi gambar. Tabel 2.3 memperlihatkan contoh perbandingan *file path* untuk memuat suatu gambar dalam *web browser*. Perhatikan bahwa direktori *situs web* berada di dalam *drive* D komputer dan penulisan *relative path* menjadikan lokasi file gambar relatif terhadap lokasi *file* skrip HTML halaman *web*.

Tabel 2.3. Contoh perbandingan *file path*

Lokasi *file* skrip HTML halaman *web*: "D:\doc\dpw\"		
Lokasi file "pic.jpg"	***Absolute Path***	***Relative Path***
Di dalam direktori yang sama dengan *file* HTML	**<img src="D:\doc\dpw\pic.jpg">**	**<img src="pic.jpg">** atau **<img src="./pic.jpg">**
Di dalam sub direktori **image** dari direktori *file* HTML	**<img src="D:\doc\dpw\image\pic.jpg">**	**<img src="image/pic.jpg">** atau **<img src="./image/pic.jpg">**
Satu level di atas direktori *file* HTML	**<img src="D:\doc\pic.jpg">**	**<img src="../pic.jpg">**

2.4.2. *Link*

Link dalam HTML adalah membuat *hyperlink* dengan menggunakan elemen **<a>** dan atribut **<href>** untuk mendefinisikan lokasi *link*. Tabel 2.4 memperlihatkan format penulisan berbagai jenis *link*.

Tabel 2.4. Penulisan *link*

Jenis *link*	Format Penulisan
dokumen lain	**<a href="**lokasi&namaFile**" >** Teks **</a>**
URL situs *web*	**<a href="**URL**" >** Teks **</a>**
alamat *email*	**<a href="mailto:**alamatEmail**" >** Teks **</a>**
file yang akan diunduh	**<a href="** lokasi&namaFile**" download>** Teks**</a>**

Catatan: *File path* juga diterapkan dalam pembuatan *link* ke file eksternal CSS (Bab 3), XML (Bab 4), dan JavaScript (Bab 5).

2.5. *List* dalam HTML

List dalam HTML adalah untuk membuat daftar item. Jenis dari *list* meliputi *unordered*, *ordered*, dan *definition*.

2.5.1. *Unordered list*

Unordered list digunakan untuk membuat daftar item dengan tanda *bullet point*. Pembuatan *unordered list* dilakukan dengan menempatkan daftar elemen **<li>** di dalam elemen **<ul>**.

2.5.2. *Ordered list*

Sama dengan *unordered list*, *ordered list* digunakan untuk membuat daftar item. Yang membedakan adalah bahwa *ordered list* adalah untuk daftar yang memiliki penomoran. Pembuatan *ordered list* dilakukan dengan menempatkan daftar elemen **<li>** di dalam elemen **<ol>**.

2.5.3. *Definition List*

Definition list digunakan untuk membuat daftar grup dimana setiap grup memiliki daftar definisi masing-masing nama-nilai. Pembuatan *definition list* dilakukan dengan menempatkan daftar kelompok (elemen **<dt>**) dan setiap daftar definisinya (elemen **<dd>**) di dalam elemen **<dl>**.

Gambar 2.8 memperlihatkan contoh perbandingan *unordered list*, *ordered list* dan *definition list* dalam dokumen HTML. Perhatikan bahwa perbedaan antara *unordered list* dan *ordered list* adalah mengenai ada tidaknya penomoran. Sedangkan contoh *definition list* adalah mengenai dua kelompok pemrograman *web* yaitu *client-side* dan *server-side*. Setiap kelompok terdiri dari beberapa bahasa pemrograman yang berkaitan dengannya.

Skrip HTML:

```
1.  <!-- unordered list -->
2.  <p><b><i>Unordered list:</i></b></p>
3.  <ul type="square">
4.      <li>Daftar 1</li>
5.      <li>Daftar 2</li>
6.      <li>Daftar 3</li>
7.  </ul>
8.  <!-- ordered list -->
9.  <p><b><i>Ordered list:</i></b></p>
10. <ol type="a" start="1">
11.     <li>Daftar 1</li>
12.     <li>Daftar 2</li>
13.     <li>Daftar 3</li>
14. </ol>
15. <!-- definition list -->
16. <p><b><i>Definition list:</i></b></p>
17. <dl>
18.     <dt><i>Client-side Programming:</i></dt>
19.         <dd>HTML</dd>
20.         <dd>CSS</dd>
21.         <dd>JavaScript</dd>
22.     <dt><i>Server-side Programming:</i></dt>
23.         <dd>PHP</dd>
24.         <dd>ASP</dd>
25.         <dd>Node.js</dd>
26. </dl>
```

Hasil Eksekusi:

Unordered list:

- Daftar 1
- Daftar 2
- Daftar 3

Ordered list:

a. Daftar 1
b. Daftar 2
c. Daftar 3

Definition list:

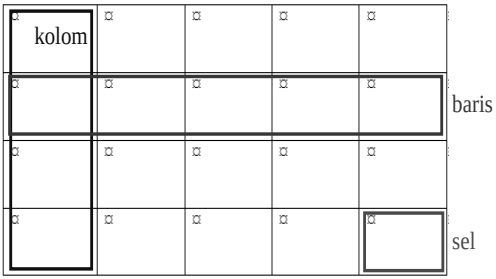
Client-side Programming:
HTML
CSS
JavaScript
Server-side Programming:
PHP
ASP
Node.js

Gambar 2.8. Contoh perbandingan *list*

Catatan: Penggunaan atribut **type** dalam elemen **<ul>** serta atribut **type** dan **start** dalam elemen **<ol>** sudah tidak diimplementasikan lagi dalam HTML5. HTML5 merekomendasikan penggunaan CSS untuk mendefinisikan jenis *bullet point* serta jenis penomoran dan nomor awal dari suatu *list*.

2.6. Tabel dalam HTML

Tabel banyak digunakan di dalam dokumen HTML untuk menampilkan informasi secara tabular. Gambar 2.9 memperlihatkan bahwa pada dasarnya setiap tabel memiliki tiga elemen utama, yaitu baris, kolom, dan sel. Setiap sel dari tabel dapat digunakan untuk menampilkan isian karakter, kata, gambar ataupun tabel lain. Di era sebelum CSS lazim digunakan, tabel juga digunakan untuk membantu mengatur tampilan halaman *web* agar lebih menarik.



Gambar 2.9 . Elemen utama tabel

2.6.1. Struktur Tabel

Pembuatan tabel dalam HTML memerlukan setidaknya tiga elemen berikut:

1. **<table>**, yang digunakan untuk membuat kontainer tabel
2. **<tr>**, singkatan dari *table row*, yang digunakan untuk membuat baris
3. **<td>**, singkatan dari *table data*, yang digunakan untuk membuat sel (yaitu kolom dari baris).

Skrip HTML:

```
1.  <table border="1">
2.      <tr>
3.          <td>Baris 1 Kolom 1</td>
4.          <td>Baris 1 Kolom 2</td>
5.      </tr>
6.      <tr>
7.          <td>Baris 2 Kolom 1</td>
8.          <td>Baris 2 Kolom 2</td>
9.      </tr>
10. </table>
```

Hasil Eksekusi:

Baris 1 Kolom 1	Baris 1 Kolom 2
Baris 2 Kolom 1	Baris 2 Kolom 2

Gambar 2.10. Contoh tabel sederhana

Gambar 2.10 memperlihatkan contoh skrip HTML untuk membuat tabel sederhana. Perhatikan bahwa pada elemen **<table>** diberikan atribut **border** yang digunakan untuk memberikan nilai garis tepi dari tabel. Nilai ini dalam ukuran **pixel**, sehingga **border="1"** memiliki arti bahwa tampilan tabel tersebut pada *web browser* akan memiliki garis tepi sebesar 1 pixel. Jika atribut **border** tidak ditambahkan maka secara *default* tabel tidak memiliki garis tepi atau sama artinya dengan **border="0"**.

Catatan: Penggunaan atribut **border** dalam elemen **<table>** sudah tidak diimplementasikan lagi dalam HTML5. HTML5 merekomendasikan penggunaan CSS untuk memformat garis tepi dari tabel.

2.6.2. Judul pada Tabel

Judul pada tabel yang dimaksud di sini meliputi judul tabel dan judul kolom/baris. Elemen untuk pembuatan judul meliputi:

- **<caption>**, yang digunakan untuk membuat judul tabel
- **<th>**, singkatan dari *table header*, yang digunakan untuk membuat judul kolom atau baris, ataupun gabungan keduanya.

Gambar 2.11 memperlihatkan contoh skrip HTML untuk membuat tabel yang memiliki judul tabel dan judul kolom. Perhatikan bahwa pada elemen **<caption>** memberikan judul tabel sebagai **DAFTAR MAHASISWA**. Sedangkan tiga elemen **<th>** diletakkan di dalam satu elemen **<tr>** dan digunakan untuk memberikan judul pada masing-masing dari ketiga kolom sebagai **No**, **NPM**, dan **Nama**.

Skrip HTML:

```
1.  <table border="1">
2.      <caption>
3.          <b>DAFTAR MAHASISWA</b>
4.      </caption>
5.      <tr>
6.          <th>No</th><th>NPM</th><th>Nama</th>
7.      </tr>
8.      <tr>
9.          <td>1</td><td>06.100.001</td><td>Amin A. Angkasa</td>
10.     </tr>
11.     <tr>
12.         <td>2</td><td>06.100.002</td><td>Beni B. Bernardi</td>
13.     </tr>
14. </table>
```

Hasil Eksekusi:

DAFTAR MAHASISWA

No	NPM	Nama
1	06.100.001	Amin A. Angkasa
2	06.100.002	Beni B. Bernardi

Gambar 2.11. Contoh tabel, judul tabel, dan judul kolom

Ada kalanya tabel yang dibutuhkan memerlukan judul pada sisi baris dan bukannya kolom. Untuk itu, maka pembuatan judul baris dilakukan dengan meletakkan elemen **<th>** pada setiap elemen **<tr>**. Gambar 2.12 memperlihatkan contoh skrip HTML untuk penambahan judul tabel dan judul baris.

Skrip HTML:

```
1.  <table border="1">
2.      <caption>
3.          <b>DATA FISIK</b>
4.      </caption>
5.      <tr>
6.          <th>Nama</th><td>Amin A. Angkasa</td><td>Beni B. Bernardi</td>
7.      </tr>
8.      <tr>
9.          <th>Tinggi</th><td>170</td><td>175</td>
10.     </tr>
11.     <tr>
12.         <th>Berat Badan</th><td>80</td><td>75</td>
13.     </tr>
14. </table>
```

Hasil Eksekusi:

DATA FISIK

Nama	Amin A. Angkasa	Beni B. Bernardi
Tinggi	170	175
Berat Badan	80	75

Gambar 2.12. Contoh tabel, judul tabel, dan judul baris

2.6.3. Penggabungan Sel Kolom

Penggabungan sel kolom dapat dilakukan dengan menambahkan atribut **colspan** pada elemen **<td>** atau **<th>**. Gambar 2.13 memperlihatkan contoh skrip HTML yang menggabungkan dua sel kolom pada baris pertama tabel.

Skrip HTML:

```
1.  <table border="1">
2.      <tr>
3.          <th colspan="2">Jumlah Mahasiswa</th>
4.      </tr>
5.      <tr>
6.          <th>Laki-laki</th>
7.          <th>Perempuan</th>
8.      </tr>
9.      <tr>
10.         <td>20</td>
11.         <td>22</td>
12.     </tr>
13. </table>
```

Hasil Eksekusi:

Jumlah Mahasiswa	
Laki-laki	Perempuan
20	22

Gambar 2.13. Contoh penggabungan sel kolom tabel

2.6.4. Penggabungan Sel Baris

Penggabungan sel baris dapat dilakukan dengan menambahkan atribut **rowspan** pada elemen **<td>** atau **<th>**. Gambar 2.14 memperlihatkan contoh skrip HTML yang menggabungkan dua sel baris pada kolom pertama tabel.

Skrip HTML:

```
1.  <table border="1">
2.      <tr>
3.          <th rowspan="2">Jumlah Mahasiswa</th>
4.          <th>Laki-laki</th>
5.          <td>20</td>
6.      </tr>
7.      <tr>
8.          <th>Perempuan</th>
9.          <td>22</td>
10.     </tr>
11. </table>
```

Hasil Eksekusi:

Jumlah Mahasiswa	Laki-laki	20
	Perempuan	22

Gambar 2.14. Contoh penggabungan sel baris tabel

2.6.5. Format *Header*, *Body* dan *Footer*

Tabel pada HTML5 dapat diformat menjadi tiga bagian, yaitu: *header* (judul kolom/baris), *body* (isi tabel), dan *footer* (*summary* tabel). Format ini berfungsi untuk mempermudah pembuatan *style* (menggunakan CSS) pada tiap bagian tabel. Elemen yang diperlukan untuk membuat format tabel adalah:

- **<thead>** yang digunakan untuk memformat bagian *header* tabel
- **<tbody>** yang untuk memformat bagian *body* tabel
- **<tfoot>** yang untuk memformat bagian *footer* tabel.

Gambar 2.15 memperlihatkan contoh format *header*, *body* dan *footer* pada suatu tabel.

Skrip HTML:

```
1.  <table border="1">
2.      <thead>
3.          <tr>
4.              <th>NIM</th><th>Nama</th>
5.          <tr>
6.      </thead>
7.      <tbody>
8.          <tr>
9.              <td>06.100.001</td>
10.             <td>Amin A. Angkasa</td>
11.         </tr>
12.         <tr>
13.             <td>06.100.001</td>
14.             <td>Beni B. Bernardi</td>
15.         </tr>
16.     </tbody>
17.     <tfoot>
18.         <tr>
19.             <td colspan="2">2 data mahasiswa</td>
20.         </tr>
21.     </tfoot>
22. </table>
```

Hasil Eksekusi:

NIM	Nama
06.100.001	Amin A. Angkasa
06.100.001	Beni B. Bernardi
2 data mahasiswa	

Gambar 2.15. Contoh format *header*, *body* dan *footer* pada tabel

Catatan: Penggunaan format *header*, *body* dan *footer* dalam pembuatan tabel bersifat opsional.

2.6.6. Mengatur lebar dan tinggi tabel

Suatu tabel dan sel dapat diatur lebar dan tingginya dengan menambahkan atribut **width** dan **height** dalam elemen **<table>**, **<th>** dan **<td>**. Gambar 2.16 memperlihatkan contoh tabel yang diatur lebarnya dan sel yang diatur lebar serta tingginya.

Skrip HTML:

```
1.  <table border="1" width="50%">
2.      <caption><b>DAFTAR MAHASISWA</b></caption>
3.      <tr>
4.          <th>No</th><th>NPM</th><th>Nama</th></tr>
5.      <tr>
6.          <td width="20">1.</td>
7.          <td width="80" height="50">06.100.001</td>
8.          <td width="180" height="50">Amin A. Angkasa</td>
9.      </tr>
10.     <tr>
11.         <td width="20">2.</td>
12.         <td width="80" height="70">06.100.002</td>
13.         <td width="180" height="70">Beni B. Bernardi</td>
14.     </tr>
15. </table>
```

Hasil Eksekusi:

DAFTAR MAHASISWA

No	NPM	Nama
1.	06.100.001	Amin A. Angkasa
2.	06.100.002	Beni B. Bernardi

Gambar 2.16. Contoh pengaturan lebar dan tinggi tabel

Catatan: Penggunaan atribut **width** dan **height** dalam elemen **<table>** dan **<td>** sudah tidak diimplementasikan lagi dalam HTML5. HTML5 merekomendasikan penggunaan CSS untuk mengatur lebar dan tinggi tabel dan sel.

2.6.7. Membuat warna pada tabel

Dalam tabel, warna latar dapat dibuat dengan menambahkan atribut **bgcolor** dalam elemen **<table>**, **<tr>**, **<th>** dan **<td>**. Gambar 2.17 memperlihatkan contoh tabel yang

berhiaskan warna latar (**bgcolor**), dan untuk memperjelas ilustrasi maka halaman *web* juga diberikan warna latar.

Skrip HTML:

```
1.  <body bgcolor="purple">
2.      <font size="3" face="arial">
3.          <table border="2" bgcolor="white" align="center">
4.              <tr bgcolor="yellow">
5.                  <th>No</th><th>NPM</th><th>Nama</th></tr>
6.              <tr><td bgcolor="blue">1.</td>
7.              <td bgcolor="red" width="80" height="40">06.100.001</td>
8.              <td bgcolor="green" width="180" height="40">Amin A. Angkasa</td></tr>
9.          </table>
10.     </font>
11. </body>
```

Hasil Eksekusi:

No	NPM	Nama
1.	06.100.001	Amin A. Angkasa

Gambar 2.17. Contoh membuat warna pada tabel

Catatan: Penggunaan atribut **bgcolor**, **align**, dan **valign** dalam elemen **<body>**, **<table>**, **<th>**, **<tr>**, dan **<td>** sudah tidak diimplementasikan lagi dalam HTML5. HTML5 merekomendasikan penggunaan CSS memformat warna dan perataan dalam tabel.

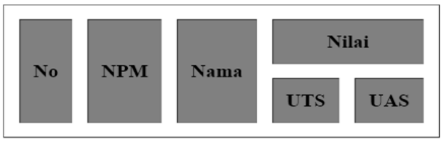
2.6.8. *Cellpading* dan *cellspacing*

Pengaturan spasi antara dua buah sel dan garis tepi dengan tulisan dalam tabel dapat dilakukan dengan menambahkan atribut **cellspacing** dan **cellpadding** dalam elemen **<table>**. Gambar 2.168 memperlihatkan contoh tabel yang diatur spasinya dengan **cellspacing** dan **cellpadding**.

Skrip HTML:

```
1.  <table border="1" bgcolor="white" cellpadding="10" cellspacing="12">
2.      <tr bgcolor="gray">
3.          <th rowspan="2">No</th>
4.          <th rowspan="2">NPM</th>
5.          <th rowspan="2">Nama</th>
6.          <th colspan="2">Nilai</th>
7.      </tr>
8.      <tr bgcolor="gray">
9.          <th>UTS</th>
10.         <th>UAS</th>
11.      </tr>
12. </table>
```

Hasil Eksekusi:



Gambar 2.18. Contoh cellpadding dan cellspacing

Catatan: Penggunaan atribut **cellspacing** dan **cellpadding** dalam elemen **<table>** sudah tidak diimplementasikan lagi dalam HTML5. HTML5 merekomendasikan penggunaan CSS untuk mengatur spasi dalam tabel.

