



INSTITUT BISNIS
& INFORMATIKA
KESATUAN



PEMROGRAMAN

WEB

Febri Damatraseta Fairuz
Suci Sutjipto
Isnain Mulia

Septian Cahyadi
Irvan Rizky Ariansyah
Thesya Marcella

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS PARIWISATA DAN INFORMATIKA
INSTITUT BISNIS & INFORMATIKA KESATUAN
Jl. Rangga Gading No.1, RT.02/RW09, Gudang,
Kecamatan Bogor Tengah, Kota Bogor, Jawa Barat 16123

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Kuasa, yang telah memberikan rahmat-Nya sehingga Buku Pembelajaran Pemrograman Web untuk mahasiswa/i Program Studi Teknologi Informasi Fakultas Informatika dan Pariwisata Institut Bisnis dan Informatika Kesatuan Bogor ini dapat diselesaikan dengan sebaik-baiknya.

Buku pembelajaran ini dibuat sebagai pedoman dalam melakukan kegiatan pembelajaran Pemrograman Web yang merupakan kegiatan penunjang mata kuliah Pemrograman Web pada Program Studi Teknologi Informasi Fakultas Informatika dan Pariwisata Institut Bisnis dan Informatika Kesatuan Bogor. Buku pembelajaran ini diharapkan dapat membantu mahasiswa/i dalam mempersiapkan dan melaksanakan pembelajaran dengan lebih baik, terarah, dan terencana. Pada setiap topik telah ditetapkan tujuan pelaksanaan pembelajaran dan semua kegiatan yang harus dilakukan oleh mahasiswa/i serta teori singkat untuk memperdalam pemahaman mahasiswa/i mengenai materi yang dibahas.

Dalam pembuatan Buku Pembelajaran Pemrograman Web ini, penyusun menyadari masih jauh dari sempurna. Oleh karena itu penyusun mengharapkan kritik dan saran yang membangun guna penyempurnaan buku pembelajaran ini dimasa yang akan datang. Akhir kata, penyusun mengucapkan banyak terima kasih kepada semua pihak yang telah terlibat baik secara langsung maupun tidak langsung dalam pembuatan buku ini.

Bogor, 10 Februari 2021

Penyusun,

DAFTAR ISI

KATA PENGANTAR	i
BAB I MENGENAL BAHASA PEMROGRAMAN HTML.....	5
1.1 Tujuan Pembelajaran	5
1.2 Dasar Teori	5
1.2.1 Sekilah Mengenai HTML	5
1.2.2 Struktur HTML	6
1.2.3 HTML Documents	7
1.2.4 HTML Elements	9
1.3 Latihan Pembelajaran	20
BAB II PENGENALAN CASCADING STYLE SHEETS (CSS).....	26
2.1 Tujuan Pembelajaran	26
2.2 Dasar Teori	26
2.2.1 Pengenalan CSS	26
2.2.2 Cara penggunaan CSS.....	26
2.2.3 Selector	27
2.2.4 CSS Inline	32
2.2.5 CSS Internal	32
2.2.6 CSS Eksternal	33
2.2.7 Component Properties.....	34
2.3 Latihan Pembelajaran	40
BAB III PENGENALAN BAHASA PEMROGRAMAN JAVASCRIPT	45
3.1 Tujuan Pembelajaran	45
3.2 Dasar Teori	45
3.2.1 Pengantar.....	45
3.2.2 Penulisan Javascript.....	46
3.2.3 Variables.....	49
3.2.4 Tipe Data.....	50
3.2.5 Tipe Numerik	50
3.2.6 Tipe String.....	50
3.2.7 Tipe Boolean.....	51
3.2.8 Tipe Null	51
3.2.9 Operator	51
3.2.10 Masukan Data	55

3.2.11	Object Javascript.....	56
3.3	Kontrol Program.....	64
3.3.1	Percabangan	64
3.3.2	Perulangan.....	71
3.4	Latihan Pembelajaran	75
BAB IV JAVASCRIPT 2		77
4.1	Tujuan Pembelajaran	77
4.2	Dasar Teori	77
4.2.1.	Object Array	77
4.2.2.	Object Date	78
4.2.3.	Object Math	80
4.2.4.	Object String	82
4.2.5.	Object Document	84
4.2.5.	Object Window	87
4.3	Event Handler.....	92
4.3.1	Penanganan Kejadian (Event).....	93
4.3.2	Event on Submit.....	101
4.4	Latihan Pembelajaran	108
BAB V MONOLITH – LARAVEL		109
5.1	Tujuan Pembelajaran	109
5.2	Dasar Teori	109
5.2.1.	Monolith.....	109
5.2.2	Laravel	109
5.2.3.	Environment Development	110
5.2.4.	Architecture Concepts.....	112
5.3	Latihan Pembelajaran	114
BAB VI LARAVEL MVC, ROUTE DAN BLADE FRAGMENT		115
6.1	Tujuan Pembelajaran	115
6.2	Dasar Teori	115
6.2.1	MVC	115
6.2.2.	Laravel Route.....	116
6.2.3.	Laravel Blade	121
6.3	Latihan Pembelajaran	131
BAB VII DATABASES		132
7.1	Tujuan Pembelajaran	132

7.2	Dasar Teori	132
7.2.1	Migration.....	132
7.2.2	Query Builder.....	134
7.3	Latihan Pembelajaran	143
BAB VIII AUTHENTICATIONS.....		132
8.1	Tujuan Pembelajaran	144
8.2	Dasar Teori	144
8.2.1	Facades	146
8.2.2	Implement Facades.....	146
8.3	Latihan Pembelajaran	152

BAB 1

MENGENAL BAHASA PEMROGRAMAN HTML

1.1 Tujuan Pembelajaran

1. Mahasiswa mengetahui sejarah Bahasa HTML
2. Instalasi HTML
3. Mahasiswa memahami struktur program Bahasa HTML
4. Setup dan instalasi git

1.2 Dasar Teori

1.2.1 Sekilah Mengenai HTML

HTML merupakan standar umum yang digunakan untuk membangun sebuah halaman website. HTML merupakan kepanjangan dari Hyper Text Markup Language, merupakan:

- a. Menggambarkan struktur halaman website.
- b. Terdiri dari serangkaian element, dimana element HTML akan memberitahukan browser untuk menampilkan konten dari berbagai platform komputer manapun.
- c. Mengontrol tampilan dari halaman website dan kontennya. Seperti memberikan label potongan konten dengan menggunakan element HTML seperti "bagian judul", "bagian paragraf", "bagian tautan" dan lain sebagainya.

Pada tahun 1980 seorang ahli fisika, Tim Berners-Lee, dan juga seorang kontraktor di CERN (Organisasi Eropa untuk Riset Nuklir) mengusulkan dan menyusun ENQUIRE, sebuah sistem untuk ilmuwan CERN dalam membagi dokumen. Sembilan tahun kemudian, Berners-Lee mengusulkan adanya sistem markah berbasis internet. Berners-Lee menspesifikasi HTML dan menulis jaringan beserta perangkat lunaknya di akhir 1990. Pada tahun yang sama, Berners-Lee dan Robert Cailliau, insinyur sistem data CERN berkolaborasi dalam sebuah permintaan untuk pendanaan, namun tidak diterima secara resmi oleh CERN. Di catatan pribadinya sejak 1990 dia mendaftar "beberapa dari banyak daerah yang menggunakan hypertext" dan pertama-tama menempatkan sebuah ensiklopedia.

Berikut ini adalah sekilas mengenai perjalanan versi dari HTML yang telah diakui sampai saat ini:

a. HTML 1.0

Penjelasan pertama kali HTML dipublikasikan sebagai bentuk bahasa pemrograman untuk sistem markah berbasis internet ialah sebuah dokumen yang disebut "Tanda HTML", pertama kali disebutkan di Internet oleh Tim Berners-Lee pada akhir 1991. Tanda ini menggambarkan 18 elemen awal mula, versi sederhana dari HTML. Kecuali untuk tag hyperlink, yang sangat dipengaruhi oleh

SGMLguid, in-house Standard Generalized Markup Language (SGML) berbasis format dokumen di CERN.

b. HTML 2.0

Dikembangkan oleh Kelompok Kerja HTML IETF, yang ditutup pada tahun 1996. Ini menetapkan standar untuk fitur inti HTML berdasarkan praktik saat ini pada tahun 1994.

c. HTML 3.0

Rekomendasi pertama W3C untuk HTML yang mewakili konsensus tentang fitur HTML untuk tahun 1996. HTML 3.2 menambahkan fitur yang digunakan secara luas seperti tabel, applet, aliran teks di sekitar gambar, skrip super, dan subskrip, sekaligus memberikan kompatibilitas mundur dengan standar HTML 2.0 yang ada.

d. HTML 4.01

Versi ini dikenalkan pada tahun 1999 dengan menambahkan dukungan untuk lebih banyak opsi multimedia, bahasa skrip, styles sheets, fasilitas pencetakan yang lebih baik, dan dokumen yang lebih mudah diakses oleh pengguna penyandang disabilitas.

e. XHTML 1.0

Pada tahun 2000 dengan banyak fitur baru, XHTML 1.0 merupakan formulasi ulang HTML 4.01 dalam XHTML, dan menggabungkan kekuatan HTML 4 dengan kekuatan XML.

f. HTML 5.0

dibuat untuk menggantikan HTML 3.2, HTML 4. dan XHTML 1.x, HTML 5 memiliki banyak fitur baru dibandingkan pendahulunya. ini termasuk dukungan penyimpanan media offline, elemen konten yang lebih spesifik (seperti footer, header, navigasi, dll), doctype inline yang lebih sederhana, dukungan penyematian audio, dan video. Pembaharuan ini diperkenalkan pada tahun 2015.

g. HTML 5.1

Selang satu tahun dari peluncuran HTML 5.0, bahasa pemrograman standar berbasis internet ini sudah memiliki pembaharuan seperti kemampuan yang lebih baik terkait pengalaman video, formulir web, aksesibilitas gambar, dan pemeriksaan ejaan dan tata bahasa.