2.7. *Form* dalam HTML

Form dalam HTML berfungsi untuk menerima informasi (atau meminta umpan balik dari *user*) dan kemudian mengirimkannya ke *server*.

2.7.1. Struktur *Form*

Struktur *form* dibuat dengan menggunakan elemen **<form>** dan (setidaknya) tambahan atribut **action** dan **method**, seperti yang terlihat dalam Gambar 2.19. Atribut **action** digunakan untuk mendefinisikan tindakan apa yang akan dilakukan ketika *form* dikirimkan (dengan menekan tombol **Submit**). Pada umumnya, tindakan yang dilakukan adalah mengirimkan isian *form* ke suatu halaman *web*. Dengan demikian

nilai dari atribut **action** umumnya adalah berupa alamat URL atau nama *file* dokumen *web*.

Atribut **method** digunakan untuk menentukan metode HTTP yang digunakan untuk mengirimkan data ke *server*. Terdapat dua metode HTTP yang dapat digunakan dalam *form*, yaitu:

a) **POST**, digunakan untuk isian *form* yang berupa data sensitif atau rahasia
b) **GET**, digunakan untuk isian *form* yang bukan berupa data sensitif karena data yang dikirimkan akan dapat dilihat melalui URL.

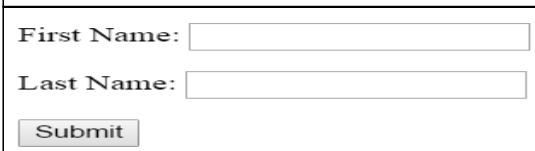
```
<form action = "..." method = "...">
   <!-- elemen-elemen form -->
</form>
```

Gambar 2.19. Struktur *form*

Skrip HTML:

```
1.  <form action="submit.html" method="post">
2.      <p><label for="fName">First Name</label>:
3.      <input id="fName" name= "fName" type="text"/>
4.      </p>
5.      <p>
6.      <label for="lName">Last Name</label>:
7.      <input id="lName" name= "lName" type="text"/>
8.      </p>
9.      <input type="submit" value="Submit">
10. </form>
```

Hasil Eksekusi:



Gambar 2.20. Contoh *form* sederhana

Gambar 2.20 memperlihatkan contoh skrip HTML untuk membuat *form* sederhana. Perhatikan bahwa contoh *form* menggunakan tiga elemen **<input>**. Dua elemen **<input>** yang pertama diberi tambahan atribut **id** dan **name**. Atribut **name** adalah atribut yang harus dimiliki oleh setiap elemen dalam *form*

karena ia digunakan sebagai pengenal elemen tersebut. Sehingga, data suatu elemen yang tidak memiliki atribut **name** tidak akan pernah dikirimkan ke *server*. Atribut **id** juga berfungsi sebagai pengenal elemen, namun ia hanya bekerja di *client-side* saja. Perhatikan juga bahwa tiap elemen **<input>** dalam contoh *form* memiliki nilai atribut **type** yang berbeda. Dua elemen **<input>** yang pertama menggunakan nilai atribut **type="text"** yang berarti bahwa elemen tersebut adalah berupa satu baris kotak masukan berbentuk teks. Elemen **<input>** yang ketiga menggunakan nilai atribut **type="submit"** yang berarti bahwa elemen tersebut menjadi tombol (**Submit**) untuk pengiriman data isian *form* ke *server*. Atribut **type** akan dibahas lebih lanjut dalam Sub-bab 2.7.3, sedangkan penggunaan lebih lanjut mengenai *form* akan dibahas dalam Bab 5.

Catatan: Elemen **<input>** yang menggunakan nilai **type="submit"** tidak memerlukan penambahan atribut **id** ataupun **name** karena ia secara otomatis dikenali sebagai tombol untuk pengiriman data ke *server*.

2.7.2. Pengelompokkan *Form*

Data *form* yang saling berkaitan dapat dikelompokkan (tampilannya) agar lebih menarik untuk dilihat dan lebih mudah untuk dibaca. Pengelompokkan dilakukan dengan menggunakan elemen **<fieldset>** yang diletakkan di dalam elemen **<form>**. Elemen **<fieldset>** juga dapat diberi judul dengan menambahkan elemen **<legend>**. Contoh penggunaan elemen **<fieldset>** dapat dilihat dalam Gambar 2.21.

2.7.3. Elemen <input>

Elemen **<input>** adalah elemen yang paling penting dalam *form*. Elemen memiliki berbagai bentuk tampilan yang ditentukan oleh nilai atribut **type** yang digunakan.

Tabel **2.5** memperlihatkan daftar atribut **type** yang dapat digunakan dalam elemen **<input>**. Selain atribut **type**, elemen **<input>** juga memiliki beberapa atribut lain seperti yang diperlihatkan dalam

Tabel 2.6. Contoh penggunaan elemen **<input>** dengan bermacam variasi atribut **type** dan atribut yang lain diilustrasikan dalam Gambar 2.21.

Catatan: Elemen <**input**> merupakan elemen HTML yang tidak memiliki konten.

Catatan: Bentuk tampilan elemen <**input**> untuk tiap atribut **type** tergantung pada aplikasi *web browser* yang digunakan.

Tabel 2.5. Atribut type dalam elemen <input>

Atribut Type	Fungsi
type="text"	Membuat satu baris kotak masukan berbentuk teks
type="password"	Membuat satu baris kotak masukan berbentuk teks yang terenkripsi
type="submit"	Membuat tombol *submit* untuk mengirim data isian *form* ke *server*
type="reset"	Membuat tombol *reset* untuk menghapus data isian *form*
type="radio"	Membuat tombol radio (*radio button*) yang digunakan untuk meminta *user* menentukan satu pilihan
type="checkbox"	Membuat *checkbox* yang digunakan untuk meminta *user* menentukan tak satupun atau satu hingga bahkan beberapa pilihan
type="button"	Membuat sebuah tombol
type="color"	Membuat kotak masukan berbentuk palet warna yang dapat dipilih

type="date"	Membuat kotak masukan berbentuk tanggal
type="datetime-local"	Membuat kotak masukan berbentuk tanggal dan waktu (tanpa ketentuan *time zone*)
type="time"	Membuat kotak masukan berbentuk pilihan waktu (tanpa ketentuan *time zone*)
type="week"	Membuat kotak masukan berbentuk pilihan minggu dan tahun
type="month"	Membuat kotak masukan berbentuk pilihan bulan dan tahun
type="tel"	Membuat kotak masukan untuk format nomer telepon
type="url"	Membuat kotak masukan untuk alamat URL
type="email"	Membuat kotak masukan untuk format alamat *email*
type="file"	Membuat sebuah kotak untuk memilih suatu *file* dan sebuah tombol "**Browse**" untuk mengunggah *file* tersebut
type="number"	Membuat kotak masukan dengan format angka
type="range"	Membuat sebuah kontrol bentuk *slider* untuk memilih sebuah nilai
type="search"	Membuat kotak masukan teks pencarian

Tabel 2.6. Atribut selain type dalam elemen <input>

Atribut	Fungsi
value	Menentukan nilai awal kotak masukan
readonly	Menentukan bahwa isi kotak masukan tidak dapat diubah
disabled	Menentukan bahwa kotak masukan tidak diaktifkan dan isinya tidak akan dikirimkan ke *server*
size	Menentukan ukuran (karakter) untuk kotak masukan
maxlength	Menentukan panjang maksimal isi kotak masukan

min	Menentukan nilai minimal isi kotak masukan (berlaku untuk **type**: **number**, **range**, **date**, **datetime-local**, **month**, **time** dan **week**)
max	Menentukan nilai maksimal isi kotak masukan (berlaku untuk **type**: **number**, **range**, **date**, **datetime-local**, **month**, **time** dan **week**)
placeholder	Menentukan petunjuk (format) pengisian kotak masukan (berlaku untuk **type**: **text**, **password**, **url**, **tel**, **email**, dan **search**)
required	Menentukan bahwa kotak masukan harus diisi sebelum *form* dikirimkan (berlaku untuk **type**: **text**, **password**, **number**, **url**, **tel**, **email**, **date**, **checkbox**, **radio**, **file**, dan **search**)
checked	Menentukan bahwa pilihan pada kotak masukan dengan **type="checkbox"** atau **type="radio"** harus diisi (*checked*)
pattern	Menentukan (format) *Regular Expression* yang harus diikuti untuk pengisian kotak masukan (berlaku untuk **type**: **text**, **password**, **url**, **tel**, **email**, dan **search**)

Skrip HTML:

```
1.  <form action="submit.html" method="post">
2.      <fieldset>
3.          <legend>Data Registrasi</legend>
4.      <!-- type = "text"  -->
5.          <label for="username">Username:</label>
6.          <input type="text" name="username" id="username" size="20" maxlength="20" required /> <br>
7.      <!-- type = "password" -->
8.          <label for="password">Password:</label>
9.          <input type="password" name="password" id="password" size="20" required /> <br>
10.     <!-- type = "date" -->
11.         <label for="birthdate">Tanggal Lahir:</label>
12.         <input type="date" name="birthdate" id="birthdate" required /> <br>
13.     <!-- type = "tel" -->
14.         <label for="telephone">Telepon:</label>
15.         <input name="telephone" id="telephone" placeholder="xxx-xxxxxxx" pattern="[0-9]{3}-[0-
    9}{7}" type="tel" required /> <br>
16.     <!-- type = "radio" -->
17.         <label for="gender">Jenis Kelamin:</label>
18.         <label for="gender">Laki-laki</label>
19.         <input type="radio" name="gender" id="gender" value="male" />
20.         <label for="gender">Perempuan</label>
21.         <input type="radio" name="gender" id="gender" value="female" checked /> <br>
22.     <!-- type = "checkbox" -->
23.         <label for="binary">Bersedia menerima iklan melalui email?</label>
24.         <input type="checkbox" name="binary" id="binary" checked /> <br>
25.     <!-- type = "submit" -->
26.         <input type="submit" value="Submit" name="button1" id="button1">
27.     <!-- type = "reset" -->
28.         <input type="reset" value="Reset" name="button2" id="button2">
29.     </fieldset>
30. </form>
```

Hasil Eksekusi:

Data Registrasi
Username:
Password:
Tanggal Lahir: mm/dd/yyyy
Telepon: xxx-xxxxxxx
Jenis Kelamin: Laki-laki Perempuan
Bersedia menerima iklan melalui email?
Submit Reset

Gambar 2.21. Contoh *form* dengan elemen <input>

2.7.4. Elemen <select> dan <datalist>

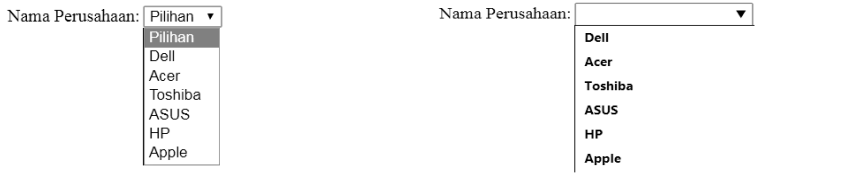
Elemen **<select>** dan **<datalist>** digunakan untuk membuat daftar pilihan berbentuk *drop-down*. Dengan elemen **<select>**, *user* hanya dapat menentukan satu pilihan berdasarkan daftar yang didefinisikan dengan menggunakan elemen **<option>**. Sedangkan dengan elemen **<datalist>**, *user* bukan

hanya dapat menentukan satu pilihan berdasarkan daftar yang sudah ada atau juga dapat mengisikan sendiri pilihan baru yang ia mau karena adanya penggunaan elemen **<input>**. Gambar 2.22 memperlihatkan contoh perbandingan elemen **<select>** dan **<datalist>**. Perhatikan bahwa nilai atribut **id** dalam elemen **<datalist>** harus sama atau terkoneksi dengan nilai atribut **id** dalam elemen **<input>**.

Skrip HTML:

```
1.  <form action="submit.html" method="get">
2.      <!-- elemen <select> -->
3.      <label for="companyName">Nama Perusahaan:</label>
4.      <select id="companyName" name="companyName">
5.          <option>Pilihan</option>
6.          <option value="dell">Dell</option>
7.          <option value="acer">Acer</option>
8.          <option value="toshiba">Toshiba</option>
9.          <option value="asus">ASUS</option>
10.         <option value="hp">HP</option>
11.         <option value="apple">Apple</option>
12.     </select>
13.     <br>
14.     <!-- elemen <datalist> -->
15.     <label for="companyName2">Nama Perusahaan:</label>
16.     <input id="companyName2" name="companyName2" list="companies" />
17.     <datalist id="companies">
18.         <option value="Dell">
19.         <option value="Acer">
20.         <option value="Toshiba">
21.         <option value="ASUS">
22.         <option value="HP">
23.         <option value="Apple">
24.     </datalist>
25. </form>
```

Hasil Eksekusi:



Gambar 2.22. Contoh perbandingan elemen <select> dan <datalist> dalam *form*

Catatan: Elemen **<datalist>** baru mulai ada sejak HTML5.

2.7.5. Elemen < textarea>

Elemen **<textarea>** digunakan untuk membuat kotak masukan berbentuk teks yang jumlahnya lebih dari satu baris. Dalam elemen **<textarea>**, atribut **rows** digunakan untuk menentukan jumlah baris kotak masukan, sedangkan atribut **cols** digunakan untuk menentukan lebarnya. Gambar 2.23 memperlihatkan contoh penggunaan elemen **<textarea>** dalam *form*.

Skrip HTML:

```
1.  <form action="submit.html" method="get">
2.      <label for="notes">Catatan:</label>
3.      <br>
4.      <textarea name="notes" id="notes" cols="40" rows="6"></textarea>
5.  </form>
```

Hasil Eksekusi:

Catatan:

Gambar 2.23. Contoh form dengan elemen <textarea>

2.8. Latihan Soal

Latihan 2.1

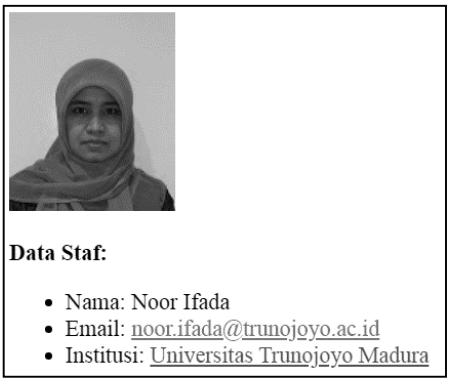
Buatlah skrip HTML untuk menampilkan *list* bersarang (*nested list*) seperti yang ditunjukkan dalam Gambar 2.24!

1. Daftar 1
    - Sub-daftar 1.1
    - Sub-daftar 1.2
2. Daftar 2
    - Sub-daftar 2.1
    - Sub-daftar 2.2
3. Daftar 3

Gambar 2.24. Tampilan *list* bersarang

Latihan 2.2

Buatlah skrip HTML untuk membuat halaman *web* yang memuat gambar foto dan *list* yang menampilkan informasi nama, alamat *email*, dan URL situs *web* seperti yang ditunjukkan pada Gambar 2.25! Sesuaikan gambar foto, data nama, alamat email dan URL situs *web* dengan gambar foto dan data pribadimu!



Gambar 2.25. Tampilan memuat gambar dan penggunaan *link*

Latihan 2.3

Buatlah skrip HTML untuk menghasilkan tabel yang memiliki judul tabel, judul baris, dan judul kolom seperti yang diperlihatkan dalam Gambar 2.26!

TABEL WARNA

	Merah	**Kuning**	**Biru**
Merah	Merah	Orange	Ungu
Kuning	Orange	Kuning	Hijau
Biru	Ungu	Hijau	Biru

Gambar 2.26. Tabel dengan gabungan judul baris dan kolom

Latihan 2.4

Buatlah skrip HTML untuk menghasilkan tabel yang menggabungkan sel kolom dan baris seperti yang diperlihatkan dalam Gambar 2.27!

STATISTIK KELAS

Mata Kuliah	Kelas	Jumlah Mahasiswa	
		Laki-laki	Perempuan
Dasar Pemrograman Web	A	20	22
	B	23	19
	C	21	21

Gambar 2.27. Tampilan tabel yang memiliki gabungan kolom dan baris

Latihan 2.5

Gunakan tabel untuk memformat tampilan *form* pada Gambar 2.21 agar terlihat menjadi rapi seperti yang diperlihatkan dalam Gambar 2.28!

Data Registrasi

Username:

Password:

Tanggal Lahir: mm/dd/yyyy

Telepon: xxx-xxxxxxx

Jenis Kelamin: Laki-laki Perempuan

Bersedia menerima iklan melalui email?

Submit Reset

Gambar 2.28. Contoh *layout* tampilan *form* pada Gambar 2.28 dengan menggunakan tabel

3

CSS

Bab ini membahas CSS sebagai bahasa *stysheet* yang digunakan untuk mengatur tampilan halaman *web*. Selain mempelajari dasar CSS, di sini kita juga akan mempelajari perintah-perintah CSS yang meliputi:

- pembuatan *selector*
- penempatan CSS
- penggunaan *the Box model*
- pembuatan *layout* halaman *web*

Setelah selesai mempelajari bab ini maka diharapkan kita dapat mendemonstrasikan aplikasi perintah-perintah CSS tersebut di atas dalam halaman *web*.

3.1 Dasar CSS

CSS (*Cascading Style Sheet*) merupakan salah satu bahasa *stylesheet*, yaitu bahasa yang digunakan untuk memformat konten atau membuat *layout* halaman *web* menjadi menarik untuk dilihat dan mudah untuk dikelola.

3.1.1 Struktur dan Sintaks CSS

Skrip CSS yang ditempatkan di dalam dokumen HTML harus diletakkan di dalam elemen **<style>**. Gambar 3.1 memperlihatkan struktur dalam sintaks CSS yang terdiri dari bagian *selector* dan deklarasi *rule-set*. Bagian *selector* digunakan untuk menentukan bagaimana deklarasi *rule-set* akan diterapkan (cek penjelasan detil dalam Sub-bab 3.2). Deklarasi *rule-set* terdiri dari properti yang digunakan beserta nilainya. Gambar 3.2 memperlihatkan contoh CSS sederhana dalam dokumen HTML yang mengatur jenis (**font-family**), warna (**color**), perataan (**text-align**) teks dalam *heading* **<h1>**.

```
Selector {
   properti: nilai; // setiap deklarasi rule-set selalu diakhiri dengan tanda titik koma (";")
}
```

Gambar 3.1. Struktur dalam sintaks CSS

Skrip HTML:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.      <title>StyleSheet Sederhana </title>
5.      <style>
6.          h1  {
7.             font-family: verdana;
8.             color: red;
9.             text-align: center;
10.         }
11.     </style>
12.     </head>
13.     <body>
14.         <h1>StyleSheet Sederhana</h1>
15.     </body>
16. </html>
```

Hasil Eksekusi:

Gambar 3.2. Contoh CSS sederhana dalam dokumen HTML

Tips: Daftar lengkap properti CSS dapat dilihat di https://www.w3schools.com/cssref/

Tips: Gunakan aplikasi *Developer Tools* dalam *web browser* untuk menginspeksi skrip CSS yang digunakan dalam dokumen HTML (cek Sub-bab 1.5.2).

3.1.2 Komentar dalam CSS

Penambahan komentar di dalam skrip CSS dapat dilakukan dengan dua cara. Pertama, komentar per baris dibuat dengan menambahkan tanda "**//**" di awal baris komentar. Kedua, komentar yang berjumlah lebih dari satu baris dibuat dengan menempatkan baris-baris komentar di dalam tanda pembuka "**/***" dan tanda penutup "***/**". Alternatif lain, komentar juga dapat dibuat dengan meletakkannya di dalam tanda pembuka "**<!--**" dan tanda penutup "**-->**". Namun, perlu untuk diperhatikan bahwa cara alternatif ini hanya dapat dilakukan jika posisi penempatan skrip CSS adalah di dalam dokumen HTML (Sub-bab 3.3.1 dan 3.3.2).

3.2 *Selector*

CSS memiliki tiga jenis *selector*, yaitu *selector* elemen HTML, *class*, dan ID.

3.2.1 *Selector* Elemen HTML

Selector elemen HTML digunakan untuk meredefinisikan elemen HTML. Dengan demikian maka nama *selector* adalah sama dengan elemen HTML. Gambar 3.3 memperlihatkan contoh penggunaan *selector* elemen HTML dalam CSS untuk meredefinisikan jenis (**font-family**), ukuran (**font-size**) dan warna (**color**) teks dalam elemen **<b>**.

Skrip HTML:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Selector HTML</title>
5.          <style>
6.              b {font-family:arial; font-size:14px; color:red}
7.          </style>
8.      </head>
9.      <body>
10.         <!-- memanggil selector b yang me-redefinisi-kan elemen <b> -->
11.         <b>Tulisan ini tebal karena menggunakan style CSS</b>
12.     </body>
13. </html>
```

Hasil Eksekusi:

Tulisan ini tebal karena menggunakan style CSS

Gambar 3.3. Contoh *selector* elemen HTML dalam CSS

3.2.2 *Selector Class*

Selector class digunakan untuk mendefinisikan *style* tanpa melakukan redefinisi elemen HTML. Selector *class* selalu diawali oleh tanda titik (“**.**”) dan diikuti dengan nama *class* yang penamaannya adalah bebas. Gambar 3.4 memperlihatkan contoh pembuatan *selector class* **.headline** yang untuk mendefinisikan jenis (**font-family**), ukuran (**font-size**) dan warna (**color**) teks. *Selector class* **.headline** kemudian diterapkan dalam elemen **<b>** dan **<i>**.

Skrip HTML:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Selector Class</title>
5.          <style>
6.              .headline {font-family:arial; font-size:14px; color:red}
7.          </style>
8.      </head>
9.      <body>
10.         <b class="headline">Tulisan ini tebal karena pengaruh selector class headline</b><br>
11.         <i class="headline">Tulisan ini dicetak miring karena selector class headline</i>
12.     </body>
13. </html>
```

Hasil Eksekusi:

Tulisan ini tebal karena pengaruh selector class headline
Tulisan ini dicetak miring karena selector class headline

Gambar 3.4. Contoh *selector* class dalam CSS

3.2.3 *Selector* ID

Selector ID digunakan untuk mendefinisikan *style* dengan ID unik dan biasanya digunakan untuk pembuatan *layer*. *Selector* ID selalu diawali oleh tanda tagar ("**#**") dan diikuti dengan nama ID yang penamaannya adalah bebas. Gambar 3.5 memperlihatkan contoh pembuatan *selector* ID **#text1** dan **#text2** untuk mendefinisikan perataan (**text-align**) dan warna (**color**) teks. *Selector* ID **#text1** dan **#text2** kemudian diterapkan dalam elemen ** **.

Skrip HTML:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Selector ID</title>
5.          <style>
6.              #text1 {text-align: center; color: red;}
7.              #text2 {text-align: left; color: blue;}
8.          </style>
9.      </head>
10.     <body>
11.         <div id="text1">Tulisan ini rata tengah dan berwarna merah karena pengaruh selector ID #t
12.         </div>
13.         <div id="text2">Tulisan ini rata kiri dan berwarna biru karena pengaruh selector ID #text
14.         </div>
15.     </body>
16. </html>
```

Hasil Eksekusi:

Tulisan ini rata tengah dan berwarna merah karena pengaruh selector ID #text1
Tulisan ini rata kiri dan berwarna biru karena pengaruh selector ID #text2

Gambar 3.5. Contoh *selector* ID dalam CSS

3.2.4 Pengelompokan *Selector*

Pengelompokkan *selector* dapat dilakukan ketika lebih dari satu *selector* memiliki deklarasi *rule-set* yang sama. Contoh dalam Gambar 3.6 memperlihatkan kesamaan deklarasi *rule-set* untuk properti **font-family**, **color** dan **background** antara *selector class* **.headlines**, **.sublines**, dan **.infotext**. Dengan kesamaan properti tersebut, maka pengelompokkan deklarasi *rule-set* dari ketiga *selector class* dapat dilakukan dan tanda koma (“,”) digunakan sebagai tanda pemisah antar *selector*. Sedangkan deklarasi *rule-set* yang berbeda antar *selector class* tetap dideskripsikan untuk secara terpisah untuk masing-masing *selector*.

Catatan: Pengelompokkan *selector* bersifat opsional.

```
1.  .headlines {
2.      font-family:arial;
3.      color:black;
4.      background:yellow;
5.      font-size:14pt;}
6.  .sublines {
7.      font-family:arial;
8.      color:black;
9.      background:yellow;
10.     font-size:12pt;}
11. .infotext {
12.     font-family:arial;
13.     color:black;
14.     background:yellow;
15.     font-size:10pt;}
```

Dikelompokkan menjadi

→

```
1.  .headlines, .sublines,
    .infotext {
2.      font-
    family:arial;
3.      color:black;
4.      background:yellow;}
5.  .headlines {
6.      font-size:14pt;}
7.  .sublines {
8.      font-size:12pt;}
9.  .infotext {
10.     font-size:10pt;}
```

Gambar 3.6. Contoh pengelompokkan *selector*

3.3 Lokasi Penempatan CSS

Implementasi CSS untuk dokumen HTML dapat dilakukan dengan tiga pilihan penempatan lokasi, yaitu di dalam elemen HTML, elemen **<head>**, dan *file* eksternal.

3.3.1 CSS di dalam elemen HTML

CSS ditempatkan di dalam elemen HTML, seperti yang ditunjukkan dalam Gambar 3.7, dilakukan jika kita hanya ingin mengimplementasikan CSS pada elemen HTML tertentu saja.

Skrip HTML:

```
1.  <!DOCTYPE>
2.  <html>
3.    <head>
4.      <title>Penggunaan CSS Elemen HTML</title>
5.    </head>
6.    <body>
7.       Ini adalah contoh
8.       <b style="font-size:16px;color:blue;"> bold </b> dengan menggunakan CSS.
9.    </body>
10. </html>
```

Hasil Eksekusi:

Ini adalah contoh **bold** dengan menggunakan CSS.

Gambar 3.7. Contoh penempatan CSS di dalam elemen HTML

3.3.2 CSS di dalam elemen <head>

CSS ditempatkan di dalam elemen **<head>**, seperti yang ditunjukkan dalam Gambar 3.8), dilakukan ketika kita ingin mengimplementasikan CSS untuk satu halaman *web*.

Skrip HTML:

```
1.  <!DOCTYPE>
2.  <html>
3.    <head>
4.      <title>Penggunaan CSS untuk satu halaman Web</title>
5.      <style>
6.          .headlines, .sublines {
7.              font-family:arial; color:blue;
8.              background:cyan; font-weight:bold;}
9.          .headlines {font-size:14pt;}
10.         .sublines {font-size:12pt;}
11.     </style>
12.    </head>
13.    <body>
14.         <span class="headlines">Selamat Datang</span><br>
15.         <div class="sublines"> Ini adalah contoh penggunaan CSS untuk satu halaman Web.<br></div>
16.    </body>
17. </html>
```

Hasil Eksekusi:

Selamat Datang
Ini adalah contoh penggunaan CSS untuk satu halaman Web.

Gambar 3.8. Contoh penempatan CSS di dalam elemen <head>

3.3.3 CSS di dalam *file* eksternal

Penempatan CSS dalam file eksternal adalah cara yang paling praktis untuk dilakukan karena skrip CSS dapat digunakan oleh lebih dari satu dokumen HTML. Dengan cara ini maka *link file* CSS dapat dicantumkan pada atribut **href** di dalam elemen **<link>** pada elemen **<head>** dokumen HTML (lihat Gambar 3.9).

Skrip HTML:

```
1.  <!DOCTYPE>
2.  <html>
3.      <head>
4.          <title>Penggunaan CSS Eksternal</title>
5.          <link rel=stylesheet href="style.css" type="text/css">
6.      </head>
7.      <body>
8.          <span class="headlines">Selamat Datang</span><br>
9.          <div class="sublines"> Ini adalah contoh penggunaan CSS Eksternal.<br></div>
10.     </body>
11. </html>
```

Hasil Eksekusi:

Selamat Datang
Ini adalah contoh penggunaan CSS Eksternal.

Gambar 3.9. Contoh penempatan CSS sebagai file eksternal

Skrip CSS:

```
1.  .headlines, .sublines {
2.      font-family:arial; color:blue;
3.      background:cyan; font-weight:bold;}
4.  .headlines {
5.      font-size:14pt;}
6.  .sublines {
7.      font-size:12pt;}
```

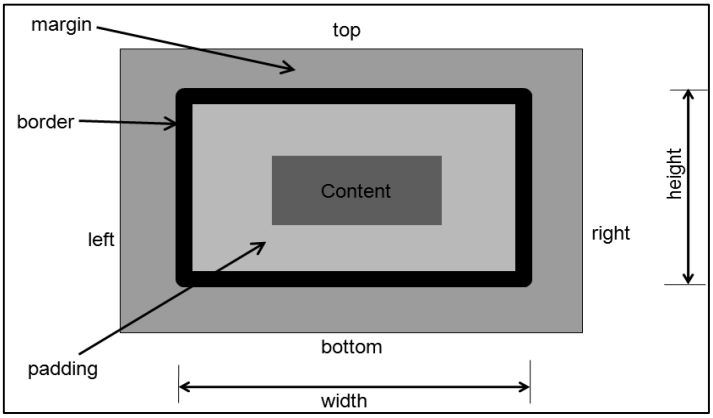
Gambar 3.10. Skrip CSS (style.css) yang digunakan dalam Gambar 3.9

3.4 *The Box model*

The Box model dalam CSS adalah sebuah kotak (*box*) yang mengelilingi setiap elemen HTML. Gambar 3.11 memperlihatkan *the Box model* yang terdiri dari empat komponen berikut:

- *Content*, yang menjadi area tempat teks dan gambar dapat muncul
- *Padding*, yang digunakan untuk membuat area kosong (berwarna transparan) di sekitar komponen *content*
- *Border*, yang digunakan untuk membuat garis tepi yang mengelilingi komponen *padding* dan *content*
- *Margin*, yang digunakan untuk membuat area kosong (berwarna transparan) di luar komponen *border*.

Contoh penggunaan *the Box model* dalam dokumen HTML dapat dilihat dalam Gambar 3.12.



Gambar 3.11. Komponen dalam The Box model

Skrip HTML:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Selector ID</title>
5.          <style>
6.              div {
7.                  margin: 2.0mm;
8.                  margin-top: 0.2cm;
9.                  margin-left: 0.1in;
10.                 padding: 10px;
11.                 padding-right: 10pc;
12.                 border-top-width: 0;
13.                 border-width: 8px;
14.                 border-color: blue;
15.                 border-style: dashed;
16.                 width: 30em;
17.                 height: 15ex;  }
18.         </style>
19.     </head>
20.     <body>
21.         <p>Contoh <i>the Box model:</i></p>
22.         <div style="background-color:yellow;">margin: 2.0mm; margin-top: 0.2cm; margin-
    left: 0.1in; padding: 4px; padding-right: 1pc; border-top-width: 0; border-width: 4px; border-
    color: blue; border-style: dashed; width: 30em; height: 20ex;
23.         </div>
24.     </body>
25. </html>
```

Hasil Eksekusi:

Contoh *the Box model:*

margin: 2.0mm; margin-top: 0.2cm; margin-left: 0.1in; padding: 4px; padding-right: 1pc; border-top-width: 0; border-width: 4px; border-color: blue; border-style: dashed; width: 30em; height: 20ex;

Gambar 3.12. Contoh skrip HTML untuk properti Box

3.5 CSS untuk *Layout* Halaman *Web*

Layout halaman *web* dapat dibuat dengan tiga pilihan metode, yaitu menggunakan elemen HTML **<table>**, elemen HTML **<frameset>**, dan kombinasi elemen HTML ** ** bersama dengan CSS.

Tabel 3.1. Metode pembuatan *layout* halaman *web*

Metode	Keterangan
Elemen HTML **<table>**	Umum diimplementasikan (terutama sebelum ada CSS)
Elemen HTML **<frameset>**	Metode yang tidak direkomendasikan
Element HTML ** ** dan CSS	Metode yang paling direkomendasikan

Sebagaimana yang diperlihatkan dalam Tabel 3.1, kombinasi elemen HTML ** ** dengan CSS adalah metode pembuatan *layout* halaman *web* yang paling direkomendasikan karena sesuai dengan standar HTML (Krause 2016). Dalam hal ini, elemen HTML ** ** digunakan sebagai kontainer per bagian *layout*, sedangkan CSS digunakan untuk mengatur posisi masing-masing ** **.

3.5.1 Properti float dan clear untuk Pengaturan Posisi

Properti **float** pada CSS digunakan untuk mengatur posisi apung suatu elemen, sedangkan properti **clear** digunakan untuk mengatur posisi elemen terhadap konten yang memiliki elemen **float**. Daftar nilai kedua properti tersebut dan keterangannya dapat dilihat dalam Tabel 3.2.

Tabel 3.2. Daftar properti float dan clear untuk pengaturan posisi

Properti	Nilai	Keterangan
float	**left**	Elemen akan berada di sisi kiri (pada baris yang sama)
	right	Elemen akan berada di sisi kanan (pada baris yang sama)
	none	Elemen akan berada sebagaimana posisinya di dalam skrip HTML (*default*)

clear	**left**	Tidak ada elemen yang boleh berada di sisi kiri (pada baris yang sama dengan konten)
	right	Tidak ada elemen yang boleh berada di sisi kanan (pada baris yang sama dengan konten)
	both	Tidak ada elemen yang boleh berada di baris yang sama (dari kedua sisi kiri dan kanan)
	none	Elemen boleh berada di baris yang sama pada kedua sisi konten (*default*)

Gambar 3.13 memperlihatkan contoh penggunaan properti **float** untuk mengatur *layout* dua buah kotak agar mengapung di sisi kiri (**left**) dan kanan (**right**) halaman *web*. Skrip CSS untuk pengaturan *layout* ada dalam Gambar 3.14.

Skrip HTML:

```
1.  <!DOCTYPE HTML>
2.  <html>
3.      <head>
4.          <meta charset="utf-8"/>
5.          <title>div using CSS</title>
6.          <link href="style_div.css" rel="stylesheet" type="text/css" />
7.      </head>
8.      <body>
9.          <div class="sidebox" style="float:left">
10.             <h1>Box One</h1>
11.             <p>Ini adalah konten dari box pertama</p>
12.         </div>
13.         <div class="sidebox" style="float:right">
14.             <h1>Box Two</h1>
15.             <p>Ini adalah konten dari box kedua</p>
16.         </div>
17.     </body>
18. </html>
```

Hasil Eksekusi:

Gambar 3.13. Contoh properti float dalam dokumen HTML

Skrip CSS:

```
1.  body
2.      {background-color: #9999FF;
3.      font-family: Verdana, geneva, Arial, Helvetica, sans-serif;
4.      margin:0px;
5.  }
6.  div.sidebox{
7.      border-style: solid;
8.      border-width: 4px;
9.      border-color: #003399;
10.     width: 160px;
11.     margin: 10px;
12.     background: white;
13.     }
14. div.sidebox h1{
15.     background-color: #003399;
16.     font-size: 15pt;
17.     color: white;
18.     margin: 0px;
19.     text-align: center;
20.     }
21. div.sidebox p{
22.     margin: 10px;
23.     }
```

Gambar 3.14. Skrip CSS (style_div.css) yang digunakan dalam Gambar 3.13

3.5.2 Contoh *Layout* Halaman *Web*

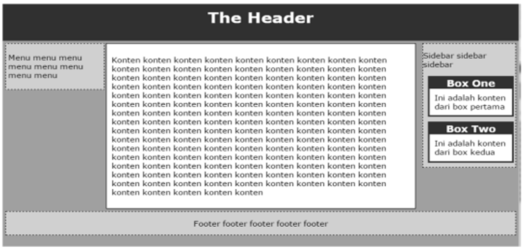
Gambar 3.15 memperlihatkan contoh pembuatan *layout* halaman *web* yang memiliki lima bagian kontainer, yaitu *Header*, *Menu*, *Content*, *Side bar*, dan *Footer*. Skrip CSS untuk pengaturan *layout* ada dalam Gambar 3.16.

Skrip HTML:

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8"/>
        <title>CSS - Layout</title>
        <link href="style_layout.css" rel="stylesheet" type="text/css" />
    </head>
    <body>
        <div class="header">
            <h1>The Header</h1>
        </div>
        <div class="menu">
            <p>Menu menu menu menu menu menu menu menu</p>
        </div>
        <div class="sidebar">
            <p>Sidebar sidebar sidebar</p>
            <div class="sidebox" >
                <h1>Box One</h1>
                <p>Ini adalah konten dari box pertama</p>
            </div>
            <div class="sidebox" >
                <h1>Box Two</h1>
                <p>Ini adalah konten dari box kedua</p>
            </div>
        </div>
        <div class="content">
            <p>Konten konten konten konten konten konten konten konten konten konten konten konten
konten konten konten konten konten konten konten konten konten konten konten konten konten konten
konten konten konten konten konten konten konten konten konten konten konten konten konten konten
konten konten konten konten konten konten konten konten konten konten konten konten konten konten
konten konten konten konten konten konten konten konten konten konten konten konten konten konten
konten konten konten konten konten konten konten konten konten konten konten konten konten konten
konten konten konten konten konten konten konten konten konten konten konten konten konten konten
konten konten konten konten konten konten konten konten konten konten konten konten konten konten
konten konten konten konten konten konten konten konten konten konten konten konten konten konten
konten konten konten konten konten konten konten konten konten konten konten konten konten konten
konten konten</p>
        </div>
        <div class="footer">
            <p>Footer footer footer footer footer</p>
        </div>

    </body>
</html>
```

Hasil Eksekusi:



Gambar 3.15. Contoh *layout* halaman *web*

Skrip CSS:

```
1.  body
2.      {background-color: #9999FF;
3.      font-family: VErdana, geneva, Arial, Helvetica, sans-serif;
4.      margin:0px;
5.      min-width: 600px;
6.      }
7.  div.header {
8.      height: 80px;
9.      text-align: center;
10.     background-color: #003399;
11.     color: white;
12.     }
13. div.header h1 {
14.     margin: 0px;
15.     padding: 10px;
16.     }
17. div.content {
18.     padding: 10px;
19.     margin-top: 5px;
20.     margin-left: 200px;
21.     margin-right: 200px;
22.     border:1px solid black;
23.     background-color: white;
24.     }
25. div.menu {
26.     margin: 5px;
27.     padding: 5px;
28.     border:1px dashed black;
29.     background-color: #CCCCFF;
30.     float: left;
31.     width: 180px;
32. }
33. div.sidebar {
34.     margin: 5px;
35.     border: 1px dashed black;
36.     background-color: #CCCCFF;
37.     float: right;
38.     width: 180px;
39.     }
40. div.footer {
41.     margin: 5px;
42.     border: 1px dashed black;
43.     background-color: #CCCCFF;
44.     text-align: center;
45.     clear: both;
46.     height: 50px;
47.     }
48. div.sidebox{
49.     border-style: solid;
50.     border-width: 4px;
51.     border-color: #003399;
52.     width: 160px;
53.     margin: 10px;
54.     background: white;
55.     }
56. div.sidebox h1{
57.     background-color: #003399;
58.     font-size: 15pt;
59.     color: white;
60.     margin: 0px;
61.     text-align: center;
62. }
63. div.sidebox p{
64.     margin: 10px;
65.     }
```

Gambar 3.16. Skrip CSS (style_layout.css) yang digunakan dalam Gambar 3.15

3.6 Latihan Soal

Latihan 3.1

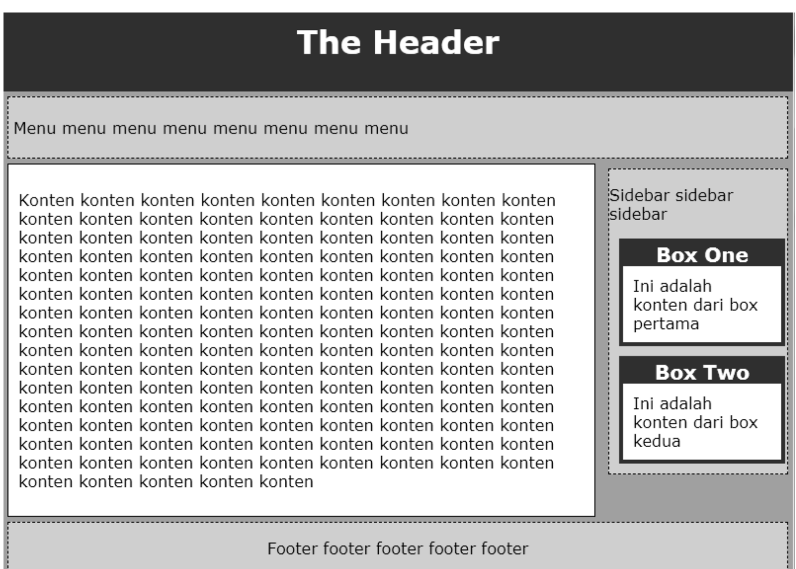
Formatlah *style* tabel pada Gambar 2.17 dengan menggunakan CSS!

Latihan 3.2

Gunakan CSS sebagai pengganti penggunaan tabel untuk mengatur *layout form* pada Gambar 2.2017!

Latihan 3.3

Ubahlah *file* CSS pada Gambar 3.15 agar *layout* halaman *web* berubah sehingga bagian kontainer *Menu* dan *Content* berubah menjadi seperti Gambar 3.17!



Gambar 3.17. *Layout* baru halaman *web* untuk Gambar 3.15

4

XML

Bab ini membahas XML sebagai bahasa markah yang juga digunakan untuk membuat halaman *web*. Di sini kita akan mempelajari perintah-perintah XML yang meliputi:

- pembuatan dokumen XML
- memformat dokumen XML untuk ditampilkan pada *web browser*

Setelah selesai mempelajari bab ini maka diharapkan kita dapat mendemonstrasikan aplikasi perintah-perintah XML tersebut di atas.

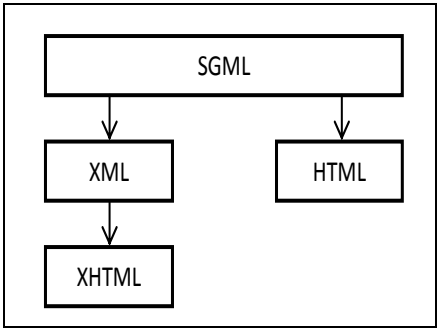
4.1 Dasar XML

Sama halnya dengan HTML, XML (*eXtensible Markup Language*) juga merupakan salah satu bahasa markah. XML memiliki peran yang penting dalam berbagai sistem IT (*Information Technology*) karena seringkali digunakan untuk API (*Application Programming Interface*)[4]. Oleh sebab itu, pengetahuan dasar mengenai struktur dokumen XML juga sangat penting untuk dimiliki ketika kita sedang belajar pemrograman *web*. Di

[4] Perbandingan statistik penggunaan XML untuk API dapat dilihat di: https://www.programmableweb.com

dalam bab ini, kita hanya akan membahas beberapa karakteristik penting dari XML, terutama untuk menunjukkan perbedaannya dengan HTML, bagaimana pembuatan dokumen dilakukan dan cara memformatnya untuk ditampilkan dalam *web browser*.

XML, sebagaimana juga HTML, adalah bahasa markah yang didasarkan pada SGML (*Standard Generalized Markup Language*) yang dikembangkan pada tahun 1986 (Krause 2016). Gambar 4.1 memperlihatkan diagram pengembangan bahasa markah yang berbasis SGML. Selain XML dan HTML yang merupakan turunan langsung dari SGML, terdapat pula XHTML (*eXtensible HyperText Markup Language*) yang identik dengan HTML namun didefinisikan sebagai aplikasi XML.



Gambar 4.1. Pengembangan bahasa markah berbasis SGML (Krause 2016)

Di samping persamaan yang dimiliki, XML dan HTML juga memiliki perbedaan karena mereka memang digunakan untuk tujuan yang berbeda. XML digunakan untuk distribusi data sehingga ia difokuskan pada arti sebenarnya dari data tersebut, sedangkan HTML digunakan untuk mendeskripsikan bagaimana data ditampilkan. Selain itu, *tag* pada XML tidak dibatasi dengan penggunaan *tag* normal seperti halnya HTML karena pemrogram dapat mendefinisikan sendiri *tag* yang ingin digunakan sesuai dengan data yang dibuatnya. Fitur ini adalah fitur khusus yang tidak ada dalam bahasa pemrograman *web* selain XML.

4.1.1 Struktur dan Sintaks XML

Elemen-elemen di dalam suatu dokumen XML membentuk sebuah struktur *tree* (lihat Gambar 4.2). Dokumen XML harus selalu diawali oleh elemen *root* yang merupakan *parent* dari seluruh elemen yang ada di dalam dokumen dan bercabang pada elemen *child* (dan *sub-child*). Perhatikan contoh skrip dokumen XML sederhana tentang data identitas siswa yang diperlihatkan dalam Gambar 4.3. *Root* dari dokumen XML ini adalah elemen **<academic>**, yang memiliki elemen **<student>** sebagai *child*. Sub-child dari elemen **<student>** adalah elemen **<id>** dan **<name>**.

```
<root>
   <child>
      <sub-child>.....</sub-child>
   </child>
</root>
```

Gambar 4.2. Struktur *tree* pada XML

Skrip XML:

```
1.  <?xml version="1.0" encoding="UTF-8"?> <!-- Deklarasi XML: XML version (1.0) -->
2.  <academic> <!-- Elemen root (<academic>) -->
3.      <student> <!-- <student>: child dari root <academic> -->
4.          <id>12345</id> <!-- <id>: sub-child dari child <student> -->
5.          <name>Almira Wijaya</name> <!-- <name>: sub-child dari child <student> -->
6.      </student>
7.      <student>
8.          <id>12346</id>
9.          <name>Danuar Aldi</name>
10.     </student>
11. </academic>
```

Gambar 4.3. Contoh XML sederhana

Untuk menghasilkan dokumen XML yang memiliki format yang benar, terdapat beberapa aturan sintaks yang harus dipenuhi yaitu:

- Dokumen XML harus memiliki sebuah elemen *root*
- Seluruh elemen XML harus memiliki *tag* penutup
- Tag XML adalah *case-sensitive*
- Elemen XML harus disarangkan dengan tepat
- Nilai atribut XML harus diberi tanda *quote* (“ “).

Aturan penamaan tag dalam dokumen XML meliputi beberapa ketentuan berikut (Fawcet, Quin dan Ayers 2012):

- Harus dimulai dengan sebuah huruf atau simbol garis bawah ("_")
- Tidak boleh diawali dengan kata "xml" (atau "XML", "Xml", dan seterusnya)
- Tidak boleh ada spasi.

Gambar 4.4 dan Gambar 4.5 memperlihatkan contoh sintaks dan penamaan elemen berdasarkan aturan yang ada dalam dokumen XML.

Skrip XML:

```
1.  <!-- ATURAN SINTAKS: Elemen XML harus memiliki tag penutup -->
2.  <name>Almira Wijaya</name> <!-- Contoh yang benar -->
3.  <name>Almira Wijaya<name> <!-- Contoh yang salah -->
4.  <name>Almira Wijaya <!-- Contoh yang salah -->
5.
6.  <!-- ATURAN SINTAKS: Tag XML adalah case-sensitive -->
7.  <name>Almira Wijaya</name> <!-- Contoh yang benar -->
8.  <Name>Almira Wijaya</name> <!-- Contoh yang salah -->
9.
10. <!-- ATURAN SINTAKS: Elemen XML harus disarangkan dengan tepat -->
11. <!-- Contoh yang benar -->
12. <student>
13.     <id>12345</id>
14.     <name>Almira Wijaya</name>
15. </student>
16.
17. <!-- ATURAN SINTAKS: Nilai atribut XML harus diberi tanda quote (" ") -->
18. <!-- Contoh yang benar -->
19. <student gender="female">
20.     <id>12345</id>
21.     <name>Almira Wijaya</name>
22. </student>
```

Gambar 4.4. Contoh aturan sintaks dalam dokumen XML

Skrip XML:

```
1.  <!-- ATURAN PENAMAAN ELEMEN: Diawali oleh sebuah huruf atau simbol garis bawah ("_") -->
2.  <name>Almira Wijaya</name> <!-- Contoh yang benar -->
3.  <_name>Almira Wijaya</_name> <!-- Contoh yang benar -->
4.  <@name>Almira Wijaya</@name> <!-- Contoh yang salah -->
5.  <1name>Almira Wijaya</1name) <!-- Contoh yang salah -->
6.
7.  <!--
     ATURAN PENAMAAN ELEMEN: Tidak boleh diawali oleh kata "xml" (atau "XML", atau "Xml", dan seterusn
    ya -->
8.  <XMLname>Almira Wijaya</XMLname> <!-- Contoh yang salah -->
9.
10. <!-- ATURAN PENAMAAN ELEMEN: Tidak boleh ada spasi -->
11. <s name>Almira Wijaya</s name> <!-- Contoh yang salah -->
```

Gambar 4.5. Contoh aturan penamaan elemen dalam dokumen XML

4.1.2 Komentar dalam XML

Penambahan komentar di dalam skrip XML dapat dilakukan dengan meletakkan baris-baris komentar di dalam tanda pembuka "**<!--** " dan tanda penutup "**-->**".

4.2 Memformat Tampilan XML

Jika skrip dokumen XML ditampilkan secara langsung pada *web browser*, maka tampilannya akan sama persis dengan skrip yang dibuat dan tidak memiliki format tampilan apapun, yang membuatnya menjadi tidak menarik untuk dilihat. Hal ini berbeda dengan skrip HTML yang secara langsung dapat memformat tampilannya pada *web browser*. Dari skrip dokumen XML pada Gambar 4.3, maka hasil tampilan pada *web browser* adalah seperti dalam Gambar 4.6. Perhatikan bahwa baris komentar yang ada di dalam Gambar 4.3 telah terlebih dahulu sengaja dihilangkan untuk menyederhanakan contoh tampilan.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<academic>
  ▼<student>
     <id>12345</id>
     <name>Almira Wijaya</name>
   </student>
  ▼<student>
     <id>12346</id>
     <name>Danuar Aldi</name>
   </student>
 </academic>
```

Gambar 4.6. Contoh tampilan dokumen XML (Gambar 4.3 tanpa komentar) pada *web browser*

Jika terdapat suatu kesalahan di dalam dokumen XML, maka *web browser* akan menyediakan suatu pesan bantuan yang memberitahukan di mana kesalahan tersebut terjadi dan menampilkan potongan skrip yang salah (lihat Gambar 4.7).

The XML page cannot be displayed

Cannot view XML input using XSL style sheet. Please correct the error and then click the Refresh button, or try again later.

End tag 'student' does not match the start tag 'name'. Error processing resource 'file:///C:/Documents and Settings/EVE/Des...

```
</student>
--^
```

Gambar 4.7. Contoh tampilan dokumen XML yang memiliki kesalahan pada *web browser*

Untuk dapat menampilkan dokumen XML secara menarik pada *web browser* sebagai halaman HTML, maka kita perlu menggunakan bahasa *stylesheet* seperti CSS atau XSLT.

4.2.1 Memformat Tampilan dengan CSS

Memformat tampilan dokumen XML dengan CSS dapat dilakukan dengan cara yang sama dengan implementasi CSS pada HTML. Kita tinggal membuatkan *link* skrip CSS ke skrip dokumen XML. Sebagai contoh, Gambar 4.8 menunjukkan hasil skrip dokumen XML pada Gambar 4.3 yang ditambahkan dengan *link file* "**xml_style.css**" (Gambar 4.9).

Skrip XML:

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <?xml-stylesheet type="text/css" href="xml_style.css"?>
3.  <academic>
4.      <student>
5.          <id>12345</id>
6.          <name>Almira Wijaya</name>
7.      </student>
8.      <student>
9.          <id>12346</id>
10.         <name>Danuar Aldi</name>
11.     </student>
12. </academic>
```

Hasil Eksekusi:

12345 Almira Wijaya

12346 Danuar Aldi

Gambar 4.8. Contoh penggunaan CSS untuk format tampilan dokumen XML

Skrip CSS:

```
1.  academic {
2.      background-color: #ffffff;
3.      width: 100%;
4.      }
5.  student {
6.      display: block;
7.      background-color: #DDDDDD;
8.      margin-bottom: 5pt;
9.      }
10. id {
11.     background-color: #999999;
12.     margin-bottom: 12pt;
13.     }
14. name {
15.     color: #0000FF;
16.     font-size: 12pt;
17.     }
```

Gambar 4.9. Skrip CSS (xml_style.css) yang digunakan dalam Gambar 4.8

Namun perlu diingat bahwa memformat tampilan dokumen XML dengan menggunakan CSS adalah tidak direkomendasikan. Metode yang lebih baik untuk diterapkan adalah dengan menggunakan XSLT.

4.2.2 Memformat Tampilan dengan XSLT

XSLT (*eXtensible Stylesheet Language Transformation*) adalah bahasa stylesheet yang dikhususkan untuk memformat tampilan dokumen XML menjadi halaman HTML[5]. Memformat tampilan dokumen XML dengan XSLT dapat dilakukan dengan membuatkan *link* skrip XSL ke skrip dokumen XML. Sebagai contoh, Gambar 4.10 menunjukkan hasil skrip dokumen XML pada Gambar 4.3 yang ditambahkan dengan *link file* "**xml_style.xsl**".