h. HTML 5.2

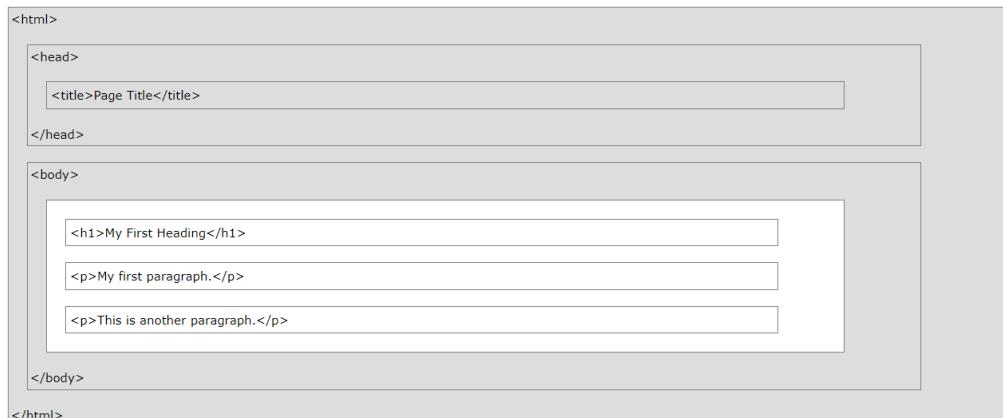
Peningkatan kebijakan keamanan konten, API permintaan pembayaran untuk situs web eCommerce, aksesibilitas aplikasi rich internet untuk penyandang disabilitas, dan perubahan pada elemen <main> untuk mendukung desain responsif. Versi HTML 5.2 dikenalkan pada tahun 2017 dan masih dipergunakan hingga saat ini sebagai bahasa pemrograman standar berbasis internet yang direkomendasikan oleh W3C.

1.2.2 Struktur HTML

Sifat dari HTML ialah clientscript dimana dokumen tersebut dapat dibuka didalam komputer secara stand alone yang tidak membutuhkan sebuah server untuk dapat menampilkanya di dalam browser. Dokumen HTML biasanya berekstensi *.html*

atau *.htm* dimana bahasa HTML tersebut tersusun atas tag atau syntax yang berformat <tagname>...</tagname>. Aturan dalam penulisan HTML sebagai berikut:

- a. Dalam penulisan tag HTML diapit dengan menggunakan dua kurung siku (angle bracket) “<” dan “>”
- b. Tag HTML selalu berpasangan, dimana tag awal dalam suatu syntax disebut dengan tag awal dan tag kedua sebagai tag terakhir. contoh: <p>...</p>
- c. Penulisan tag HTML tidak case sensitive dimana tag dengan huruf kecil sama dengan tag huruf besar, contoh: <P>...</P> sama dengan <p>...</p> Namun berdasarkan aturan pada W3C (HTML5.0 dan XHTML) penulisan tag HTML sebaiknya menggunakan huruf kecil.
- d. Jika memiliki tag nested atau terdapat suatu tag berisi tag lagi, maka penulisan tag akhir tidak boleh bersilang dan harus berurutan. Contoh: <p>Ini contoh Huruf Tebal dan <i>Huruf cetak miring</i></p>
- e. Penulisan syntax dokumen HTML selalu diawali dengan tag <html> dan diakhiri dengan tag </html>



Gambar 1.2.2.1 Stuktur HTML

1.2.3 HTML Documents

Semua dokumen HTML harus dimulai dengan deklarasi tipe dokumen: <!DOCTYPE html>. Dokumen HTML itu sendiri diawali dengan <html> dan diakhiri dengan </html>. Bagian yang terlihat dari dokumen HTML adalah antara <body> dan </body>.

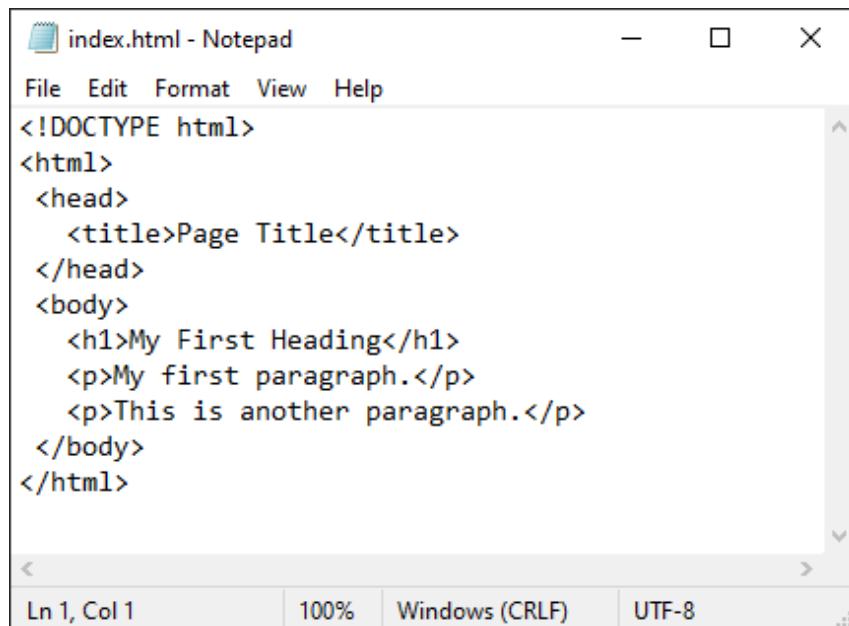
```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

Gambar 1.2.3.1 Simple Document HTML

Dari contoh penulisan dokumen HTML diatas dapat diartikan sebagai berikut:

- a. Deklarasi `<!DOCTYPE html>` mendefinisikan bahwa dokumen ini adalah dokumen HTML5.
- b. Elemen `<html>` adalah elemen root dari halaman HTML.
- c. Elemen `<head>` berisi informasi meta tentang halaman HTML.
- d. Elemen `<title>` menentukan judul untuk halaman HTML (yang ditampilkan di bilah judul browser atau di tab halaman).
- e. Elemen `<body>` mendefinisikan tubuh dokumen, dan merupakan wadah untuk semua konten yang terlihat, seperti judul, paragraf, gambar, hyperlink, tabel, daftar, dll.
- f. Elemen `<h1>` mendefinisikan heading besar.
- g. Elemen `<p>` mendefinisikan paragraf.

Dalam membangun sebuah halaman website dengan menggunakan HTML, text editor yang dapat digunakan cukup sederhana dimana aplikasi text editor ini terdapat di semua operating sistem pada komputer. Jika menggunakan Windows OS dapat menggunakan aplikasi Notepad, sedangkan jika menggunakan UNIX OS dapat menggunakan aplikasi TextEdit.



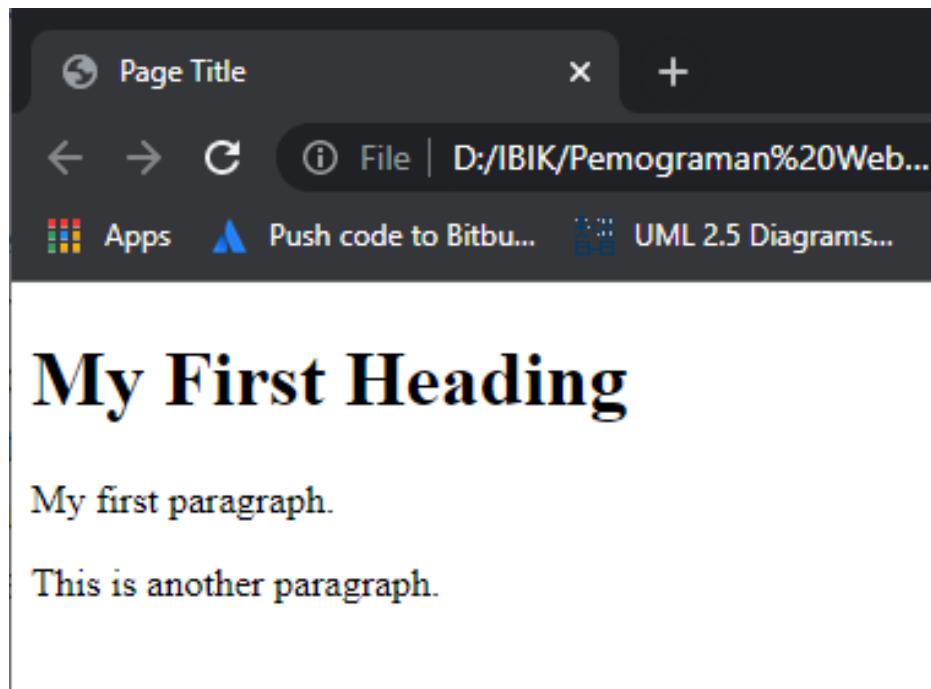
The screenshot shows a Microsoft Notepad window titled "index.html - Notepad". The window contains the following HTML code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

The status bar at the bottom of the window displays "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

Gambar 1.2.3.2 HTML Documents dalam text editor

Untuk membuat sebuah dokumen HTML tahap pertama ialah buka text editor yang dimiliki seperti Notepad dan masukan script dibawah ini, lalu simpan dengan nama `index.html` atau `index.htm`. Setelah berhasil tersimpan dengan format extension `.html` atau `.htm`, maka file tersebut akan membentuk sebuah file dengan format icon browser. Jika dibuka file tersebut maka akan membuka sebuah aplikasi browser dan menampilkan sebuah tampilan berdasarkan tag HTML.



Gambar 1.2.3.3 Hasil Preview dari HTML Documents

1.2.4 HTML Elements

1.2.4.1 HTML Element Head

Berikut adalah tag atau elemen yang dapat digunakan di bagian <head>...</head> pada sebuah dokumen HTML:

1. Title

Elemen <title>, menentukan judul untuk halaman HTML (yang ditampilkan di bilah judul browser atau di tab halaman)

```
<title>Page Title</title>
```

2. Favicon

Elemen ini diperuntukan untuk menentukan icon pada halaman HTML (yang ditampilkan di bilah judul browser atau di tab halaman). Untuk menggunakan favicon, dikhkususkan memiliki file gambar dengan extension .ico, .svg atau .png dan size yang digunakan ialah 16x16 pixel atau 32x32 pixel.

```
<link rel="icon" type="image/x-icon" href="/path/to/icons/favicon.ico">  
<link rel="icon" type="image/png" href="/path/to/icons/favicon.png">  
<link rel="icon" type="image/svg" href="/path/to/icons/favicon.svg">
```

3. Meta

Elemen <meta> biasanya digunakan untuk menentukan set karakter, deskripsi halaman, kata kunci, penulis dokumen, dan pengaturan *viewport*. Metadata tidak akan ditampilkan di halaman, tetapi digunakan oleh browser (cara menampilkan konten atau memuat ulang halaman), oleh mesin pencari (kata kunci), dan layanan web lainnya.

- Menentukan set karakter

```
<meta charset="UTF-8">
```

- Menentukan kata kunci untuk search engine

```
<meta name="keywords" content="HTML, CSS, JavaScript">
```

- Menentukan deskripsi halaman web

```
<meta name="description" content="Website IBIK – Pemrograman Web">
```

- Menentukan penulis halaman

```
<meta name="author" content="Febry D Fairuz">
```

- Merefresh halaman setiap 30 detik

```
<meta http-equiv="refresh" content="30">
```

- Mengatur *viewport* agar situs website terlihat bagus di semua perangkat

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

4. Javascript

JavaScript membuat halaman HTML lebih dinamis dan interaktif. Tag HTML <script> digunakan untuk mendefinisikan skrip sisi klien (*JavaScript*). Elemen <script> berisi pernyataan skrip, atau menunjuk ke file skrip eksternal melalui atribut src. Penggunaan umum untuk *JavaScript* adalah manipulasi gambar, validasi formulir, dan perubahan konten yang dinamis. Untuk memilih elemen HTML, *JavaScript* paling sering menggunakan metode document.getElementById(). Contoh *JavaScript* ini menulis "Halo *JavaScript*!" menjadi elemen HTML dengan id="demo":

```
<p id="demo"></p>

<script>
  document.getElementById("demo").innerHTML = "Hello
  JavaScript!";
</script>
```

5. Styles CSS

Cascading Style Sheets (CSS) digunakan untuk memformat tata letak halaman web. Dengan CSS, dapat mengontrol warna, font, ukuran teks, jarak antar elemen, bagaimana elemen diposisikan dan ditata, gambar latar belakang atau warna latar apa yang akan digunakan, tampilan berbeda untuk perangkat dan ukuran layar yang berbeda, dan lebih banyak!

```

<style>
    body {background-color: powderblue;}
    h1 {color: blue;}
    p {color: red;}
</style>

```

1.2.4.2 HTML Element Body

Sedangkan berikut ini adalah tag atau elemen yang bisa digunakan di bagian <body>...</body> pada sebuah dokumen HTML:

- **Elements Contents HTML**

1. Headings

Headings adalah judul atau subjudul yang ingin ditampilkan di halaman web. *Headings* didefinisikan dengan tag <h1> hingga <h6>. <h1> mendefinisikan heading yang paling penting. <h6> mendefinisikan heading yang paling tidak penting.

```

<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>

```

2. Paragraphs

Sebuah paragraf selalu dimulai pada baris baru, dan biasanya berupa blok teks. Elemen HTML <p> mendefinisikan sebuah paragraf. Sebuah paragraf selalu dimulai pada baris baru, dan browser secara otomatis menambahkan beberapa spasi (margin) sebelum dan sesudah paragraf.

```

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

```

3. Formatting

HTML berisi beberapa elemen untuk mendefinisikan teks dengan arti khusus. Elemen pemformatan dirancang untuk menampilkan jenis teks khusus:

Element	Description
	Bold
	Important
<i>	Italic

<code></code>	Emphasized
<code><mark></code>	Marked
<code><small></code>	Smaller
<code></code>	Deleted text
<code><sub></code>	Subscript text
<code><sup></code>	Superscript text

4. Quotations

Bagian ini kita akan membahas elemen HTML `<blockquote>`, `<q>`, `<abbr>`, `<address>`, `<cite>`, dan `<bdo>`.

Element	Description
<code><abbr></code>	singkatan atau akronim
<code><address></code>	informasi kontak untuk penulis/pemilik dokumen
<code><bdo></code>	arah teks
<code><blockquote></code>	bagian yang dikutip dari sumber lain
<code><cite></code>	judul sebuah karya
<code><q></code>	kutipan inline pendek

5. Comments

Untuk menambahkan komentar ke sumber HTML Anda dengan menggunakan sintaks berikut:

```
<!-- Write your comments here --&gt;</pre>

```

6. List

List HTML memungkinkan programmer untuk mengelompokkan satu set item terkait dalam bentuk list. Daftar tidak berurutan dimulai dengan tag `` atau ``. Setiap item list dimulai dengan tag ``.

Syntax	Display
<pre style="background-color: #f0f0f0; padding: 5px;"><code> Coffee Tea Milk </code></pre>	<ul style="list-style-type: none"> • Coffee • Tea • Milk

<pre> Coffee Tea Milk </pre>	<p>1. Coffee 2. Tea 3. Milk</p>
--	---

7. Block and Inline

Setiap elemen HTML memiliki nilai tampilan default, tergantung pada jenis elemennya. Ada dua nilai tampilan: block dan inline.

- Block Element

Elemen level blok selalu dimulai pada baris baru. Elemen level blok selalu menggunakan lebar penuh yang tersedia (membentang ke kiri dan kanan sejauh mungkin). Elemen level blok memiliki margin atas dan bawah, sedangkan elemen inline tidak.

<pre><div>Hello World</div></pre>

- Inline Element

Elemen Inline tidak dimulai pada baris baru. Elemen Inline hanya membutuhkan lebar sebanyak yang diperlukan. Ini adalah elemen `` di dalam paragraf.

<pre>Hello World</pre>

8. Links

Links ditemukan di hampir semua halaman web. Links memungkinkan pengguna mengklik jalan mereka dari halaman ke halaman lain. Links HTML disebut dengan *hyperlink*. Untuk dapat mengklik tautan dan melompat ke halaman lain. Saat Anda menggerakkan mouse di atas tautan, panah mouse akan berubah menjadi tangan kecil.

Atribut terpenting dari elemen `<a>` adalah atribut `href`, yang menunjukkan tujuan tautan. Teks tautan adalah bagian yang akan terlihat oleh pembaca. Mengklik teks tautan, akan mengirim pembaca ke alamat URL yang ditentukan.

<pre> Visit IBIK</pre>

HTML Links - The target Attribute

Secara default, halaman tertaut akan ditampilkan di jendela browser saat ini. Untuk mengubahnya, Anda harus menentukan target lain untuk tautan tersebut.

Atribut target dapat memiliki salah satu dari nilai berikut:

Target	Description
<_self>	Membuka windows atau tab dalam satu halaman yang sama
<_blank>	Membuka windows atau tab baru
<_parent>	Membuka windows atau tab di dalam elemen induk
<_top>	Membuka windows atau tab secara keseluruhan pada body

Create Bookmarks

Link HTML dapat digunakan untuk membuat bookmark, sehingga pembaca dapat melompat ke bagian tertentu dari halaman web.

```
<h2 id="C4">Chapter 4</h2>
<a href="#C4">Jump to Chapter 4</a>
```

9. Images

Gambar dapat meningkatkan desain dan tampilan halaman web. Tag HTML digunakan untuk menyematkan gambar di halaman web. Gambar secara teknis tidak dimasukkan ke dalam halaman web; gambar ditautkan ke halaman web. Tag membuat ruang penyimpanan untuk gambar yang direferensikan. Tag kosong, hanya berisi atribut, dan tidak memiliki tag penutup. Tag memiliki dua atribut yang diperlukan:

- o src = Menentukan jalur ke gambar
- o alt = Menentukan teks alternatif untuk gambar.
- o width = Menentukan panjang
- o height = Menentukan tinggi

```

```

10. Tables

Tabel HTML memungkinkan pengembang web untuk mengatur data ke dalam baris dan kolom.

Table Kolom

```
<table>
<tr>
  <td>Emil</td>
  <td>Tobias</td>
  <td>Linus</td>
```

```

</tr>
</table>

```

Each table cell is defined by a <td> and a </td> tag. Segala sesuatu di antara <td> dan </td> adalah konten sel tabel.

Table Baris

```

<table>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
  <tr>
    <td>16</td>
    <td>14</td>
    <td>10</td>
  </tr>
</table>

```

Setiap baris tabel dimulai dengan <tr> dan diakhiri dengan tag </tr>. Berikut ini adalah tag elemen tabel yang dapat dipergunakan:

Element	Description
<table>	Mendefinisikan tabel
<th>	Mendefinisikan sel header dalam tabel
<tr>	Mendefinisikan baris dalam tabel
<td>	Mendefinisikan sel dalam tabel
<caption>	Mendefinisikan judul tabel
<colgroup>	Menentukan grup dari satu atau lebih kolom dalam tabel untuk pemformatan
<col>	Menentukan properti kolom untuk setiap kolom dalam elemen <colgroup>
<thead>	Mengelompokkan konten header dalam sebuah tabel
<tbody>	Mengelompokkan konten tubuh dalam sebuah tabel
<tfoot>	Mengelompokkan konten footer dalam tabel

11. Iframe

Tag HTML <iframe> menentukan bingkai sebaris. Bingkai sebaris digunakan untuk menyematkan dokumen lain dalam dokumen HTML saat ini.

```
<iframe src="url" title="description" width="100px"  
height="50px"></iframe>
```

12. Audio/Video

HTML5 memiliki metode, properti, dan peristiwa untuk elemen <audio> dan <video>. Atribut kontrol menambahkan kontrol audio, seperti putar, jeda, dan volume. Elemen <source> memungkinkan Anda menentukan file audio alternatif yang dapat dipilih oleh browser. Browser akan menggunakan format pertama yang dikenali. Teks diantara tag <audio> dan </audio> hanya akan ditampilkan di browser yang tidak mendukung elemen <audio>.

```
<audio controls>  
  <source src="lagu-1.ogg" type="audio/ogg">  
  <source src="lagu-2.mp3" type="audio/mpeg">  
  Your browser does not support the audio element.  
</audio>
```

Elemen HTML <video> digunakan untuk menampilkan video di halaman web.