Skrip XML:

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <?xml-stylesheet type="text/xsl" href="xml_style.xsl"?>
3.  <academic>
4.      <student>
5.          <id>12345</id>
6.          <name>Almira Wijaya</name>
7.      </student>
8.      <student>
9.          <id>12346</id>
10.         <name>Danuar Aldi</name>
11.     </student>
12. </academic>
```

Hasil Eksekusi:

ID	Name
12345	Almira Wijaya
12346	Danuar Aldi

Gambar 4.10. Contoh penggunaan XSL untuk format tampilan dokumen XML

Gambar 4.11 menjabarkan file XSL yang digunakan dalam Gambar 4.10. Oleh karena skrip XSL adalah dokumen XML, maka ia selalu diawali dengan deklarasi XML: **<?xml version="1.0"**

---

5 *Web browser* Chrome secara *default* memblokir dokumen XSLT

encoding="UTF-8"?>. Selanjutnya, elemen **<xsl:stylesheet>** bersama dengan atribut **version** dan **xmlns** (XML *namespace*) mendefinisikan bahwa dokumen XML adalah merupakan dokumen XSLT. Elemen **<xsl:template>** digunakan untuk membangun *template* dari tampilan berbentuk halaman HTML yang diinginkan dan atribut **match="/"** menunjukkan bahwa *template* digunakan untuk seluruh dokumen XML. Dalam hal ini penerapan *template* dimulai dari *root* **<academic>**. Elemen **<xsl:for-each>** menandakan bahwa kita dapat melakukan *looping* di dalam XSLT. Di sini *looping* dilakukan untuk setiap elemen yang ada di dalam elemen **<student>** yang merupakan *child* dari *root* **<academic>**. Sedangkan elemen **<xsl:value-of>** digunakan untuk mengekstrak nilai dari elemen, yaitu nilai dari elemen **<id>** dan **<name>** yang merupakan *sub-child* dari elemen **<student>**).

Skrip XSL:

```
1.  <?xml version="1.0"  encoding="UTF-8"?>
2.  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.  <xsl:template match="/">
4.      <html>
5.          <body>
6.              <table border="1">
7.                  <tr style="background-color:#EEEEEE;">
8.                      <th>ID</th>
9.                      <th>Name</th>
10.                 </tr>
11.                 <xsl:for-each select="academic/student">
12.                     <tr>
13.                         <td><xsl:value-of select="id"/></td>
14.                         <td><xsl:value-of select="name"/></td>
15.                     </tr>
16.                 </xsl:for-each>
17.             </table>
18.         </body>
19.     </html>
20. </xsl:template>
21. </xsl:stylesheet>
```

Gambar 4.11. Skrip XSL (xml_style.xsl) yang digunakan dalam Gambar 4.10

4.3 Latihan Soal

Latihan 4.1

Cari empat kesalahan sintaks yang terdapat dalam skrip XML pada Gambar 4.12!

Skrip XML:

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <course cid="c1">
3.      <title>Math 1</title>
4.      <description>Introduction to Math<description>
5.  </course>
6.  <course cid=c2>
7.      <title>Math 2</title>
8.      <description>Intermediate Math</description>
9.  </course>
10. <course cid="c3">
11.     <title>Math 3</title>
12.     <Description>Advance Math</description>
13. </course>
```

Gambar 4.12. Skrip XMLdengan empat kesalahan sintaks

Latihan 4.2

Perbaharui skrip XML pada Gambar 4.3 untuk menambahkan data dosen berikut:

- Dosen 1 memiliki nomer identitas A123 dan bernama Beni Bernardi
- Dosen 2 memiliki nomer identitas A124 dan bernama Catur Cahyono

Latihan 4.3

Buatlah skrip XSL yang baru agar hasil eksekusi skrip XML Gambar 4.10 menjadi seperti Gambar 4.13!

ID :12345
Name: Almira Wijaya
ID :12346
Name: Danuar Aldi

Gambar 4.13. Tampilan baru skrip dalam Gambar 4.3 pada web browser

5

JAVASCRIPT

Bab ini membahas JavaScript sebagai bahasa pemrograman yang digunakan dalam *web* statis. Selain mempelajari dasar JavaScript, di sini kita akan mempelajari perintah-perintah JavaScript yang meliputi:

- penempatan JavaScript
- pembuatan variabel dan tipe data
- penggunaan operator
- penggunaan struktur kondisi dan struktur perulangan
- pembuatan fungsi, *built-in objects*, *RegExp*
- penggunaan HTML DOM
- mengakses elemen HTML dengan JavaScript
- pembuatan kotak dialog
- penggunaan *event*
- validasi *form* dengan JavaScript

Setelah selesai mempelajari bab ini maka diharapkan kita dapat mendemonstrasikan aplikasi perintah-perintah JavaScript tersebut di atas.

5.1 Dasar JavaScript

JavaScript merupakan bahasa pemrograman yang digunakan untuk mengatur perilaku halaman *web* pada *web browser*. JavaScript adalah pemrograman berbasis *object* yang berarti bahwa ia memiliki *object*, *method* dan *properties*. Namun tidak seperti bahasa pemrograman Java, *object* pada JavaScript bukanlah merupakan *class*.

5.1.1 Struktur dan Sintaks JavaScript

Skrip JavaScript yang ditempatkan di dalam dokumen HTML harus diletakkan di dalam elemen **<script>** dan setiap perintah JavaScript harus diakhiri dengan simbol titik koma ("**;**"). Gambar 5.1 memperlihatkan contoh pembuatan JavaScript sederhana untuk menampilkan teks "Membuat JavaScript Sederhana" pada *web browser*.

Skrip HTML dan JavaScript:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head><title>JavaScript Sederhana</title></head>
4.      <body>
5.          <p>
6.              <script >
7.                  document.write("Membuat JavaScript Sederhana!"); // menampilkan tulisan di halamai
8.              </script>
9.          </p>
10.     </body>
11. </html>
```

Hasil Eksekusi:

Membuat JavaScript Sederhana!

Gambar 5.1. Contoh JavaScript sederhana

Catatan: Sintaks JavaScript bersesuaian dengan C dan mirip dengan Java dan C#, namun sedikit berbeda dari Python.

Tips: Gunakan aplikasi *Developer Tools* dalam *web browser* untuk menginspeksi skrip JavaScript yang digunakan dalam dokumen HTML (cek Sub-bab 1.5.2).

5.1.2 Komentar dalam JavaScript

Penambahan komentar di dalam skrip JavaScript, yang posisi penempatannya di dalam dokumen HTML, dapat dilakukan dengan cara menggunakan tanda "//" di awal baris komentar. Alternatif lain, baris-baris komentar juga dapat dibuat dengan menempatkannya di dalam tanda pembuka "**/***" dan tanda penutup "***/**". Namun, perlu untuk diperhatikan bahwa cara alternatif ini hanya dapat dilakukan jika posisi penempatan JavaScript adalah di dalam *file* eksternal (Sub-bab 5.2.3).

5.2 Lokasi Penempatan JavaScript

Implementasi JavaScript untuk dokumen HTML dapat dilakukan dengan pilihan penempatan di tiga lokasi, yaitu di dalam: elemen **<head>**, elemen **<body>**, dan *file* eksternal.

5.2.1 JavaScript di dalam Elemen <body>

JavaScript ditempatkan di dalam elemen **<body>**, seperti yang ditunjukkan dalam Gambar 5.1, dilakukan jika kita memerlukan JavaScript untuk selalu dieksekusi setiap kali dokumen HTML dibuka di *web browser*.

5.2.2 JavaScript di dalam Elemen <head>

JavaScript ditempatkan di dalam elemen **<head>**, seperti yang ditunjukkan dalam Gambar 5.2, dilakukan jika kita

memerlukan JavaScript untuk dipanggil ketika ada *event* yang berfungsi sebagai *trigger*.

Skrip HTML dan JavaScript:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>JavaScript Sederhana</title>
5.          <script>
6.              document.write("Membuat JavaScript Sederhana!");
7.          </script>
8.      </head>
9.      <body>
10.     </body>
11. </html>
```

Gambar 5.2. Contoh penempatan JavaScript di dalam tag <head>

5.2.3 Penempatan JavaScript di dalam *File* Eksternal

Penempatan JavaScript di dalam *file* eksternal adalah cara yang paling praktis untuk dilakukan karena hal ini menjadikan skrip JavaScript dapat digunakan oleh lebih dari satu dokumen HTML. Dengan cara ini maka *link file* JavaScript dapat dicantumkan sebagai nilai dari atribut **src** dalam elemen **<script>** dokumen HTML. Gambar 5.3 memperlihatkan contoh JavaScript yang ditempatkan di dalam *file* eksternal dengan *link* yang ditempatkan di dalam elemen **<body>**.

Skrip HTML dan JavaScript:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head></head>
4.      <body>
5.          <p>
6.              <script src="eksternal.js">
7.              </script>
8.          </p>
9.      </body>
10. </html>
```

Gambar 5.3. Contoh link penempatan JavaScript sebagai file eksternal

Skrip JavaScript:

```
1.  document.write("Membuat JavaScript Sederhana!");
```

Gambar 5.4. Skrip pada *file* eksternal.js yang digunakan dalam Gambar 5.3

5.3 Variabel dan Tipe Data

Variabel di dalam JavaScript merupakan kontainer yang digunakan untuk menyimpan nilai suatu data. Terdapat beberapa aturan penamaan variabel di dalam JavaScript, yaitu:

- Nama variabel adalah *case-sensitive* ("x" dan "X" adalah variabel yang berbeda)
- Nama variabel hanya boleh diawali dengan suatu huruf, karakter garis bawah ("_"), ataupun simbol dollar ("**$**")
- Tidak menggunakan kata-kata khusus dalam JavaScript seperti **abstract**, **break**, **case**, **default**, **else**, **false**, **int**, **long**, dan lain-lain (daftar lengkap dapat dilihat di https://www.w3schools.com/js/js_reserved.asp).

Di dalam JavaScript terdapat enam macam tipe data yaitu:

- Numerik, yang merepresentasikan angka seperti bilangan bulat, real, dan exponensial
- *String*, yang merepresentasikan teks seperti karakter, kata, atau kalimat
- *Array*, yang berbentuk kontainer untuk menampung satu atau lebih data (dengan berbagi tipe)
- *Boolean*, yang merepesentasikan nilai *true* (benar)atau salah *false* (salah)
- *Null*, yang merepresentasikan suatu nilai yang kosong atau tidak ada
- *Undefined*, yang merepesentasikan bahwa suatu variabel tidak memiliki nilai.

Tabel 5.1 memperlihatkan contoh deklarasi variabel masing-masing tipe data.

Tabel 5.1. Contoh deklarasi variabel dalam JavaScript

Tipe Data	Contoh
Numerik	var angka1 = 17; angka2 = 3.78;
String	var nama = 'eve';
Array	var arrDayNames = ["Sunday", "Monday", "Tuesday" "Wednesday", "Thursday", "Friday", "Saturday"]; arrDayNames[2] = "Tuesday";
Boolean	var status = true;
Null	var huruf = null;
Undefined	var huruf1; huruf2 = undefined;

Catatan: Variabel dalam JavaScript tidak perlu dideklarasikan tipe datanya (*weakly typed*), sama seperti Python, namun berbeda dengan Java/C# (*strongly typed languages*).

5.4 Operator

Operator dalam JavaScript meliputi operator: aritmatika, penugasan, logika, perbandingan, *bitwise*, dan *string*.

5.4.1 Operator Aritmatika (*Arithmetic*)

Operator aritmatika dalam JavaScript digunakan untuk melakukan operasi aritmatika pada angka. Daftar operator aritmatika dapat dilihat dalam Tabel 5.2.

Tabel 5.2. Daftar Operator Aritmatika dalam JavaScript

Simbol	Deskripsi	Contoh (x = 10)	Hasil
+	Penjumlahan	x + 5	15
–	Pengurangan	x – 5	5
*	Perkalian	x * 5	50
/	Pembagian	x / 5	2
%	Modulus	x % 5	0
++	*Increment* (menambahkan nilai variabel dengan 1)	x++	11

--	*Decrement* (mengurangi nilai variabel dengan 1)	x--	9

5.4.2 Operator Penugasan (*Assignment*)

Operator penugasan dalam JavaScript digunakan untuk memberikan nilai pada suatu variabel. Daftar operator penugasan dapat dilihat dalam Tabel 5.3.

Tabel 5.3. Daftar Operator Penugasan dalam JavaScript

Simbol	Contoh	Ekivalensi
=	x = y	x = y
+=	x += y	x = x + y
−=	x −= y	x = x − y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

5.4.3 Operator Logika (*Logical*)

Operator logika dalam JavaScript digunakan untuk menentukan logika antar variabel atau nilai. Daftar operator logika dapat dilihat dalam Tabel 5.4.

Tabel 5.4. Daftar Operator Logika dalam JavaScript

Operator	Nama	Deskripsi	Contoh (x = 1 dan y = 2)
&&	Logika AND	Bernilai *true* jika kedua kondisi (di sisi kiri dan kanan operator) adalah *true*	(x < 2 && y > 1) bernilai *true*
??	Logika OR	Bernilai *true* jika salah satu kondisi (di sisi kiri	(x < 2 && y > 2) bernilai *true*

		atau kanan operator) adalah *true*	
!	Logika NOT	Kebalikan dari nilai kondisi	!(x > y) bernilai *true*

5.4.4 Operator Perbandingan

Operator perbandingan dalam JavaScript digunakan dalam pernyataan logika untuk menentukan kesamaan atau perbedaan antar variabel atau nilai. Daftar operator perbandingan dapat dilihat dalam Tabel 5.5.

Tabel 5.5. Daftar Operator Perbandingan dalam JavaScript

Operator	Deskripsi	Contoh
==	Sama dengan	1==2 bernilai *false* 2==2 bernilai *true* 2==”2” bernilai *true*
===	Sama nilai dan sama tipe data	2===2 bernilai *true* 2===”2” bernilai *false*
!=	Tidak sama dengan	1!=2 bernilai *true* 2!=2 bernilai *false*
!==	Tidak sama nilai atau tidak sama tipe data	1!==2 bernilai *false* 2!==2 bernilai *false* 2!==”2” bernilai *true*
>	Lebih besar dari	1>2 bernilai *false* 2>2 bernilai *false* 3>2 bernilai *true*
<	Lebih kecil dari	1 < 2 bernilai *true* 2 < 2 bernilai *false* 3 < 2 bernilai *false*
>=	Lebih besar atau sama dengan	1 >= 2 bernilai *false* 2 >= 2 bernilai *true* 3 >= 2 bernilai *true*

<=	Lebih kecil atau sama dengan	1 <= 2 bernilai *true* 2 <= 2 bernilai *true* 3 <= 2 bernilai *false*
?	Operator kondisional atau *ternary* (memberikan nilai pada suatu variabel berdasarkan suatu kondisi)	var keterangan = (angka < 55) ? "Tidak Lulus":"Lulus"; angka = 50; Keterangan bernilai "Tidak Lulus"

5.4.5 Operator *Bitwise*

JavaScript menyimpan suatu angka dalam format 64 bit *floating point*, namun operasi *bitwise* dilakukan dalam format 32 bit angka biner. Dengan demikian, angka yang digunakan dalam operasi *bitwise* akan terlebih dahulu dikonversi menjadi format 32 bits, dan kemudian hasil operasi akan dikonversikan kembali ke dalam format angka JavaScript. Daftar operator *bitwise* dapat dilihat dalam Tabel 5.6.

Tabel 5.6. Daftar Operator *Bitwise* dalam JavaScript

Operator	Deskripsi	Contoh	Ekivalensi[6]	Hasil	Desimal
&	AND	2 & 1	0010 & 0001	0000	0
|	OR	2 | 1	0010 & 0001	0011	3
~	NOT	~ 2	~0010	1101	13
^	XOR	2 ^ 1	0010 ^ 0001	0011	3
<<	Geser kiri dan isi bit paling kanan	2 << 1	0010 << 1	0100	4

[6] Ekivalensi ditunjukkan dalam format 4 bit angka biner untuk kemudahan ilustrasi

	dengan sejumlah bit 0				
>>	Geser kanan dan isi bit paling kiri dengan sejumlah bit yang sama dengan sebelumnya	-2 >> 1	1110 >> 1	1111	-1
>>>	Geser kanan dan isi bit paling kiri dengan sejumlah bit 0	2 >>> 1	0010 >>> 1	0001	1

5.4.6 Operator *String*

Operator *string* dalam JavaScript adalah operator "+" yang digunakan untuk menggabungkan atau konkatenasi *string*. Daftar operator *string* dan contoh penggunaannya dapat dilihat dalam Tabel 5.7.

Tabel 5.7. Daftar Operator *String* dalam JavaScript

Operator	Deskripsi	Contoh
+	Konkatenasi atau menggabungkan string	firstName = "Almira"; lastName = "Wijaya"; name = firstName + " " + lastName; name bernilai "Almira Wijaya"
+=	Konkatenasi atau menggabungkan string	txt1="Hello "; txt1 += "World" txt1 bernilai "Hello World"

5.5 Struktur Kondisi

Struktur kondisi dalam JavaScript meliputi struktur kondisi **IF** dan **SWITCH**.

5.5.1 Kondisi IF

Struktur kondisi **IF** digunakan untuk memastikan bahwa suatu perintah akan dieksekusi ketika kondisi bernilai *true*. Tambahan struktur **ELSE** diperlukan jika suatu eksekusi akan dilakukan ketika kondisi bernilai *false*. Sedangkan struktur **ELSE IF** digunakan untuk menambahkan kondisi baru ketika kondisi bernilai *false*.

Skrip JavaScript:

```
1.  var warna = "red";
2.  if (warna == "green") {
3.      warna = "hijau";
4.  } else if (warna == "blue") {
5.      warna = "biru";
6.  } else {
7.      warna = "merah";
8.  }
9.  warna;
```

Gambar 5.5. Contoh kondisi IF

Berdasarkan penyeleksian kondisi dalam Gambar 5.5, maka nilai akhir dari variabel **warna** adalah "merah".

5.5.2 Kondisi SWITCH

Struktur kondisi **SWITCH** digunakan untuk memberikan beberapa alternatif kondisi untuk eksekusi.

Skrip JavaScript:

```
1.  var warna = "red";
2.  switch (warna){
3.      case "green":
4.          warna = "hijau";
5.          break;
6.      case "blue":
7.          warna = "biru";
8.          break;
9.      case "red":
10.         warna = "merah";
11.         break;
12.     default:
13.         warna="";
14. }
15. warna;
```

Gambar 5.6. Contoh kondisi SWITCH

Perhatikan bahwa alternatif kondisi dalam Gambar 5.6 memberikan hasil yang sama dengan dengan contoh yang diberikan pada struktur kondisi **IF** (Gambar 5.5), yaitu nilai akhir dari variabel **warna** adalah "merah". Hal ini menunjukkan bahwa suatu permasalahan dengan struktur **IF** selalu dapat digunakan untuk menyelesaikan permasalahan dengan struktur **SWITCH**. Namun, ini tidak berarti bahwa struktur **SWITCH** akan selalu dapat digunakan untuk menyelesaikan permasalahan dengan struktur **IF**.

5.6 Struktur Perulangan

Struktur perulangan dalam JavaScript meliputi struktur perulangan **FOR** dan **WHILE**.

5.6.1 Perulangan FOR

Struktur perulangan **FOR** digunakan untuk melakukan perulangan yang jumlahnya sudah diketahui sebelumnya.

Skrip JavaScript:

```
1.  for (var count = 1; count <= 10; count++)
2.  {
3.      console.log("Iteration number " + count);
4.  }
```

Gambar 5.7. Contoh perulangan *FOR*

Berdasarkan perulangan dalam Gambar 5.7, maka pada *console* akan tercetak teks dengen penomoran terurut ke atas mulai dari "Iteration number 1" sampai dengan "Iteration number 10".

5.6.2 Perulangan WHILE

Struktur perulangan **WHILE** digunakan ketika ingin melakukan perulangan selama kondisi yang ditentukan bernilai *true*. Struktur ini memiliki dua bentuk, yaitu **WHILE** dan **DO-WHILE**. Bentuk yang pertama memastikan bahwa perulangan baru akan dilakukan ketika kondisi yang ditentukan bernilai *true*, sedangkan bentuk yang kedua akan selalu setidaknya melakukan satu kali perulangan, meskipun kondisi yang ditentukan bernilai *false*.

Skrip JavaScript:

```
1.  var count = 1;
2.  while (count <= 10)
3.  {
4.      console .log("Iteration number " + count);
5.      count++;
6.  }
```

Gambar 5.8. Contoh perulangan WHILE

Skrip JavaScript:

```
1.  var count = 1;
2.  do
3.  {
4.      console .log("Iteration number " + count);
5.      count++;
6.  }
7.  while (count <= 10)
```

Gambar 5.9. Contoh perulangan DO-WHILE

Perhatikan bahwa perulangan dalam Gambar 5.8 dan Gambar 5.9 memberikan hasil yang sama dengan contoh yang diberikan pada perulangan **FOR** (Gambar 5.7), yaitu pada *console* akan tercetak teks dengan penomoran terurut ke atas mulai dari "Iteration number 1" sampai dengan "Iteration number 10". Dengan kata lain, dapat disimpulkan bahwa permasalahan yang dapat diselesaikan dengan perulangan **FOR** akan selalu dapat

diselesaikan dengan menggunakan perulangan **WHILE**. Namun tidak demikian halnya dengan kebalikannya, yaitu bahwa perulangan **FOR** mungkin tidak dapat digunakan untuk menyelesaikan permasalahan yang pada perulangan **WHILE**.

5.6.3 Perulangan Tidak Terhingga (*Infinite*)

Perulangan tidak terhingga adalah perulangan yang tidak memiliki kondisi kapan ia akan berhenti. Keadaan semacam ini akan sangat mempengaruhi kinerja komputer dan bahkan dapat membuatnya menjadi *crash*. Permasalahan perulangan tidak terhingga dapat diselesaikan dengan penggunaan perintah **break**. Gambar 5.10 memperlihatkan contoh penggunaan perintah **break** dalam JavaScript. Meskipun skrip contoh tidak memiliki kondisi kapan ia akan berhenti, namun penggunaan perintah **break** akan menyebabkan program berhenti ketika **"if (count == 10)"**. Hasil akhir yang diberikan adalah sama dengan hasil eksekusi skrip dalam Gambar 5.7, Gambar 5.8, dan Gambar 5.9.

Skrip JavaScript:

```
1.  for (var count = 1; /* no condition here */ ; count++)  {
2.      console.log("Iteration number " + count);
3.      if (count == 10) {
4.          break;
5.      }
6.  }
```

Gambar 5.10. Contoh perintah *break*

5.7 Fungsi (*Function*)

Fungsi adalah sub-program yang dibuat untuk melaksanakan suatu pekerjaan tertentu, seperti menghitung hasil penjumlahan, perkalian, luas suatu area, dan lain-lain. Fungsi di dalam Javascript meupakan sebuah *objek*, karena ia memiliki *method* (metode) dan *properties*.

5.7.1 Deklarasi Fungsi

Deklarasi fungsi di dalam JavaScript memerlukan tiga komponen berikut:

- Nama fungsi. Suatu fungsi harus diikuti oleh tanda kurung ("**()**") dan aturan penamaan fungsi sama dengan aturan penamaan variabel
- Nilai atau argumen yang diperlukan (bila ada). Daftar argumen direpresentasikan melalui parameter yang diletakkan di dalam tanda kurung: (parameter1, parameter2, ...)
- Operasi yang akan dilakukan di dalam badan fungsi. Operasi yang akan dieksekusi oleh fungsi diletakkan di dalam tanda kurung kurawal ("**{}**"). Fungsi yang menghasilkan suatu nilai keluaran harus menggunakan perintah **return**.

Gambar 5.11 memperlihatkan sebuah contoh pembuatan fungsi dalam JavaScript (baris 1 – 5). Fungsi yang diberi nama **addNumber** dibuat untuk menghitung penjumlahan dua buah argumen yang diisikan ke dalam fungsi melalui parameter **value1** dan **value2**. Hasil penjumlahan disimpan dalam variabel **total** dan menjadi keluaran dari fungsi **addNumber**.

Skrip JavaScript:

```
1.  function addNumber(value1, value2)
2.  {
3.      total = value1 + value2;
4.      return total;
5.  }
6.
7.  var sum = addNumber(1,2);
```

Gambar 5.11. Contoh pembuatan dan penggunaan fungsi

5.7.2 Memanggil Fungsi

Fungsi dapat dipanggil dengan cara menyebutkan nama fungsi dan tanda kurun, beserta daftar parameter jika ada. Untuk contoh dalam Gambar 5.11, maka pemanggilan fungsi

addNumber dapat dilakukan dengan cara seperti yang ditunjukkan pada baris 7. Sebagai hasilnya, isi variabel **sum** menjadi bernilai 3.

5.8 *Built-in Objects*

JavaScript memiliki empat *built-in objects* yang meliputi: *String*, *Array*, *Math* dan *Date*.

5.8.1 *String Object*

Beberapa *method* dalam *String Object* adalah: **length**, **charAt()**, **concat()**, **indexOf()**, **replace()**, **substr()**, **split()**, dan lain-lain. Gambar 5.12 memperlihatkan contoh implementasi *method* dalam *String Object*. Sebagai hasilnya, nilai dari variabel **aIndex**, **arrName**, dan **arrDomain** berturut-turut adalah 8, ["noor", "ifada"], dan ["trunojoyo", "ac", "id"].

Skrip JavaScript:

```
1.  // Membuat dan mengisi variabel "str" yang akan digunakan sebagai String Object
2.  str = "this is a string";
3.  // Mengimplementasikan method "indexof"  untuk mencari index dari karakter "a" pada object "str"
4.  aIndex = str.indexOf('a');
5.
6.  // Membuat dan mengisi  variabel "strEmail" yang akan digunakan sebagai String Object
7.  var strEMail = "noor.ifada@trunojoyo.ac.id";
8.  // Mengimplementasikan method "split" untuk memecah object "strEmail" menjadi object string "arrEm
    ail" yang berbentuk array dengan menggunakan  karakter "@" sebagai separator
9.  var arrEMail = strEMail.split("@");
10. // Mengimplementasikan method "split" untuk memecah object "arrEmail[0]" menjadi object string "ar
    rName" yang berbentuk array dengan menggunakan  karakter "." sebagai separator
11. var arrName = arrEMail[0].split(".");
12. // Mengimplementasikan method "split" untuk memecah object "arrEmail[1]" menjadi object string "ar
    rDomain" yang berbentuk array dengan menggunakan  karakter "." sebagai separator
13. var arrDomain = arrEMail[1].split(".");
```

Gambar 5.12. Contoh implementasi *method* dalam *String Object*

5.8.2 *Array Object*

Beberapa *method* dalam *Array Object* adalah: **length**, **sort()**, **reverse()**, **shift()**, **pop()**, dan lain-lain. Gambar 5.13 memperlihatkan contoh implementasi *method* dalam *Array Object*. Sebagai hasilnya, nilai **myArray** adalah [7, 8, 25, 41].

Skrip JavaScript:

```
1.  // Membuat dan mengisi variabel "myArray" yang akan digunakan sebagai Array Object
2.  var myArray = [25, 8, 7, 41];
3.  // Mengimplementasikan method "sort" untuk mengurutkan angka pada object "myArray" secara dari yan
    g terkecil ke yang paling besar
4.  myArray.sort(function(a,b){return a - b});
```

Gambar 5.13. Contoh implementasi *method* dalam *Array Object*

5.8.3 *Math Object*

Beberapa *method* dalam *Math Object* adalah: **E**, **PI**, **abs()**, **cos()**, **pow()**, **log()**, **floor()**, **sqrt()**, dan lain-lain. Gambar 5.14 memperlihatkan contoh implementasi *method* dalam *Math Object*. Sebagai hasilnya, nilai dari variabel **randomInteger** akan selalu berubah-ubah karena nilai **randomNumber** adalah selalu acak.

Skrip JavaScript:

```
1.  // Mengimplementasikan method "random" untuk menghasilkan bilangan acak antara 0 -
     1 dan disimpan sebagai "randomNumber"
2.  var randomNumber = Math.random();
3.  // Mengimplementasikan method "floor" untuk menghasilkan  pembulatan ke bawah dari hasil perkalian
     "randomNumber" dengan angka 100. Hasilnya disimpan sebagai "randomInteger"
4.  var randomInteger = Math.floor(randomNumber*100);
```

Gambar 5.14. Contoh implementasi *method* dalam *Math Object*

5.8.4 *Date Object*

Beberapa *method* dalam Date Object: **getFullYear()**, **getTime()**, **getDay()**, dan lain-lain. Gambar 5.15 memperlihatkan contoh implementasi *method* dalam *Date Object*. Sebagai hasilnya, nilai dari variabel **strTodayIs** akan selalu berubah-ubah berdasarkan hari ketika skrip dieksekusi.

Skrip JavaScript:

```
1. // Membuat dan mengisi variabel "myDate" dengan Date object yang dihasilkan dengan konstruktor men
   ggunakan new Date()
2. var myDate = new Date();
3. // Membuat dan mengisi variabel "arrDaynames" dengan data array
4. var arrDayNames = ["Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"];
5. // Mengimplementasikan method "getDay" pada object "myDate" untuk mengetahui konversi angka (antar
   a 0-
   6) dari hari ini dalam suatu minggu. Konversi angka yang didapatkan digunakan sebagai indeks untuk
    memanggil nilai tertentu dalam variabel arrDayNames. Hasil akhir disimpan dalam variabel strToday
   Is
6. var strTodayIs = arrDayNames[myDate.getDay()];
```

Gambar 5.15. Contoh implementasi *method* dalam *Date Object*

5.9 *RegExp*

RegExp (*Regular Expression*) adalah sebuah *object* yang digunakan untuk mendeskripsikan pola dari suatu (kumpulan) karakter dan mengecek kesesuaian suatu teks (atau *string*) dengan pola tersebut. *RegExp* terdiri dari dua komponen yaitu *pattern* dan *modifiers*, dengan sintaks penulisan: "**/pattern/modifiers;**". Daftar *pattern* dan *modifiers* yang ada dalam JavaScript diperlihatkan dalam Tabel 5.8 dan Tabel 5.9.

Tabel 5.8. Daftar *Patterns* dalam *RegExp*[7]

Simbol	Deskripsi
Brackets (untuk mencari batasan karakter)	
[abc]	Mencari karakter yang ada di dalam kurung siku
[^abc]	Mencari karakter yang tidak ada di dalam kurung siku
[0-9]	Mencari karakter (angka) yang ada di dalam kurung siku
[^0-9]	Mencari karakter (non angka) yang tidak ada ada di dalam kurung siku
(red | blue | green)	Mencari berdasarkan pilihan yang tersedia

[7] Sumber: https://www.w3schools.com/jsref/jsref_obj_regexp.asp

Metacharacters (kumpulan karakter yang memiliki makna tertentu)	
.	Mencari karakter tunggal, kecuali *new line* atau *line terminator*
\w	Mencari karakter kata
\W	Mencari karakter bukan kata
\d	Mencari sebuah digit angka
\D	Mencari sebuah karakter non angka
\s	Mencari sebuah karakter *whitespace*
\S	Mencari sebuah karakter bukan *whitespace*
\b	Pencarian kesesuaian di awal/akhir kata
\B	Pencarian kesesuaian tidak di awal/akhir kata
\0	Mencari karakter NUL
\n	Mencari karakter *new line*
\f	Mencari karakter *form feed*
\r	Mencari karakter *carriage return*
\t	Mencari karakter tabulasi
\v	Mencari karater tabulasi vertikal
\xxx	Mencari karakter dengan format angka oktal *xxx*
\xdd	Mencari karakter dengan format angka heksadesimal *dd*
\uxxxx	Mencari karakter *Unicode* dengan format angka heksadesimal *xxxx*
Quantifiers (menunjukkan kuantitas)	
n+	Sesuai dengan *string* manapun yang memiliki setidaknya satu *n*
n*	Sesuai dengan *string* manapun yang memiliki kemunculan *n* sebanyak nol atau lebih
n?	Sesuai dengan *string* manapun yang memiliki kemunculan *n* sebanyak nol atau satu kali
n{X}	Sesuai dengan *string* manapun yang memiliki urutan *n* sebanyak *X*

n{X,Y}	Sesuai dengan *string* manapun yang memiliki urutan *n* sebanyak *X* hingga *Y*
n{X,}	Sesuai dengan *string* manapun yang memiliki urutan *n* setidaknya sebanyak *X*
n$	Sesuai dengan *string* manapun yang diakhiri dengan *n*
^n	Sesuai dengan *string* manapun yang awalannya adalah *n*
?=n	Sesuai dengan *string* manapun yang diikuti oleh string *n*
?!n	Sesuai dengan *string* manapun yang tidak diikuti oleh string *n*

Tabel 5.9. Daftar *Modifier* dalam *RegExp*

Simbol	Deskripsi
i	Mengecek kesesuaian yang *case-insensitive*
g	Mengecek kesesuaian secara keseluruhan
m	Mengecek kesesuaian *multiline*

Sebagai contoh, mari kita membuat *RegExp* untuk validasi tanggal dengan format "yyyy-mm-dd" atau "yyyy/mm/dd", dan batasan tahun yang diinginkan adalah antara 1800 – 2099. Pembuatan *pattern* dari *RegExp* dengan ketentuan tersebut dapat dilakukan dengan membaginya menjadi tujuh komponen berikut:

- Komponen awalan *pattern*. *Pattern*: **^**
- Komponen tahun yang berupa 4 digit karakter angka, dengan ketentuan:
  - ✓ 2 digit pertama, yaitu berupa pilihan karakter angka "18", "19", atau "20". *Pattern*: **(18|19|20)**
  - ✓ 1 digit ketiga, yaitu berupa angka dengan batasan 0 sampai 9. *Pattern*: **\d**
  - ✓ 1 digit keempat, yaitu berupa angka dengan batasan 0 sampai 9. . *Pattern*: **\d**

Pattern komponen tahun: **(18|19|20) \d\d**

- Komponen tanda pemisah antara tahun dan bulan. *Pattern*: **[-/]**
- Komponen bulan yang berupa 2 digit karakter angka. *Pattern* komponen bulan dapat dibuat dengan memberikan dua pilihan kombinasi sebagai berikut:
  - ✓ Untuk bulan antara 01 – 09: Digit pertama adalah karakter angka "0", sedangkan digit kedua adalah angka antara 1 sampai 9. *Pattern*: **0[1-9]**
  - ✓ Untuk bulan antara 10 – 12: Digit pertama adalah karakter angka "1", sedangkan digit kedua dengan pilihan karakter "0", "1", dan "2". *Pattern*: **1[012]**

    Pattern komponen bulan: **(0[1-9]|1[012])**
- Komponen tanda pemisah antara bulan dan tanggal. *Pattern*: **[-/]**
- Komponen tanggal yang berupa 2 digit karakter angka. *Pattern* komponen tanggal dapat dibuat dengan memberikan tiga pilihan kombinasi sebagai berikut:
  - ✓ Untuk tanggal antara 01 – 09: Digit pertama adalah karakter angka "0", sedangkan digit kedua adalah angka antara 1 sampai 9. *Pattern*: **0[1-9]**
  - ✓ Untuk bulan antara 10 – 29: Digit pertama dengan pilihan karakter angka "1" dan "2", sedangkan digit kedua adalah angka antara 0 sampai 9. *Pattern*: **[12][0-9]**
  - ✓ Untuk tanggal antara 30 – 31: Digit pertama adalah karakter angka "3", sedangkan digit kedua dengan pilihan karakter "0" dan "1". *Pattern*: **3[01]**

    Pattern komponen tanggal: **(0[1-9]|[12][0-9]|3[01])**
- Komponen akhiran *pattern*. *Pattern*: **$**

Hasil akhir *pattern* adalah "**^(18|19|20)\d\d[-/](0[1-9]|1[012])[-/](0[1-9]|[12][0-9]|3[01])$**". Dengan menggunakan modifier "**g**", maka *RegExp* dapat digunakan untuk membuat fungsi **verifyDate** seperti diperlihatkan dalam Gambar 5.16.

Skrip JavaScript:

```
1.  function verifyDate(strDate) {
2.      // RegExp untuk validasi tanggal  dengan format “yyyy-mm-
    dd”  atau “yyyy/mm/dd”, dan batasan tahun antara 1800 – 2099
3.      var patt=/^(18|19|20)\d\d[-/](0[1-9]|1[012])[-/](0[1-9]|[12][0-9]|3[01])$/g;
4.      return patt.test(strDate);
5.  }
```

Gambar 5.16. Contoh fungsi dengan RegExp untuk validasi tanggal

Selain digunakan di dalam fungsi, *RegExp* juga dapat diimplementasikan dengan menggunakan *method* sebagai *object*. *RegExp object* memiliki empat *method*, yaitu: **exec()**, **test()**, **compile()**, **toString()**. Gambar 5.17 memperlihatkan contoh implementasi method dalam *RegExp Object*. Sebagai hasilnya, variabel **result** menjadi bernilai *true*.

Skrip JavaScript:

```
1.  var str="Dasar Pemrograman Web";
2.  // RegExp
3.  var patt=/Web/g;
4.  // Mencari kata "Web" di dalam variabel "str". Jika kata "Web" ada maka hasilnya adalah benar/true
    , dan salah/false jika sebaliknya
5.  var result = patt.test(str);
```

Gambar 5.17. Contoh implementasi *method* dalam RegExp *Object*

5.10 HTML DOM

HTML DOM (*Document Object Model*) merupakan model representasi suatu halaman *web* dari sudut pandang *web browser*. Elemen-elemen dari HTML DOM membentuk struktur *tree* dari sekumpulan *object*. JavaScript dapat memanfaatkan HTML DOM untuk:

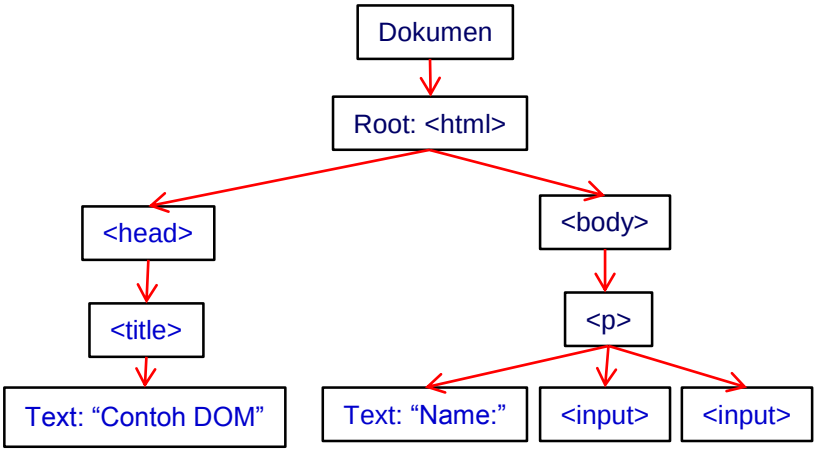
- Membaca konten halaman *web* (termasuk nilai yang saat ini diisikan melalui elemen *form* HTML)
- Mengubah konten halaman *web* (termasuk nilai yang saat ini ditampilkan dengan menggunakan elemen HTML dan juga secara dinamis menambahkan atau menghapus elemen HTML dari halaman *web*).

Gambar 5.18 memperlihatkan contoh skrip dokumen HTML yang struktur DOM *tree*-nya direpresentasikan dalam Gambar 5.19.

Skrip HTML:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title> Contoh DOM </title>
5.      </head>
6.      <body>
7.          <p>Name: <input type="input" name="surname"/>
8.              <input type="button" name="Press" value="Press"/>
9.          </p>
10.     </body>
11. </html>
```

Gambar 5.18. Skrip HTML untuk contoh pembuatan DOM tree



Gambar 5.19. Struktur DOM *tree* dari dokumen HTML pada Gambar 5.18

5.11 Mengakses Elemen HTML

Mengakses nilai elemen HTML dapat dilakukan dengan dua cara, yaitu:

- menggunakan *node* dari DOM *tree*
- menggunakan *method* **getElementById()**

Perhatikan Gambar 5.20 untuk melihat perbandingan bagaimana nilai elemen HTML diakses dengan menggunakan node dari DOM *tree* ataupun *method* **getElementById()**.

Skrip HTML dan JavaScript:

```
1.  <!-- Form yang akan diakses nilai elemen HTML-nya -->
2.  <form name="myForm">
3.      <input type="text" name="surname"/>
4.  </form>
5.
6.
7.  <script>
8.      <!-- Mengakses elemen HTML dengan menggunakan node pada DOM tree-->
9.      document.myForm.surname.value = "";
10.     <!-- Mengakses elemen HTML dengan menggunakan method getElementById -->
11.     document.getElementById("EmployeeSurname").value = "";
12. </script>
```

Gambar 5.20. Contoh akses elemen HTML dengan *node* pada DOM *tree* atau *method getElementById*

5.12 Kotak Dialog

JavaScript memiliki tiga macam kotak dialog, yaitu:

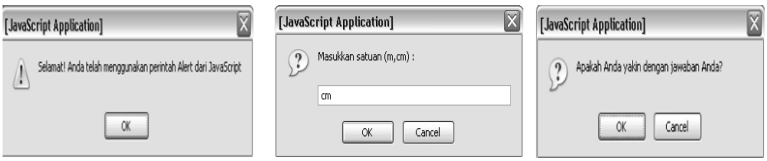
- Kotak dialog **Alert**, digunakan untuk memperingatkan *user* mengenai sesuai hal, atau dalam kasus tertentu untuk memberikan instruksi kepada *user*. Sintaks penulisan: **window.alert(‘**text**’)**
- Kotak dialog **Prompt**, digunakan untuk menampilkan sebuah kotak isian yang dapat menerima sesuai informasi dari *user*. Sintaks penulisan: **window.prompt(‘**text**’,’**defaultvalue**’)**
- Kotak dialog **Confirm**, digunakan untuk memperingatkan *user* untuk mengkonfirmasikan suatu pilihan (OK atau Cancel). Sintaks penulisan: **window.confirm(‘**text**’)**.

Perhatikan Gambar 5.21 yang memperlihatkan contoh pembuatan ketiga kotak dialog dalam JavaScript.

Skrip HTML dan JavaScript:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Kotak Dialog</title>
5.      </head>
6.      <body>
7.          <script>
8.              window.alert('Selamat Anda telah menggunakan perintah Alert dari JavaScript');
9.              window.prompt('Masukkan satuan (m,cm)','cm');
10.             window.confirm('Apakah Anda yakin dengan jawaban Anda?');
11.         </script>
12.     </body>
13. </html>
```

Hasil Eksekusi:



Gambar 5.21. Contoh pembuatan kotak dialog *alert*, *prompt* dan *confirm*

5.13 HTML *Event*

HTML *Event* adalah suatu kondisi yang terjadi pada elemen HTML yang dapat menjadi *trigger* eksekusi *JavaScript*. HTML *event* dapat berkaitan dengan sesuatu yang dilakukan oleh *web browser* (contoh: *load* atau *unload* halaman *web*), ataupun *user* (contoh: aktivitas *user* ketika menggunakan *keyboard* ataupun *mouse* pada elemen HTML). Tabel 5.10 memperlihatkan daftar HTML *event* yang dapat menjadi *trigger* bagi eksekusi JavaScript.

Tabel 5.10. Daftar HTML *Event* (Lemay, Colburn dan Kyrnin 2015)

Event	Dapat diimpementasikan pada
onload	**<body>, <frameset>**
onunload	**<body>, <frameset>**
onclick	Sebagian besar elemen
ondblclick	Sebagian besar elemen
onmousedown	Sebagian besar elemen
onmouseup	Sebagian besar elemen

onmouseover	Sebagian besar elemen
onmousemove	Sebagian besar elemen
onmouseout	Sebagian besar elemen
onfocus	**<a>**, **<area>**, **<label>**, **<input>**, **<select>**, **<textarea>**, **<button>**
onblur	**<a>**, **<area>**, **<label>**, **<input>**, **<select>**, **<textarea>**, **<button>**
onkeypress	Sebagian besar elemen
onkeydown	Sebagian besar elemen
onkeyup	Sebagian besar elemen
onsubmit	**<form>**
onreset	**<form>**
onselect	**<input>**, **<textarea>**
onchange	**<input>**, **<select>**, **<textarea>**

5.14 Validasi *Form*

JavaScript umum digunakan untuk melakukan validasi data isian suatu *form* sebelum dikirimkan ke *server*. Bentuk validasi yang dilakukan misalnya adalah ketika data pada *form* akan dikirimkan ke *server* (yaitu dengan menekan tombol **Submit**), Dalam hal ini, JavaScript dapat diaktivasikan dengan mengimplementasikan HTML *event* **onsubmit** untuk mengecek apakah:

- Data pada elemen *form* sudah diisikan, menampilkan pesan kesalahan berbentuk kotak dialog *alert* apabila kotak masukan belum diisi, dan selanjutnya mengarahkan fokus kursor pada elemen *form* (Gambar 5.22)
- Data yang diisikan pada elemen *form* adalah angka bilangan bulat, menampilkan pesan kesalahan berbentuk kotak dialog *alert* apabila kotak masukan belum diisi dengan angka bilangan bulat, dan selanjutnya mengarahkan fokus kursor pada elemen *form* (Gambar 5.23).

Perhatikan bahwa jika kotak masukan dari *form* dalam Gambar 5.22 dan Gambar 5.23 telah diisi dengan benar dan kemudian tombol **Submit** ditekan, maka *user* akan diarahkan ke URL *file* **successful.html** sesuai dengan nilai atribut **action**.

Skrip HTML dan JavaScript:

```
1.  <script>
2.      function checkFill() {
3.          if (document.myForm.surname.value == "" ) {
4.              alert ("Please fill in the 'Your Name' box.");
5.              document.myForm.surname.focus();
6.              return false;
7.          }
8.          return true;
9.      }
10. </script>
11.
12. <form name="myForm" action="successful.html" method="post" onsubmit="return checkFill();">
13.     Your Name: <input type="text" name="surname">
14.     <input type="submit">
15. </form>
```

Hasil Eksekusi:

Gambar 5.22. Contoh pengecekan isian dan tampilan pesan kesalahan untuk validasi *form*

Skrip HTML dan JavaScript:

```
1.  <script>
2.      function isIntegerNumber() {
3.          var i, allowed = "0123456789";
4.          var numval=document.myForm.age.value;
5.          for (i=0; i < numval.length; i++) {
6.              // if character is not a digit return false
7.              if (allowed.indexOf(numval.charAt(i)) == -1) {
8.                  window.alert ("Please fill in 'Your Age' with numeric values");
9.                  document.myForm.age.focus();
10.                 return false;
11.             }
12.         }
13.         return true;
14.     }
15. </script>
16.
17. <form name="myForm" action="successful.html" method="post" onsubmit="return isInteger-
    Number();">
18.     Your Age: <input type="text" name="age">
19.     <input type="submit">
20. </form>
```

Hasil Eksekusi:

Gambar 5.23. Contoh pengecekan format angka bilangan bulat dan tampilan pesan kesalahan untuk validasi *form*

5.15 Latihan Soal

Latihan 5.1

Gunakan perulangan **FOR** pada JavaScript untuk mencetak teks dengan penomoran terurut ke bawah pada *console*, yaitu dari "Iteration number 10" hingga "Iteration number 1"!

Latihan 5.2

Gunakan perulangan **WHILE** pada JavaScript untuk menyelesaikan permasalahan pada Latihan 5.1!

Latihan 5.3

Buatlah fungsi dengan *RegExp* pada JavaScript untuk menggantikan fungsi **isIntegerNumber** yang digunakan untuk melakukan validasi format angka bilangan bulat pada Gambar 5.23!

Latihan 5.4

Tambahkan fungsi untuk validasi *form* pada Gambar 5.23 dengan menggunakan JavaScript agar juga dapat menampilkan pesan kesalahan berbentuk kotak dialog **alert** apabila kotak masukan belum diisi!

INDEKS

A

Argumen, 83
Atribut, ix, 11, 15, 18, 19, 22, 24, 25, 27, 28, 29, 30, 31, 32, 36, 37, 48, 61, 67, 72, 95, 111
Atribut, xiii, 11, 29, 30, 31, 32, 33

B

Built-in Object, viii, 84

C

Client-side, 4, 5, 20, 31, 102
CSS, i, ii, iii, vi, vii, x, xi, 5, 6, 7, 21, 26, 41, 42, 43, 44, 45, 46, 47, 48, 49, 51, 52, 53, 54, 56, 57, 64, 65, 66, 100, 102, 107, 108, 110

D

Dinamis, 5, 90
Div, x, 45, 51, 52, 54, 110
DOM, viii, xii, 69, 90, 91, 92

E

elemen, vi, vii, ix, x, xii, xiii, 7, 11, 12, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 35, 36, 37, 41, 43, 44, 45, 47, 48, 49, 51, 52, 53, 61, 62, 63, 67, 69, 70, 71, 72, 90, 91, 92, 93, 94, 111

F

For, viii, xi, 80, 81, 82, 96, 113
Form, ix, x, xii, 9, 29, 30, 31, 32, 34, 35, 36, 37, 39, 57, 69, 87, 90, 94, 95, 96, 106, 108, 114
Fungsi, xi, xii, 1, 69, 83, 89, 90, 96, 113, 114

H

Heading, ix, 13, 14, 42
HTML, i, ii, iii, v, vi, vii, viii, ix, x, xii, xiii, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 29, 30, 37, 38, 39, 41, 42, 43, 44, 47, 48, 49, 50, 51, 52, 53, 59, 60, 63, 64, 65, 66, 67, 69, 70, 71, 72, 90, 91, 92, 93,94, 100, 102, 103, 104, 105, 106, 107, 108
HTTP, v, 1, 3, 30

I

IF, viii, xi, 79, 80
image, 9, 17, 18
Image, vi, 17, 18
internet, ix, 1, 2, 3
Internet, v, 1, 2, 7, 117

J

JavaScript, iii, vii, xi, xiii, 5, 6, 7, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 82,

83, 84, 86, 90, 92, 93, 94, 96, 100, 102, 113, 114

L

Link, ix, xi, xiii, 9, 17, 19, 38, 48, 65, 66, 72

O

Operator, 69, 74, 75, 76, 77, 78

P

Parameter, 83
Protokol, 3, 5
Protokol, 3

S

Selector, x, 41, 42, 43, 44, 45, 46, 47, 110
Server-side, 5, 20
Statis, iii, 4, 6, 8, 69, 102
SWITCH, viii, xi, 79, 80

T

Table, vi, 21, 22, 27, 28, 51, 52
tag, ix, xi, 11, 60, 61, 62, 72, 111

V

Variabel, xiii, 69, 73, 74, 75, 76, 77, 79, 80, 83, 84, 85, 90

W

Web, iii, ix, x, xi, xiii, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 17, 18, 19, 20, 21, 22, 28, 29, 38, 41, 48, 51, 52, 53, 54, 55, 57, 59, 60, 63, 64, 68, 69, 70, 71, 90, 93, 102, 103, 110, 116, 117
While, viii, xi, 80, 81, 82, 96, 113
World Wide, 1, 9

X

XML, i, ii, iii, vii, x, xi, 6, 59, 60, 61, 62, 63, 64, 65, 66, 68, 100, 111, 112
XSL, xi, 66, 67, 68, 112
XSLT, vii, 64, 66, 67

DAFTAR PUSTAKA

Duckett, Jon. 2010. *Beginning HTML, XHTML, CSS, and JavaScript.* Indiana: Wiley Publishing.

Fawcet, Joe, Liam R. E. Quin, dan Danny Ayers. 2012. *Beginning XML (5th Edition).* Indiana: John Wiley & Sons.

Krause, Jörg. 2016. *Introducing Web Development.* Berlin: Apress.

Lemay, Laura, Rafe Colburn, dan Jennifer Kyrnin. 2015. *HTML, CSS & JavaScript Web Publishing in One Hour a Day, Sams Teach Yourself: Covering HTML5, CSS3, and jQuery (7th Edition).* Indiana: Sams Publishing.

LAMPIRAN

Jawaban Soal Latihan

Latihan 1.1

Persamaan antara halaman *web* (*webpage*) dan halaman utama (*homepage*) adalah bahwa keduanya adalah merupakan halaman *web*, yang biasanya berbentuk dokumen HTML. Perbedaannya adalah bahwa, dalam satu situs *web*, *homepage* hanya dapat berjumlah satu saja sedangkan *webpage* dapat berjumlah lebih.

Latihan 1.2

URL (*Uniform Resource Locator*) adalah penentu lokasi setiap dokumen atau informasi yang ada di dalam situs *web* pada *server*.

Latihan 1.3

Tiga Bahasa standar yang digunakan dalam *web* statis:

1. HTML, yaitu bahasa *markah* yang digunakan untuk mendeskripsikan konten atau struktur suatu halaman *web*
2. CSS, yaitu bahasa *stylesheet* yang untuk memformat konten atau membuat *layout* halaman *web*
3. JavaScript yaitu bahasa pemrograman yang digunakan untuk mengatur perilaku halaman *web* pada *web browser* pada *web browser* atau *client-side*

Latihan 2.1

Skrip HTML untuk menampilkan *list* bersarang (*nested list*):

Skrip HTML:

```
1.  <ol>
2.      <li>Daftar 1
3.          <ul>
4.              <li>Sub-daftar 1.1</li>
5.              <li>Sub-daftar 1.2</li>
6.          </ul>
7.      </li>
8.      <li>Daftar 2
9.          <ul>
10.             <li>Sub-daftar 2.1</li>
11.             <li>Sub-daftar 2.2</li>
12.         </ul>
13.     </li>
14.     <li>Daftar 3</li>
15. </ol>
```

Latihan 2.2

Skrip HTML untuk membuat halaman *web* yang memuat gambar foto dan *list*:

Skrip HTML:

```
1.  <img src="NoorIfada.jpg">
2.  <p><b>Data Staf:</b></p>
3.  <ul>
4.      <li>Nama: Noor Ifada</li>
5.      <li>Email:
6.          <a href="mailto:noor.ifada@trunojoyo.ac.id">noor.ifada@trunojoyo.ac.id</a>
7.      </li>
8.      <li>Institusi:
9.          <a href="http://www.trunojoyo.ac.id/">Universitas Trunojoyo Madura</a>
10.     </li>
11. </ul>
```

Latihan 2.3

Skrip HTML untuk membuat tabel:

Skrip HTML:

```
1.  <table border="1">
2.      <caption><b>TABEL WARNA</b></caption>
3.      <tr>
4.          <th></th>
5.          <th>Merah</th>
6.          <th>Kuning</th>
7.          <th>Biru</th>
8.      </tr>
9.      <tr>
10.         <th>Merah</th>
11.         <td>Merah</td>
12.         <td>Orange</td>
13.         <td>Ungu</td>
14.     </tr>
15.     <tr>
16.         <th>Kuning</th>
17.         <td>Orange</td>
18.         <td>Kuning</td>
19.         <td>Hijau</td>
20.     </tr>
21.     <tr>
22.         <th>Biru</th>
23.         <td>Ungu</td>
24.         <td>Hijau</td>
25.         <td>Biru</td>
26.     </tr>
27. </table>
```

Latihan 2.4

Skrip HTML untuk membuat tabel:

Skrip HTML:

```
1.  <table border="1">
2.      <caption>
3.          <b>STATISTIK KELAS</b>
4.      </caption>
5.      <tr>
6.          <th rowspan="2">Mata Kuliah</th>
7.          <th rowspan="2">Kelas</th>
8.          <th colspan="2">Jumlah Mahasiswa</th>
9.      </tr>
10.     <tr>
11.         <th>Laki-laki</th>
12.         <th>Perempuan</th>
13.     </tr>
14.     <tr>
15.         <td rowspan="3">Dasar Pemrograman Web</td>
16.         <td>A</td>
17.         <td>20</td>
18.         <td>22</td>
19.     </tr>
20.     <tr>
21.         <td>B</td>
22.         <td>23</td>
23.         <td>19</td>
24.     </tr>
25.     <tr>
26.         <td>C</td>
27.         <td>21</td>
28.         <td>21</td>
29.     </tr>
30. </table>
```

Latihan 2.5

Skrip HTML untuk memformat tampilan *form* dengan menggunakan tabel:

Skrip HTML:

```
1.  <form action="submit.html" method="post">
2.      <fieldset>
3.          <legend>Data Registrasi</legend>
4.          <table>
5.          <!-- type = "text" -->
6.              <tr>
7.                  <td><label for="username">Username:</label></td>
8.                  <td><input type="text" name="username" id="username" size="20" maxlength="20" r
    equired /> </td>
9.              </tr>
10.         <!-- type = "password" -->
11.             <tr>
12.                 <td><label for="password">Password:</label></td>
13.                 <td><input type="password" name="password" id="password" size="20" required />
    </td>
14.             </tr>
15.         <!-- type = "date" -->
16.             <tr>
17.                 <td><label for="birthdate">Tanggal Lahir:</label></td>
18.                 <td><input type="date" name="birthdate" id="birthdate" required /> </td>
19.             </tr>
20.         <!-- type = "tel" -->
21.             <tr>
22.                 <td><label for="telephone">Telepon:</label></td>
23.                 <td><input name="telephone" id="telephone" placeholder="xxx-
    xxxxxxx" pattern="[0-9]{3}-[0-9}{7}" type="tel" required /> </td>
24.             </tr>
25.         <!-- type = "radio" -->
26.             <tr>
27.                 <td><label for="gender">Jenis Kelamin:</label></td>
28.                 <td><label for="gender">Laki-laki</label>
29.                     <input type="radio" name="gender" id="gender" value="male" />
30.                     <label for="gender">Perempuan</label>
31.                     <input type="radio" name="gender" id="gender" value="female" checked />
32.                 </td>
33.             </tr>
34.         <!-- type = "checkbox" -->
35.             <tr>
36.                 <td colspan="2">
37.                     <label for="binary">Bersedia menerima iklan melalui email?</label>
38.                     <input type="checkbox" name="binary" id="binary" checked />
39.                 </td>
40.             </tr>
41.         </table>
42.     <!-- type = "submit" -->
43.         <input type="submit" value="Submit" name="button1" id="button1">
44.     <!-- type = "reset" -->
45.         <input type="reset" value="Reset" name="button2" id="button2">
46.     </fieldset>
47. </form>
```

Latihan 3.1

Skrip untuk memformatlah tampilan tabel dalam dokumen HTML dengan menggunakan CSS:

Skrip HTML dan CSS:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Mewarnai tabel dengan CSS</title>
5.          <style>
6.              body {
7.                  background-color: purple;
8.                  color: black;
9.                  font-family:arial;
10.                 }
11.             table.center {
12.                 margin-left:auto;
13.                 margin-right:auto;
14.                 }
15.             table {
16.                 background-color: white;
17.                 }
18.             table, tr, th, td{
19.                 border: 1px solid black;
20.                 }
21.         </style>
22.     </head>
23.     <body>
24.         <table class="center">
25.             <tr style="background-color:yellow;">
26.                 <th>No</th><th>NPM</th><th>Nama</th></tr>
27.             <tr>
28.                 <td style="background-color:blue;">1.</td>
29.                 <td style="background-color:red; width:80px; height:40px;">06.100.001</td>
30.                 <td style="background-
    color:green; width:180px; height:40px;">Amin A. Angkasa</td></tr>
31.         </table>
32.     </body>
33. </html>
```

Latihan 3.2

Skrip HTML dan CSS untuk untuk mengatur *layout form*:

Skrip HTML:

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <meta charset="utf-8"/>
5.          <title>Layout Form dengan CSS</title>
6.          <link href="style_form.css" rel="stylesheet" />
7.      </head>
8.      <body>
9.          <form action="submit.html" method="post">
10.             <fieldset>
11.                 <legend>Data Registrasi</legend>
12.                 <div class="field">
13.                 <!-- type = "text" -->
14.                     <label class=layout1 for="username">Username:</label>
15.                     <in-
    put type="text" name="username" id="username" size="20" maxlength="20" required />
16.                 </div>
17.                 <!-- type = "password" -->
18.                 <div class="field">
19.                     <label class=layout1 for="password">Password:</label>
20.                     <input type="password" name="password" id="password" size="20" re-
    quired />
21.                 </div>
22.                 <!-- type = "date" -->
23.                 <div class="field">
24.                     <label class=layout1 for="birthdate">Tanggal Lahir:</label>
25.                     <input type="date" name="birthdate" id="birthdate" required />
26.                 </div>
27.                 <!-- type = "tel" -->
28.                 <div class="field">
29.                     <label class=layout1 for="telephone">Telepon:</label>
30.                     <input name="telephone" id="telephone" placeholder="xxx-xxxxxxx" pat-
    tern="[0-9]{3}-[0-9}{7}" type="tel" required />
31.                 </div>
32.                 <!-- type = "radio" -->
33.                 <div class="field">
34.                     <label class=layout1 for="gender">Jenis Kelamin:</label>
35.                     Laki-laki<input type="radio" name="gender" id="gender" value="male" />
36.                     Perempuan<input type="radio" name="gender" id="gender" value="fe-
    male" checked />
37.                 <!-- type = "checkbox" -->
38.                 <div class="field">
39.                     <label class=layout2 for="binary">Bersedia menerima iklan me-
    lalui email?</label>
40.                     <input type="checkbox" name="binary" id="binary" checked />
41.                 </div>
42.                 <div class="field">
43.                 <!-- type = "submit" -->
44.                     <input type="submit" value="Submit" name="button1" id="button1">
45.                 <!-- type = "reset" -->
46.                     <input type="reset" value="Reset" name="button2" id="button2">
47.                 </div>
48.             </fieldset>
49.         </form>
50.     </body>
51. </html>
```

Skrip CSS:

```
1.  body {
2.      background-color: white;
3.      font-family: Verdana, geneva, Arial, Helvetica, sans-serif;
4.      margin:0px;
5.      min-width: 600px; }
6.  fieldset
7.      {
8.      background-color: white;
9.      padding: 10px;
10.     margin-bottom: 100px;
11.     margin-left: 10px;
12.     margin-right: 10px; }
13. legend
14.     {
15.     font-size: large;
16.     font-weight: bold;
17.     margin-bottom: 5px;
18. }
19. .layout1
20.     {
21.     float: left;
22.     margin-right: 20px;
23.     width: 140px;
24.     text-align: left;
25. }
26. .layout2
27.     {
28.     float: left;
29.     margin-right: 20px;
30.     width: 325px;
31.     text-align: left;
32. }
33. div.field
34.     {
35.     clear: both;
36.     line-height:30px;
37.     }
```

Latihan 3.3

Skrip CSS untuk mengubah *layout* halaman *web*:

Skrip CSS:

```
1.  body {
2.      background-color: #9999FF;
3.      font-family: VErdana, geneva, Arial, Helvetica, sans-serif;
4.      margin:0px;
5.      min-width: 600px; }
6.  div.header {
7.      height: 80px;
8.      text-align: center;
9.      background-color: #003399;
10.     color: white; }
11. div.header h1 {
12.     margin: 0px;
13.     padding: 10px; }
14. div.content {
15.     padding: 10px;
16.     margin-top: 5px;
17.     margin-left: 5px; /* margin-left: 200px; */
18.     margin-right: 200px;
19.     border:1px solid black;
20.     background-color: white; }
21. div.menu {
22.     margin: 5px;
23.     padding: 5px;
24.     border:1px dashed black;
25.     background-color: #CCCCFF;
26.     float: none; /*float: left; */
27.     /* width: 180px; */ }
28. div.sidebar {
29.     margin: 5px;
30.     border: 1px dashed black;
31.     background-color: #CCCCFF;
32.     float: right;
33.     width: 180px; }
34. div.footer {
35.     margin: 5px;
36.     border: 1px dashed black;
37.     background-color: #CCCCFF;
38.     text-align: center;
39.     clear: both;
40.     height: 50px; }
41. div.sidebox{
42.     border-style: solid;
43.     border-width: 4px;
44.     border-color: #003399;
45.     width: 160px;
46.     margin: 10px;
47.     background: white; }
48. div.sidebox h1{
49.     background-color: #003399;
50.     font-size: 15pt;
51.     color: white;
52.     margin: 0px;
53.     text-align: center; }
54. div.sidebox p{
55.     margin: 10px; }
```

Perhatikan bahwa pengubahan skrip CSS dilakukan *selector class* **div.content** (baris 17) dan **div.menu** (baris 26 dan 27).

Latihan 4.1

Empat kesalahan sintaks XML adalah:

1. Tidak ada elemen *root* dokumen XML (baris 2)
2. Kesalahan *tag* penutup pada elemen **<description>** (baris 4)
3. Nilai atribut **cid** pada elemen **<course>** belum diberi tanda quote (baris 6)
4. Kesalahan nama tag pembuka pada elemen **<description>** (baris 12)

Latihan 4.2

Skrip XML untuk menambah data dosen:

Skrip XML:

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <academic>
3.      <student_data>
4.        <student>
5.           <id>12345</id>
6.           <name>Almira Wijaya</name>
7.        </student>
8.        <student>
9.            <id>12346</id>
10.           <name>Danuar Aldi</name>
11.        </student>
12.      </student_data>
13.      <lecturer_data>
14.         <lecturer>
15.             <id>A123</id>
16.             <name>Beni Bernardi</name>
17.         </lecturer>
18.         <lecturer>
19.             <id>A124</id>
20.             <name>Catur Cahyono</name>
21.         </lecturer>
22.      </lecturer_data>
23. </academic>
```

Perhatikan bahwa elemen **<student_data>** dibuat untuk mengelompokkan data pada elemen **<student>**. Dan elemen **<lecturer_data>** dibuat untuk mengelompokkan data penambahan dosen pada elemen **<lecturer>**.

Latihan 4.3

Skrip XML dan XSL yang baru:

Skrip XML:

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <?xml-stylesheet type="text/xsl" href="xml_style2.xsl"?>
3.  <academic>
4.      <student>
5.          <id>12345</id>
6.          <name>Almira Wijaya</name>
7.      </student>
8.      <student>
9.          <id>12346</id>
10.         <name>Danuar Aldi</name>
11.     </student>
12. </academic>
```

Skrip XSL:

```
1.  <?xml version="1.0"  encoding="UTF-8"?>
2.  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.      <xsl:template match="/">
4.          <html>
5.              <body style="font-family:Arial, helvetica, sans-serif; font-size:12pt;background-
    color:#FFFFFF">
6.              <xsl:for-each select="academic/student">
7.                  <div style="background-color:#EEEEEE; padding:4px">
8.                      <span style="color:black">ID  :<xsl:value-of select="id"/></span>
9.                  </div>
10.                 <div style="background-color:#EEEEEE; padding:4px">
11.                     <span style="color:black">Name: <xsl:value-of select="name"/></span>
12.                 </div>
13.             </xsl:for-each>
14.             </body>
15.         </html>
16.     </xsl:template>
17. </xsl:stylesheet>
```

Latihan 5.1

Skrip JavaScript dengan perulangan **FOR**:

Skrip JavaScript:

```
1.  number=10;
2.  for (var count = 1; count <= 10; count++)
3.  {
4.      console.log("Iteration number " + number);
5.      number--;
6.  }
```

Latihan 5.2

Skrip JavaScript dengan perulangan **WHILE**:

Skrip JavaScript:

```
1.  var count = 10;
2.  while (count >= 1)
3.  {
4.      console .log("Iteration number " + count);
5.      count--;
6.  }
```

Latihan 5.3

Skrip JavaScript untuk membuat fungsi dengan *RegExp*:

Skrip JavaScript:

```
1.  function verifyNumber(strNumber)
2.  {
3.      var patt=/([^0-9])+/g;
4.      return !patt.test(strNumber);
5.  }
```

Latihan 5.4

Skrip JavaScript untuk menambahkan validasi *form*:

Skrip JavaScript:

```
1.  <script>
2.      function validate() {
3.          if(checkFill()) {
4.              if(isIntegerNumber()){
5.                  return true;
6.              }
7.          }
8.          return false;
9.      }
10.
11.     function checkFill() {
12.         if (document.myForm.age.value == "" ) {
13.             alert ("Please fill in the 'Your Age' box.");
14.             document.myForm.age.focus();
15.             return false;
16.         }
17.         return true;
18.     }
19.
20.     function isIntegerNumber() {
21.         var i, allowed = "0123456789";
22.         var numval=document.myForm.age.value;
23.         for (i=0; i < numval.length; i++) {
24.             // if character is not a digit return false
25.             if (allowed.indexOf(numval.charAt(i)) == -1) {
26.                 window.alert ("Please fill in 'Your Age' with numeric values");
27.                 document.myForm.age.focus();
28.                 return false;
29.             }
30.         }
31.         return true;
32.     }
33. </script>
34.
35. <form name="myForm" action="successful.html" method="post" onsubmit="return validate();">
36.     Your Age: <input type="text" name="age">
37.     <input type="submit">
38. </form>
```

Perhatikan bahwa fungsi **validate** digunakan untuk menggabungkan dua buah fungsi validasi **checkFill** (untuk memastikan bahwa kotak masukan **age** sudah ada isinya) dan **isIntegerNumber** (untuk memastikan bahwa isi kotak masukan **age** adalah angka bilangan bulat).

BIOGRAFI PENULIS

Dr. Noor Ifada memperoleh gelar Sarjana bidang Teknik Elektro dari Institut Teknologi Sepuluh Nopember (ITS) Indonesia pada tahun 2000, gelar Master bidang *Information Systems Development* dari HAN University The Netherlands pada tahun 2007, dan gelar Doktor bidang *Electrical Engineering dan Computer Science* dari Queensland University of Technology (QUT) Australia pada tahun 2016.

Penulis adalah dosen dan peneliti di bidang *Web Technology* dan Sistem Rekomendasi di Universitas Trunojoyo Madura (UTM) Indonesia sejak tahun 2003. Mata kuliah yang pernah diajarkan adalah mata kuliah yang berkaitan dengan pemrograman dasar, basisdata, pemrograman *web*, dan sistem rekomendasi. Penulis juga pernah bekerja sebagai *Sessional Academic* selama masa studi doktoral di QUT. Selain memiliki kualifikasi sebagai pendidik dari Kementrian Pendidikan Nasional Republik Indonesia pada tahun 2011, penulis juga tersertifikasi sebagai *Associate Fellow* dari *The Higher Education Academy* dengan *UK Professional Standards Framework* pada tahun 2016. Buku yang pernah ditulis adalah Diktat mata kuliah Algoritma Pemrograman, Modul ajar mata kuliah Basisdata serta Pemrograman Basisdata berbasis *Web*.

Korespondensi ke penulis dapat dilakukan dengan mengirimkan *email* ke noor.ifada@ trunojoyo.ac.id.

BIOGRAFI EDITOR

Husni, S.Kom., MT. Dosen, peneliti dan praktisi di bidang *Web Technology*, *Mining and Retrieval* di Universitas Trunojoyo Madura. Sudah 20-an tahun aktif sebagai praktisi, *trainer* dan penulis di bidang Jaringan komputer, *Internet web hosting* dan Pemrograman Aplikasi *Web*. Beberapa buku karyanya yang sudah terbit di antaranya Pembangunan Jaringan Komputer menggunakan Linux Redhat, Pemrograman *Web* dengan PHP, Pemrograman Basis Data dengan Delphi, Pengembangan Aplikasi *Database* berbasis *Web* dengan PHP dan MySQL, Instalasi *Local Area Network* berbasis Linux, Pembangunan *Server Internet* berbasis Linux, *Remote Administrator* dengan RAdmin. Buku-buku tersebut diterbitkan oleh Penerbit Andi atau Graha Ilmu dan didistribusikan pada skala nasional. Pengelola situs *web* husni.trunojoyo.ac.id ini telah memberikan beberapa pelatihan yang berkaitan dengan aplikasi *web* modern seperti *Setup* layanan *Virtual Hosting Modern* berbasis Linux, Virtualisasi Aplikasi *Web* berbasis *Container* menggunakan *Docker*, Pengembangan *Search Engine* sebagai terapan *Information Retrieval*. Saat ini juga aktif sebagai konsultan IT di beberapa perusahaan dan lembaga pemerintahan terutama yang berkaiatan dengan teknologi jaringan, *web hosting* dan aplikasi *E-Government*.

Korespondensi ke editor dapat dilakukan dengan mengirimkan *email* ke husni@trunojoyo.ac.id.