```
<video width="320" height="240" controls>  
  <source src="vidio-1.mp4" type="video/mp4">  
  <source src="video-2.ogg" type="video/ogg">  
  Your browser does not support the video tag.  
</video>
```

Atribut kontrol menambahkan kontrol video, seperti putar, jeda, dan volume. Sebaiknya selalu sertakan atribut lebar dan tinggi. Jika tinggi dan lebar tidak disetel, halaman mungkin berkedip saat video dimuat. Elemen <source> memungkinkan Anda menentukan file video alternatif yang dapat dipilih oleh browser. Browser akan menggunakan format pertama yang dikenali. Teks antara tag <video> dan </video> hanya akan ditampilkan di browser yang tidak mendukung elemen <video>.

1.2.4.3 HTML Element Content Form

Form HTML digunakan untuk mengumpulkan input data. Input data paling sering dikirim ke server untuk diproses.

Elemen HTML <form> digunakan untuk membuat formulir HTML untuk input pengguna:

First name: <input type="text" value="Febry"/>	<form>
Last name: <input type="text" value="Fairuz"/>	<label for="fname">First name:</label> <input type="text" id="fname" name="fname">
<input type="button" value="Submit"/>	<label for="lname">Last name:</label> <input type="text" id="lname" name="lname"></form>

1. Form Attributes

List atribut <form> yang dapat digunakan:

Attribute	Description
accept-charset	Menentukan pengkodean karakter yang digunakan untuk pengiriman formulir
action	Menentukan kemana harus mengirim formulir-data saat formulir dikirimkan
autocomplete	Menentukan apakah formulir harus memiliki pelengkapan otomatis aktif atau nonaktif
enctype	Menentukan bagaimana formulir-data harus dikodekan saat mengirimkannya ke server (hanya untuk metode = "posting")
method	Menentukan metode HTTP yang akan digunakan saat mengirim data formulir
name	Menentukan nama formulir
novalidate	Menentukan bahwa formulir tidak boleh divalidasi saat dikirimkan
rel	Menentukan hubungan antara sumber daya yang ditautkan dan dokumen saat ini
target	Menentukan tempat untuk menampilkan respons yang diterima setelah mengirimkan formulir

2. Form Elements

Elemen <form> HTML dapat berisi satu atau lebih elemen form sebagai berikut:

- Elemen <input>

Salah satu elemen form yang paling sering digunakan adalah elemen <input>. Elemen <input> dapat ditampilkan dalam beberapa cara, tergantung pada atribut type.

```
<label for="fname">First name:</label>  
<input type="text" id="fname" name="fname">
```

- Elemen <label>

Elemen <label> mendefinisikan label untuk beberapa elemen form. Elemen <label> berguna untuk pengguna pembaca layar, karena pembaca layar akan membacakan label dengan keras saat pengguna fokus pada elemen input.

Elemen <label> juga membantu pengguna yang mengalami kesulitan mengklik wilayah yang sangat kecil (seperti tombol radio atau kotak centang) - karena ketika pengguna mengklik teks di dalam elemen <label>, itu akan mengaktifkan tombol radio/kotak centang. Atribut for dari tag <label> harus sama dengan atribut id dari elemen <input> untuk mengikatnya bersama.

- Elemen <textarea>

Elemen <textarea> mendefinisikan bidang input multi-baris (area teks):

```
<textarea name="message" rows="10" cols="30">  
Type something in here  
</textarea>
```

Atribut *rows* menentukan jumlah baris yang terlihat di area teks. Atribut *cols* menentukan lebar yang terlihat dari area teks.

- Elemen <button>

Elemen <button> mendefinisikan tombol yang dapat diklik:

```
<button type="button">Click Me! Button</button>  
<button type="submit">Click Me! Submit</button>  
<button type="reset">Click Me! Reset</button>
```

- Elemen <select>

Elemen <select> mendefinisikan daftar drop-down:

```
<label for="cars">Choose a car:</label>  
<select id="cars" name="cars">  
  <option value="volvo">Volvo</option>  
  <option value="saab">Saab</option>  
  <option value="fiat">Fiat</option>  
  <option value="audi">Audi</option>  
</select>
```

The <option> elements defines an option that can be selected. By default, the first item in the drop-down list is selected. To define a pre-selected option, add the selected attribute to the option:

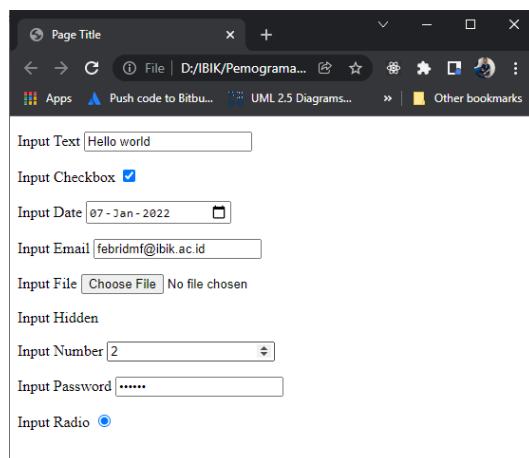
```
<option value="fiat" selected>Fiat</option>
```

3. Input Types

Berikut adalah berbagai jenis input yang dapat digunakan dalam HTML:

<input type="text">	<input type="hidden">
<input type="checkbox">	<input type="number">
<input type="date">	<input type="password">
<input type="email">	<input type="radio">
<input type="file">	<input type="range">

Display:



4. Input Attributes

Bagian ini menjelaskan atribut yang berbeda untuk elemen <input> HTML.

Attribute	Description
value	Menentukan nilai awal untuk bidang input
readonly	Menentukan bahwa bidang input hanya-baca
disabled	Menentukan bahwa bidang input harus dinonaktifkan
size	Menentukan lebar yang terlihat, dalam karakter, dari bidang input
maxlength	Menentukan jumlah karakter maksimum yang diizinkan dalam bidang input
min & max	Menentukan nilai minimum dan maksimum untuk bidang input

pattern	Menentukan ekspresi reguler yang memeriksa nilai bidang input, saat formulir dikirimkan.
placeholder	Menentukan petunjuk singkat yang menjelaskan nilai yang diharapkan dari bidang input (nilai sampel atau deskripsi singkat tentang format yang diharapkan)
required	Menentukan bahwa bidang input harus diisi sebelum mengirimkan formulir

1.3 Latihan Pembelajaran

Kerjakan dan unggahlah jawaban ke repository GitHub Anda!

1. Latihan *Typography*

Buatlah dokumen HTML bernama Latihan-1.

Buatlah sebuah artikel yang berisi *Heading*, *Paragraph* dan *Formatting*.

Minimum penggunaan paragraf sebanyak 4 buah.

2. Latihan *Image*

Buatlah dokumen HTML bernama Latihan-2.

Dari dokumen HTML Latihan-1, tambahkanlah elemen Image. Penggunaan image minimum 2 buah. Dan tambahkan atribut gambar untuk mengatur panjang dan lebar dari gambar tersebut.

3. Latihan *Table*

Buatlah dokumen HTML bernama Latihan-3.

Silakan implementasikan bentuk dibawah ini kedalam scripting HTML:

Oksigen mempunyai beberapa SIFAT FISIKA, diantaranya yang terdapat dalam tabel berikut:

Sifat Fisika	Oksigen
Massa atom <i>relative</i>	15,994
Nomor Atom	8
Konfigurasi <i>electron</i>	$2s^2 2p^4$
Jari-jari atom (nm)	0,074
Jari-jari X^{2-} (nm)	0.140
Keelectronegatifan	3,5
Energy ionisasi I (kJ/mol)	1316
Energy ionisasi II (kJ/mol)	3396

Kerapatan (g/cm ³)	1.27 (padatan)	
Titik leleh (°C)	+183	
Titik beku (°C)	-219	
Potensial elektroda (V)	+0.401	
X _{2(g)} +2e ⁺ _(aq) ⇌ 2X ⁻ _(aq)	-	

4. Latihan List

Buatlah dokumen bernama Latihan-4.

Ubahlah bentuk dibawah ini kedalam scripting HTML:

The 10 Most Popular Programming Languages to Learn in 2022

There's no question that software programming is a hot career right now. The [U.S. Bureau of Labor Statistics](#) projects 21 percent growth for programming jobs from 2018 to 2028, which is more than four times the average for all occupations. What's more, the median annual pay for a software programmer is about \$106,000, which is nearly three times the median pay for all U.S. workers.

Not all programming jobs are the same, however. Different roles, companies, and types of software require knowing and understanding different programming languages—and it's often beneficial to know multiple languages. Trying to break into the field of software programming can be a daunting experience, especially for professionals with no prior programming experience.

The [Master of Science in Computer Science Align program](#) at Northeastern University is specifically designed for students who want to transition into computer science from another field of study.

“Our aim is to transport students from a variety of different backgrounds and have them come out as software engineers,” says Ian Gorton, PhD and

director of the graduate computer science programs at Northeastern University—Seattle. “*We focus on math, programming, and a variety of computer science and engineering concepts.*”

Whether you’re new to programming or looking to brush up on your skills, it helps to know which languages are in high demand. Here are 10 of the most popular programming languages of 2020 based on the number of job postings listed on job search sites. Indeed, the average annual salary for those jobs, and factors such as ease of use and potential for growth.

Top 10 Most Popular Programming Languages

1. Python
Average annual salary: \$120.000
2. JavaScript
Average annual salary: \$118.000
3. Java
Average annual salary: \$104.000
4. C#
Average annual salary: \$97.000
5. C
Average annual salary: \$97.000
6. C++
Average annual salary: \$97.000
7. Go
Average annual salary: \$93.000
8. R
Average annual salary: \$93.000
9. Swift
Average annual salary: \$93.000
10. PHP
Average annual salary: \$81.000

7 Other Programming Languages to Consider

- **Web-based** startups are more likely to be programming in **Python** and **JavaScript**.
- Larger companies tend to develop their internal software applications using **C#** or **Java** and their **Web applications** using **PHP**.
- Programs for data *analytics* typically use the **R** and **MATLAB** programming languages.

- Embedded devices, such as those in the automotive and healthcare industries, run software written in **C, C++, or Rust**.
- Applications that run on the cloud are increasingly written in **Go or Scala**.
- **Mobile applications** are increasingly written in Swift or Kotlin.

5. Latihan Iframe, *Block* dan *Inline*

Buatlah dokumen HTML bernama *Latihan-5*.

a. Ubahlah bentuk dibawah ini menjadi script HTML:

Programming Language Base on Career



Top 10 Programming Language in 2022

...*Iframe to Latihan-4...*

List of programming languages from Wikipedia



The screenshot shows a Wikipedia article titled "List of programming languages". The page contains a grid of programming language names, each with a small icon. The categories include:

- Front-end Web Development: JS, Elm, TypeScript
- Back-end Web Development: JS, Scala, GO, Ruby, Python
- Mobile Development: JS, Java, Objective C, Swift

On the right side of the page, there is a sidebar titled "Programming language lists" with the following options:

- Alphabetical
- Complex
- Chronological
- Generational

- Buatlah 1 buah iframe yang memanggil file dokumen *Latihan-4*
- Buatlah 1 buah iframe yang memanggil sebuah url

6. Latihan Link

Buatlah dokumen HTML bernama Latihan-6.

a. Implementasikan bentuk dibawah ini kedalam script HTML:

Top 10 Daftar Bahasa Pemrograman

Daftar isi:

- [Python](#)
- [JavaScript](#)
- [Java](#)
- [C#](#)
- [C](#)
- [C++](#)
- [Go](#)
- [R](#)
- [Swift](#)
- [PHP](#)

Python

Python adalah ...

JavaScript

JavaScript adalah ...

... dan seterusnya...

Buatlah *hyperlink* pada elemen **Daftar Isi**, dimana link tersebut merujuk ke *different area*, ketika mengklik salah satu *link* tersebut maka akan merujuk ke area yang telah dituju. Contoh: jika mengklik *link Java*, maka akan dirujuk ke area *Java*.

b. Buatlah tiga buah link baru dimana:

- 1) Link “*Go to Latihan-4*” menuju ke dokumen HTML Latihan-4 dengan menggunakan target [_self](#)
- 2) Link “*Go to Latihan-5*” menuju ke dokumen HTML Latihan-5 dengan menggunakan target [_blank](#)
- 3) Link “*Go to URL*” menuju URL dengan menggunakan target [_parent](#)

7. Latihan Form

Buatlah dokumen HTML bernama *Latihan-7*.

Buatlah form yang berisi isian field sebagai berikut:

- NPM (input text)
- Fullname (input text)
- Email (input email)
- Password (input password)
- Place Birth (input text)
- Birthdate (input date)
- Gender (radio button, *Male and Female*)
- Address (textarea)

Dari form diatas memiliki dua buah button. Button submit dengan nama Save dan button reset dengan nama Clear.

8. Latihan Layout

Buatlah folder project dengan nama *my-portfolio*. Di Dalam folder tersebut terdapat dua buah folder yaitu folder *css* dan *images*. Dan terdapat satu buah file bernama *index.html*. Dimana file tersebut berisi script HTML yang akan menampilkan bentuk portofolio atau curriculum vitae pada *browser*. Silakan membuat bentuk portofolio dengan bentuk bebas namun mengimplementasikan bentuk elemen HTML yang telah diberikan minimal 5 buah elemen.

BAB 2

PENGENALAN CASCADING STYLE SHEETS (CSS)

2.1 Tujuan Pembelajaran

Tujuan dari pembelajaran pada chapter ini ialah memperkenalkan template yang berupa style untuk mempercantik tampilan website

2.2 Dasar Teori

2.2.1 Pengenalan CSS

CSS atau Cascading Style Sheets adalah sebuah dokumen yang berisi aturan yang digunakan untuk memisahkan isi dengan layout dalam halaman-halaman web yang dibuat. CSS memperkenalkan “template” yang berupa style untuk dibuat dan mengijinkan penulisan kode yang lebih mudah dari halaman-halaman web yang dirancang. CSS mampu menciptakan halaman yang tampak sama pada resolusi layar yang berbeda-beda tanpa memerlukan penggunaan tabel seperti pada html klasik.

Umumnya HTML memiliki beberapa elemen semantik yang mendefinisikan berbagai bagian halaman web seperti gambar dibawah ini:



- <header> □ Mendefinisikan header untuk dokumen atau bagian
- <nav> □ Mendefinisikan sekumpulan link navigasi
- <section> □ Mendefinisikan bagian dalam dokumen
- <article> □ Mendefinisikan konten mandiri yang mandiri
- <aside> □ Mendefinisikan konten selain dari konten (seperti sidebar)
- <footer> □ Mendefinisikan footer untuk dokumen atau bagian

Gambar diatas menggambarkan sebuah Layouting dari sebuah template website. Untuk mempercantik tampilan dari layout tersebut dapat menggunakan CSS sebagai template.

CSS dapat ditambahkan ke dokumen HTML dengan 3 cara:

- Inline - dengan menggunakan atribut style di dalam elemen HTML
- Internal - dengan menggunakan elemen <style> di bagian <head>
- Eksternal - dengan menggunakan elemen <link> untuk menautkan ke file CSS eksternal

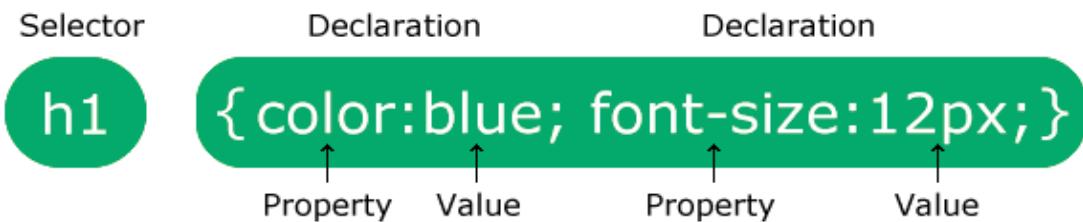
Cara paling umum untuk menambahkan CSS, adalah dengan menyimpan style di file CSS eksternal.

2.2.2 Cara penggunaan CSS

Aturan CSS terdiri dari selector dan blok deklarasi:

- a. Selector menunjuk ke elemen HTML yang ingin diberikan style. Blok deklarasi berisi satu atau lebih deklarasi yang dipisahkan oleh titik koma (,).

- b. Setiap deklarasi menyertakan nama properti CSS dan nilai, dipisahkan oleh titik dua(:).
- c. Beberapa deklarasi CSS dipisahkan dengan titik koma, dan blok deklarasi dikelilingi oleh kurung kurawal.



Contoh pemberian style pada element HTML paragraph <p> :

```
p {
  color: red;
  text-align: center;
}
```

Penjelasan:

- **p** adalah pemilih di CSS (ini menunjuk ke elemen HTML yang ingin diberikan style: <p>).
- **color** adalah *properti*, dan **red** adalah *nilai properti*
text-align adalah *properti*, dan **center** adalah *nilai properti*

2.2.3 Selector

Selector CSS digunakan untuk "menemukan" (atau memilih) elemen HTML yang ingin diberikan style. Pemilih selector CSS dibagi menjadi lima kategori, yaitu:

- a. Selektor sederhana (pilih elemen berdasarkan nama, id, kelas)
- b. Selektor kombinator (memilih elemen berdasarkan hubungan spesifik di antara mereka)
- c. Selector Pseudo-class (memilih elemen berdasarkan status tertentu)
- d. Selector Pseudo-element (memilih dan menata bagian elemen)
- e. Selector atribut (memilih elemen berdasarkan atribut atau nilai atribut)

Contoh syntax selector:

- Selektor Sederhana

<pre>p { text-align: right; color: red; }</pre>	<pre>#para1 { text-align: center; color: blue; }</pre>	<pre>p.description { text-align: left; color: green; }</pre>
<p><p>Every paragraph will be affected by the style.</p></p> <p><p id="para1">Me too!</p></p>		

```
<p class="description">And me!</p>
```

- Selektor kombinator

```
div p {  
    background-color: yellow;  
}  
  
<h1>Descendant Selector</h1>  
  
<div>  
    <p>Every paragraph will be affected by the style.</p>  
</div>
```

- Selector *Pseudo-class*

Pseudo-class digunakan untuk mendefinisikan keadaan khusus suatu elemen. Misalnya, dapat digunakan untuk:

- style elemen saat pengguna mengarahkan mouse ke atasnya
- style tautan yang dikunjungi dan yang belum dikunjungi secara berbeda
- style elemen saat mendapat fokus

Syntax *pseudo-class*

```
selector:pseudo-class {  
    property: value;  
}
```

Contoh:

```
.para1:hover {  
    background-color: tomato;  
    font-weight: bold;  
}  
  
<h1>Descendant Selector</h1>  
  
<div id="content">  
    <p class="para1">Every paragraph will be affected by the style.</p>  
    <p>Every paragraph will be affected by the style.</p>  
</div>
```

Berikut ini adalah beberapa pseudo-class yang dapat digunakan:

Selector	Example	Example description
<u>:active</u>	a:active	Selects the active link
<u>:checked</u>	input:checked	Selects every checked <input> element
<u>:disabled</u>	input:disabled	Selects every disabled <input> element
<u>:empty</u>	p:empty	Selects every <p> element that has no children
<u>:enabled</u>	input:enabled	Selects every enabled <input> element
<u>:first-child</u>	p:first-child	Selects every <p> elements that is the first child of its parent
<u>:first-of-type</u>	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
<u>:focus</u>	input:focus	Selects the <input> element that has focus
<u>:hover</u>	a:hover	Selects links on mouse over
<u>:in-range</u>	input:in-range	Selects <input> elements with a value within a specified range
<u>:invalid</u>	input:invalid	Selects all <input> elements with an invalid value
<u>:lang(language)</u>	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
<u>:last-child</u>	p:last-child	Selects every <p> elements that is the last child of its parent
<u>:last-of-type</u>	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
<u>:link</u>	a:link	Selects all unvisited links
<u>:not(selector)</u>	:not(p)	Selects every element that is not a <p> element
<u>:nth-child(n)</u>	p:nth-child(2)	Selects every <p> element that is the second child of its parent
<u>:nth-last-child(n)</u>	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child

<u>:nth-last-of-type(n)</u>	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
<u>:nth-of-type(n)</u>	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
<u>:only-of-type</u>	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
<u>:only-child</u>	p:only-child	Selects every <p> element that is the only child of its parent
<u>:optional</u>	input:optional	Selects <input> elements with no "required" attribute
<u>:out-of-range</u>	input:out-of-range	Selects <input> elements with a value outside a specified range
<u>:read-only</u>	input:read-only	Selects <input> elements with a "readonly" attribute specified
<u>:read-write</u>	input:read-write	Selects <input> elements with no "readonly" attribute
<u>:required</u>	input:required	Selects <input> elements with a "required" attribute specified
<u>:root</u>	root	Selects the document's root element
<u>:target</u>	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
<u>:valid</u>	input:valid	Selects all <input> elements with a valid value
<u>:visited</u>	a:visited	Selects all visited links

- Selector Pseudo-element

Elemen pseudo CSS digunakan untuk menata bagian tertentu dari suatu elemen. Misalnya, dapat digunakan untuk:

- Gaya huruf pertama, atau baris, dari suatu elemen
- Sisipkan konten sebelum, atau setelah, konten elemen

Syntax pseudo-elements

```
selector::pseudo-element {
    property: value;
}
```

Contoh:

```
p::first-line {
    color: #ff0000;
    font-variant: small-caps;
}
```

`<p>`You can use the ::first-line pseudo-element to add a special effect to the first line of a text. Some more text. And even more, and more.

Berikut ini adalah beberapa pseudo-elements yang dapat digunakan:

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::marker	::marker	Selects the markers of list items
::selection	p::selection	Selects the portion of an element that is selected by a user

- Selector atribut

Selector [attribute] digunakan untuk memilih elemen dengan atribut tertentu. Dimungkinkan untuk menata elemen HTML yang memiliki atribut atau nilai atribut tertentu. Contoh berikut memilih semua <a> elemen dengan atribut target:

Syntax atribut

<code>a[target] { background-color: yellow; }</code>	<code>a[target="_blank"] { background-color: purple; }</code>
--	---

```
<h2>CSS [attribute="value"] Selector</h2>  
<a href="https://www.cnnindonesia.com/">CNN Indonesia</a>  
<a target="_self" href="https://www.detik.com/">Detik.com</a>  
<a target="_blank" href="https://www.kompas.com/">Kompas.com</a>
```

2.2.4 CSS *Inline*

Atribut style digunakan untuk menambahkan gaya ke elemen, seperti warna, font, ukuran, dan lainnya. Pengaturan style pada suatu elemen, dapat dilakukan dengan atribut style. Atribut style HTML memiliki sintaks berikut:

```
<tagname style="property:value;">
```

Properti adalah properti CSS. *Value* adalah nilai CSS.

Contoh penggunaan *css inline*:

```
<h1 style="text-align:center;">Centered Heading</h1>  
<p style="text-align:center;">Centered paragraph.</p>  
  
<h1 style="font-size:300%;">This is a heading</h1>  
<p style="font-size:160%;">This is a paragraph.</p>  
  
<h1 style="font-family:verdana;">This is a heading</h1>  
<p style="font-family:courier;">This is a paragraph.</p>  
  
<h1 style="color:blue;">This is a heading</h1>  
<p style="color:red;">This is a paragraph.</p>  
  
<h1 style="background-color:blue;">This is a heading</h1>  
<p style="background-color:tomato;">This is a  
paragraph.</p>
```

2.2.5 CSS *Internal*

Kode CSS internal diletakkan di dalam bagian <head> pada halaman. *Class* dan *ID* bisa digunakan untuk merujuk pada kode CSS, namun hanya akan aktif pada halaman tersebut. Style CSS yang dipasang dengan metode ini akan di-download setiap kali halaman dipanggil, jadi ini akan meningkatkan kecepatan akses.

Namun, ada beberapa kasus dimana penggunaan internal *stylesheet* justru berguna. Salah satu contohnya adalah untuk mengirimkan template halaman ke seseorang – karena semuanya bisa terlihat dalam 1 halaman, maka akan lebih mudah untuk melihat previewnya. CSS internal diletakkan di dalam tag <style></style>. Contohnya:

```
<head>

<style>
    body {background-color: blue;}
    h1   {color: blue; font-family: verdana; font-family: courier; }

    p   {color: red;}
    .center   {text-align: "center";}

    #card {border: 2px solid blue; margin: 50px;}

    #card > h1 {color: tomato}
    #card > p {background-color: pink}

</style>

</head>

<body>

<h1>Centered Heading</h1>
<p>Centered paragraph.</p>

<h1 class="center">This is a heading</h1>
<p class="center">This is a paragraph.</p>

<div id="card">
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
</div>

</body>
```

2.2.6 CSS Eksternal

Salah satu cara yang paling nyaman untuk menambahkan CSS ke website adalah dengan menghubungkannya ke file .CSS eksternal. Dengan cara tersebut, perubahan apapun yang Anda buat pada file CSS akan tampil pada website Anda secara keseluruhan. File CSS eksternal biasanya diletakkan setelah bagian <head> pada halaman:

```
<head>

    <link rel="stylesheet" href="styles.css">

</head>
```

File styles.css

```
body {background-color: blue;}  
h1 {color: blue; font-family:verdana; font-family:courier; }  
  
p {color: red;}  
.center {text-align: "center";}  
  
#card {border: 2px solid blue; margin: 50px; }  
  
#card > h1 {color: tomato}  
#card > p {background-color: pink}
```

2.2.7 Component Properties

1. Comments

Komentar digunakan untuk menjelaskan kode, dan dapat membantu saat Anda mengedit kode sumber di kemudian hari. Komentar diabaikan oleh browser. Komentar CSS ditempatkan di dalam elemen <style>, dan dimulai dengan /* dan diakhiri dengan */:

```
/* Set text paragraph color to red */  
p {  
    color: red;  
}
```

2. Colors

Color ditentukan menggunakan nama warna yang telah ditentukan sebelumnya, atau nilai RGB, HEX, HSL, RGBA, HSLA. Di CSS, warna dapat ditentukan dengan menggunakan nama warna yang telah ditentukan:



Mengatur warna latar belakang untuk elemen HTML:

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>  
<p style="background-color: #4CAF50;">Lorem ipsum...</p>
```

Mengatur set color pada text:

```
<h1 style="color:Tomato;">Hello World</h1>  
<p style="color: #1e90ff;">Lorem ipsum...</p>  
<p style="color: rgb(30 255 203);">Ut wisi enim...</p>
```

Mengatur set color pada border:

```
<h1 style="border:1px dashed Tomato;">Hello World</h1>
<p style="border:2px solid #1e90ff;">Lorem ipsum...</p>
<p style="border:3px dotted rgb(30 255 203);">Ut wisi enim...</p>
```

3. Backgrounds

Property *background* CSS digunakan untuk menambahkan efek latar belakang untuk elemen. Dalam bagian ini, berikut ini beberapa property background CSS yang dapat digunakan:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position
- background

4. Borders

Property *border* CSS memungkinkan untuk menentukan style, lebar, dan warna pada setiap elemen. Property border-style menentukan jenis border apa yang akan ditampilkan.

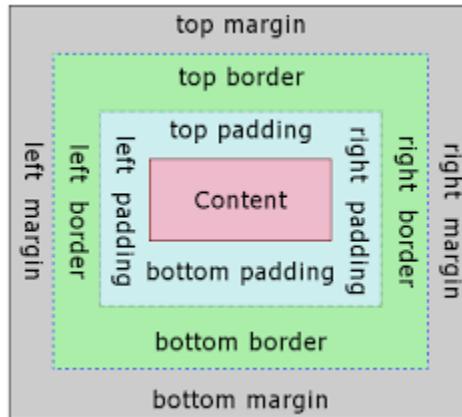
```
p{border-style: value;}
```

Berikut ini beberapa jenis **value** border yang dapat digunakan:

- dotted
- dashed
- solid
- double
- groove
- ridge
- inset
- outset
- none
- hidden

5. Margins dan Paddings

Margin dan padding merupakan properties CSS yang diperlukan untuk menentukan batas setiap element. Margin diperlukan untuk menentukan batas luar sedangkan padding diperlukan untuk batas dalam pada suatu elemen.



- Margin

Margin digunakan untuk membuat ruang di sekitar elemen, di luar batas yang ditentukan. Properti margin CSS digunakan untuk membuat ruang di sekitar elemen, batas diluar elemen. Dengan CSS, memiliki kontrol penuh atas margin. Ada properti untuk mengatur margin untuk setiap sisi elemen (top, right, bottom, dan left).

CSS memiliki properti untuk menentukan margin untuk setiap sisi elemen:

- margin-top
- margin-right
- margin-bottom
- margin-left
- margin

Semua properti margin dapat memiliki nilai sebagai berikut:

- auto ↗ browser menghitung margin
- length ↗ menentukan margin dalam px, pt, cm, dll.
- % ↗ menentukan margin dalam % dari lebar elemen yang mengandung
- inherit ↗ menentukan bahwa margin harus diwarisi dari elemen induk

Contoh

```
p {
    margin-top: 100px;
    margin-bottom: 100px;
    margin-right: 150px;
    margin-left: 80px;
}

h1{ margin: 25px 50px 75px 100px; }
```

- Paddings

Jika margin diperlukan untuk menentukan batas luar pada elemen, berbeda dengan padding. Padding diperlukan untuk menentukan batas dalam pada element.

Properti padding CSS digunakan untuk menghasilkan ruang di sekitar konten elemen, di dalam batas yang ditentukan. Dengan CSS, memiliki kontrol penuh

atas padding. Ada properti untuk mengatur padding untuk setiap sisi elemen (atas, kanan, bawah, dan kiri):

- padding-top
- padding-right
- padding-bottom
- padding-left
- padding

Semua properti margin dapat memiliki nilai sebagai berikut:

- length ↗ menentukan margin dalam px, pt, cm, dll.
- % ↗ menentukan margin dalam % dari lebar elemen yang mengandung
- inherit ↗ menentukan bahwa margin harus diwarisi dari elemen induk

Contoh:

```
div {  
    padding-top: 50px;  
    padding-right: 30px;  
    padding-bottom: 50px;  
    padding-left: 80px;  
}  
  
.card {  
    padding: 25px 50px 75px 100px;  
}
```

6. Height dan Width

Properti Height dan Width digunakan untuk mengatur tinggi dan lebar suatu elemen. Properti Height dan Width tidak termasuk padding, border, atau margin. Ini mengatur tinggi/lebar area di dalam padding, border, dan margin elemen.

Properti height dan width mungkin memiliki nilai berikut:

- auto ↗ Ini adalah default. Browser menghitung tinggi dan lebar
- length ↗ Mendefinisikan tinggi/lebar dalam px, cm dll.
- % ↗ Mendefinisikan tinggi/lebar dalam persen dari blok yang berisi
- initial ↗ Mengatur tinggi/lebar ke nilai default
- inherit ↗ Tinggi/lebar akan diwarisi dari nilai induknya

Contoh:

```
div {  
    height: 200px;  
    width: 50%;  
    background-color: powderblue;  
}
```

7. Text

- Text alignment

Properti text-align digunakan untuk mengatur perataan *horizontal* teks.

Sebuah teks dapat diratakan kiri atau kanan, di tengah, atau diratakan.

Contoh berikut menunjukkan rata tengah, dan teks rata kiri dan kanan

(perataan kiri adalah default jika arah teks dari kiri ke kanan, dan perataan kanan adalah default jika arah teks dari kanan ke kiri):

```
h1 {text-align: center; }  
h2 {text-align: left; }  
h3 {text-align: right; }  
  
p {text-align: justify; }
```

- Text Decoration

Properti text-decoration digunakan untuk mengatur atau menghapus dekorasi dari teks. Nilai text-decoration: none; sering digunakan untuk menghapus garis bawah dari tautan:

```
a {text-decoration: none; }
```

Contoh:

```
h2 {text-decoration: overline; }  
h3 {text-decoration: line-through; }  
h4 {text-decoration: underline; }
```

- Text Transformation

Properti text-transform digunakan untuk menentukan huruf besar dan huruf kecil dalam sebuah teks. Ini dapat digunakan untuk mengubah semuanya menjadi huruf besar atau kecil, atau menggunakan huruf besar untuk huruf pertama setiap kata:

```
.title {text-transform: uppercase; }  
span {text-transform: lowercase; }  
p {text-transform: capitalize; }
```

- Text Spacing

Text spacing merupakan properti yang diperlukan untuk memberikan jarak tertentu pada suatu text word. Berikut ini ada beberapa properti yang dapat digunakan untuk menentukan text spacing:

```
p {text-indent: 50px; }  
  
h1 {letter-spacing: 5px; }  
h2 {letter-spacing: -2px; }  
  
p.big {line-height: 1.8; }  
  
.two {word-spacing: -2px; }  
  
p {white-space: nowrap; }
```

- Text Shadow

Property text-shadow menambahkan bayangan ke teks. Dalam penggunaannya yang paling sederhana, Anda hanya menentukan bayangan

horizontal (2px), bayangan vertikal (2px), efek blur (5px) dan warna red untuk bayangannya:

```
h1 {text-shadow: 2px 2px 5px red; }
```

8. Font

Memilih font yang tepat memiliki dampak besar pada bagaimana pembaca mengalami situs web. Font yang tepat dapat menciptakan identitas yang kuat untuk merek Anda. Menggunakan font yang mudah dibaca adalah penting. Font menambah nilai pada teks Anda. Penting juga untuk memilih warna dan ukuran teks yang benar untuk font.

Property	Description
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-size	Specifies the font size of text
font-style	Specifies the font style for text
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-weight	Specifies the weight of a font

Contoh

```
.p1 {  
    font-family: "Times New Roman", Times, serif;  
    font-size: 40px;  
}  
  
.normal {  
    font-style: normal;  
    font-variant: normal;  
  
    font-weight: normal;  
  
    font-style: normal;  
}  
  
.italic {font-style: italic;}  
.oblique {font-style: oblique;}  
  
.thick {font-weight: bold;}  
  
.small {font-variant: small-caps;}
```

9. List

Dalam HTML, ada dua jenis daftar utama:

- List unordered ()  item daftar ditandai dengan peluru atau bulat
- List ordered ()  item daftar ditandai dengan angka atau huruf

Properti daftar CSS memungkinkan Anda untuk:

- Tetapkan penanda item daftar yang berbeda untuk daftar yang dipesan
- Tetapkan penanda item daftar yang berbeda untuk daftar yang tidak berurutan
- Tetapkan gambar sebagai penanda item daftar
- Tambahkan warna latar belakang ke daftar dan daftar item

Property list-style-type menentukan tipe penanda item daftar. Contoh berikut menunjukkan beberapa penanda item daftar yang tersedia:

```
ul.a {list-style-type: circle;}  
ul.b {list-style-type: square;}  
ol.c {list-style-type: upper-roman;}  
ol.d {list-style-type: lower-alpha;}
```

10. Display

Properti display menentukan perilaku tampilan (tipe kotak rendering) dari suatu elemen. Dalam HTML, nilai properti display default value diambil dari spesifikasi HTML atau dari style.css default browser/pengguna. Nilai default dalam XML adalah inline, termasuk elemen SVG.

```
p.ex1 {display: none;}  
p.ex2 {display: inline;}  
p.ex3 {display: block;}  
p.ex4 {display: inline-block;}
```

11. Float

Properti float digunakan untuk memposisikan dan memformat konten, misalkan, biarkan gambar melayang ke kiri ke teks dalam wadah. Properti float dapat memiliki salah satu dari nilai berikut:

- left  Elemen mengapung di sebelah kiri wadahnya
- right  Elemen mengapung di sebelah kanan wadahnya
- none  Elemen tidak mengapung (akan ditampilkan tepat di tempat kemunculannya dalam teks). Ini adalah default
- inherit  Elemen mewarisi nilai float dari induknya

Dalam penggunaannya yang paling sederhana, properti float dapat digunakan untuk membungkus teks di sekitar gambar.

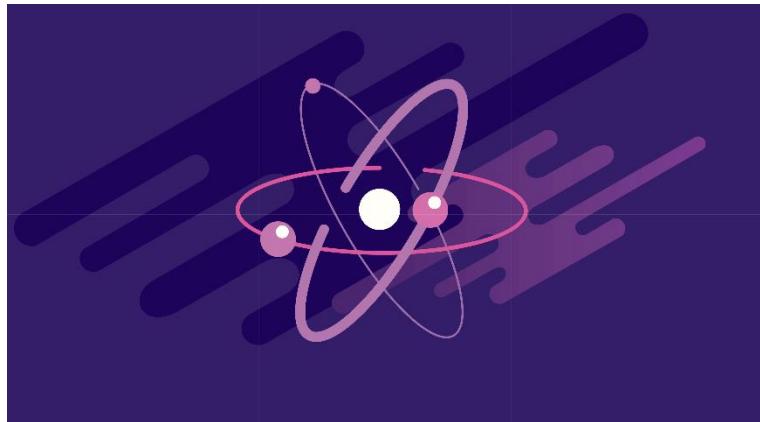
```
img {float: right;}
```

2.3 Latihan Pembelajaran

1. Buatlah file dokumen HTML bernama Latihan-1, ubahlah contoh dibawah ini kedalam scripting HTML dan menggunakan CSS sebagai variasi style pada elemen HTML:

Mengenal Partikel dan Notasi Atom

Nama anda, 20 Feb 2022 – 08:55 am



Perhatikan sekeliling kalian, matahari terbit dari timur di pagi hari, bulan muncul pada malam hari, bumi mengelilingi matahari dalam dua belas bulan, dan banyak lagi keteraturan di alam semesta ini. Hebat ya *Sang Pencipta* kita mengatur alam semesta ini dengan rapi. Bahkan, sampai tingkat paling kecil pun, elektron-elektron di alam semesta ini telah diatur dengan rapi menurut bilangan kuantumnya! **Wow apa tuh bilangan kuantum?**

Elektron-elektron tersebar di sekeliling atom dengan teratur berdasarkan tingkat energinya. Nah, tingkat energi inilah yang digambarkan dengan bilangan kuantum. Artinya, dari bilangan kuantum, lokasi-lokasi penyebaran elektron dapat digambarkan. Sedetail itu loh *Sang Pencipta* kita mengaturnya. Bayangkan kalau elektron, penyusun segala sesuatu di alam semesta ini, tidak teratur. Alam semesta ini tidak stabil dong. Mana bisa kita hidup di dunia seperti itu. Keren kan?

Salah satu contoh atom di alam semesta ini adalah atom karbon. Atom karbon adalah penyusun dari berbagai benda yang sangat berguna. Mulai dari bensin, plastik, berlian, bahkan tubuh kita pun tersusun dari karbon! Nah, karbon (*biasa dilambangkan dengan huruf C*) punya 6 elektron. Bagaimana bilangan kuantum dari elektron terakhirnya? Tinggal ikuti deh langkah-langkahnya.

Partikel Dasar Penyusun Atom dan Lambang Atom

Partikel dasar penyusun atom ada tiga yaitu proton (p), neutron (n) dan elektron (e). Jadi, massa atom = (massa p + massa n) + massa e . Massa elektron jauh lebih kecil

dari pada massa proton dan massa neutron, maka massa elektron dapat diabaikan.

Dengan demikian: $\text{massa atom} = \text{massa } p + \text{massa } n$.

Partikel	Lambang	Massa(g)	Muatan	
			Satuan	Coulomb
proton	p	1.673×10^{-24}	+1	1.6×10^{-9}
neutron	n	1.673×10^{-24}	0	0
elektron	e	9.109×10^{-31}	-1	1.6×10^{-19}

Lambang Atom



X Simbol dari unsur.

a nomor atom merupakan jumlah proton. Saat netral (tidak bermuatan) akan sama dengan jumlah elektron.

b nomor massa melambangkan jumlah proton ditambah jumlah neutron atau disebut juga jumlah nukleon.

c Muatan/bilangan oksidasi (biloks) terdiri dari melepas elektron (positif) dan menangkap elektron atau bertambah (negatif).

Bagikan artikel ini

2. Buatlah file dokumen HTML bernama Latihan-2, ubahlah contoh dibawah ini kedalam scripting HTML dan menggunakan CSS sebagai variasi style pada elemen HTML:

Daftar Hewan Dilindungi



Orang Utan

Orang utan dicirikan oleh rambut di seluruh badannya yang berwarna kemerahan. Satwa ini merupakan mamalia arboreal terbesar yang menghabiskan hampir seluruh waktunya di pepohonan. Lengannya yang panjang dan kuat serta tangan dan kakinya yang dapat mencengkeram erat.



Kera Emas

Monyet yang unik ini juga dikenal sebagai hewan yang memiliki adaptasi yang sangat baik dengan lingkungan yang bersuhu sangat dingin. Umumnya, monyet hidung pesek rambut emas tinggal di hutan pegunungan.

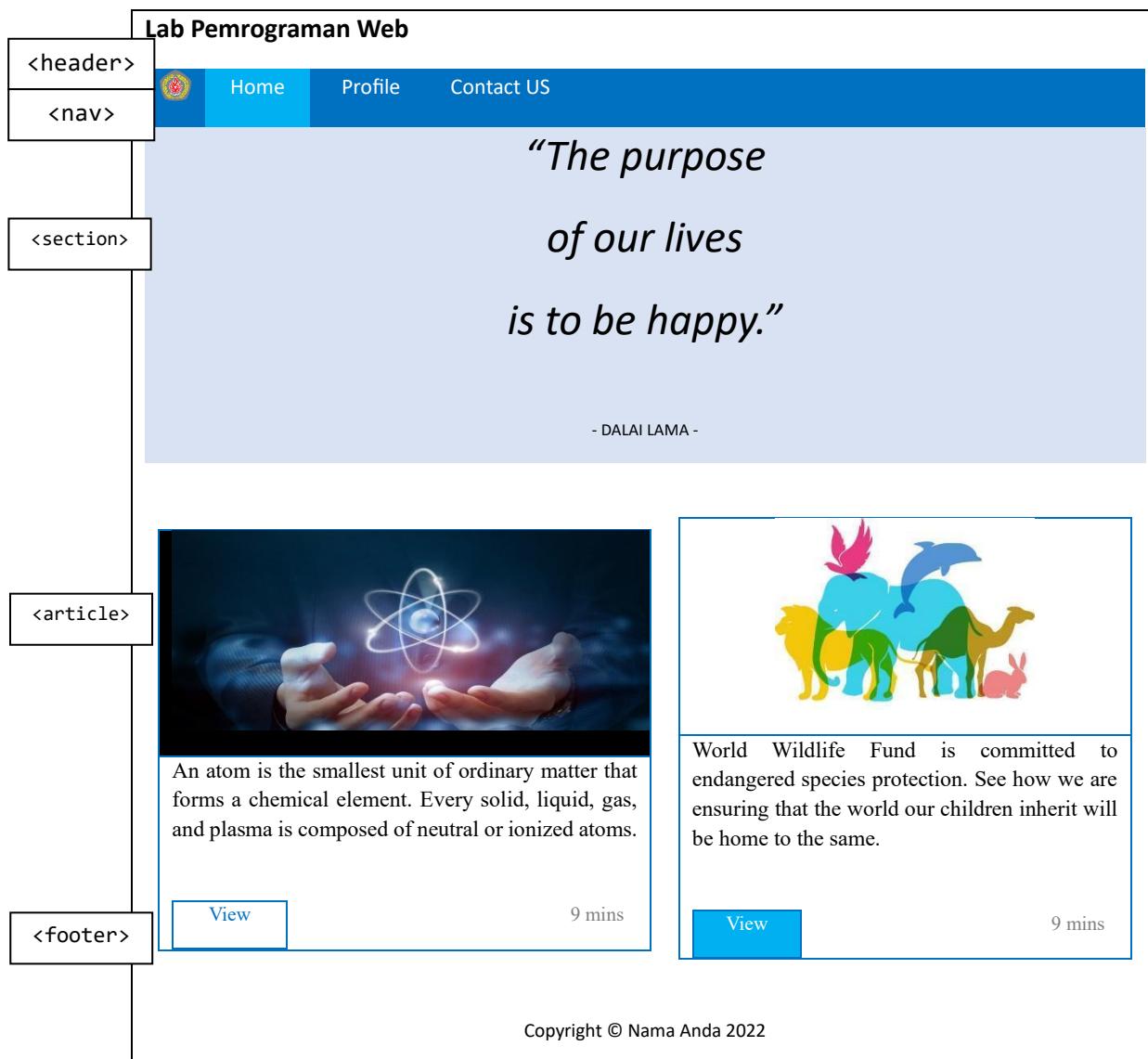


Burung Cendrawasih

Burung cenderawasih termasuk ke dalam hewan langka dan hanya bisa ditemukan di Papua. Burung cendrawasih punya tampilan yang cantik yang membuat burung ini sering menjadi perburuan liar untuk didagangkan.

Copyright © Nama Anda 2022

3. Buatlah file dokumen HTML bernama Latihan-3, ubahlah contoh dibawah ini kedalam scripting HTML dan menggunakan CSS sebagai variasi style pada elemen HTML:



Pada storyboard diatas untuk elemen `<article>` terdapat dua buah artikel dimana masing-masing artikel tersebut akan me-redirect ke halaman `Latihan-1.html` dan `Latihan-2.html` ketika mengklik tombol `view`. Jika tombol `view` disorot maka menampilkan background color.

4. Buatlah file dokumen HTML bernama Latihan-4, buatlah satu halaman dokumen HTML berisi data portofolio anda dan gunakan CSS sebagai style untuk membentuk Layout. Sambungkan halaman tersebut dengan Latihan-3 ketika mengklik tombol Profile pada elemen `<nav>`

BAB 3

PENGENALAN BAHASA PEMROGRAMAN JAVASCRIPT

3.1 Tujuan Pembelajaran

Mahasiswa memahami simbol atau karakter yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi.

3.2 Dasar Teori

3.2.1 Pengantar

Javascript diperkenalkan pertama kali oleh Netscape pada tahun 1995. Pada awalnya bahasa ini dinamakan “LiveScript” yang berfungsi sebagai bahasa sederhana untuk browser Netscape Navigator 2. Pada masa itu bahasa ini banyak dikritik karena kurang aman, pengembangannya yang terkesan buru buru dan tidak ada pesan kesalahan yang ditampilkan setiap kali kita membuat kesalahan pada saat menyusun suatu program. Kemudian sejalan dengan sedang giatnya kerjasama antara Netscape dan Sun (pengembang bahasa pemrograman “Java”) pada masa itu, maka Netscape memberikan nama “JavaScript” kepada bahasa tersebut pada tanggal 4 desember 1995. Pada saat yang bersamaan Microsoft sendiri mencoba untuk mengadaptasikan teknologi ini yang mereka sebut sebagai “Jscript” di browser Internet Explorer 3.

Javascript adalah bahasa yang berbentuk kumpulan skrip yang pada fungsinya berjalan pada suatu dokumen HTML, sepanjang sejarah internet bahasa ini adalah bahasa skrip pertama untuk web. Bahasa ini adalah bahasa pemrograman untuk memberikan kemampuan tambahan terhadap bahasa HTML dengan mengizinkan pengeksekusian perintah perintah di sisi user, yang artinya di sisi browser bukan di sisi server web.

Javascript bergantung kepada browser(navigator) yang memanggil halaman web yang berisi skrip skrip dari Javascript dan tentu saja terslip di dalam dokumen HTML. Javascript juga tidak memerlukan kompilator atau penterjemah khusus untuk menjalankannya (pada kenyataannya kompilator Javascript sendiri sudah termasuk di dalam browser tersebut). Lain halnya dengan bahasa “Java” (dengan mana JavaScript selalu dibanding bandingkan) yang memerlukan kompilator khusus untuk menerjemahkannya di sisi user/klien. Untuk mempelajari pemrograman Java Script, ada dua piranti yang diperlukan, yaitu :

- Teks Editor

Digunakan untuk menuliskan kode-kode Java Script, teks editor yang dapat digunakan antara lain notepad dan ultraedit.

- Web Browser

Digunakan untuk menampilkan halaman web yang mengandung kode-kode Java Script. Web browser yang digunakan harus mendukung Javascript. Browser yang dapat digunakan adalah internet explorer dan Netscape Navigator.

3.2.2 Penulisan Javascript

Kode Java Script dituliskan pada file HTML. Terdapat dua cara untuk menuliskan kode-kode Java Script agar dapat ditampilkan pada halaman HTML, yaitu :

- a) Java script ditulis pada file yang sama

Untuk penulisan dengan cara ini, perintah yang digunakan adalah

`<SCRIPT LANGUAGE = "JavaScript" >program java script disini</SCRIPT>`. Perintah tersebut biasanya diletakkan diantara Tag `<BODY>...</BODY>` Contoh Penulisan :

```
<HTML>
<HEAD>
<TITLE>.....</TITLE>
<SCRIPT LANGUAGE="Javascript">
    kode javascript disini
</SCRIPT>
</HEAD>
<BODY>
    kode HTML disini
</BODY>
</HTML>
```

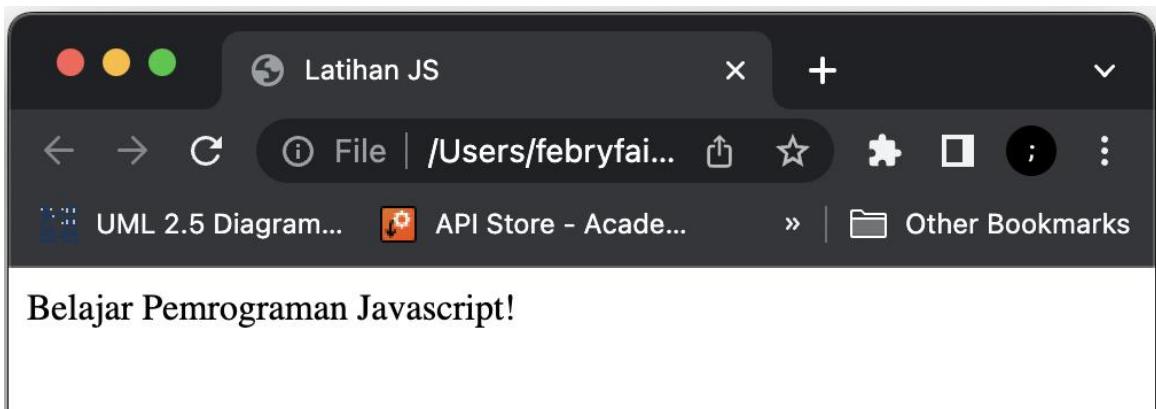
- b) Javascript ditulis pada file terpisah

Kode Javascript bisa juga kita buat dalam file terpisah dengan tujuan agar dokumen HTML isinya tidak terlalu panjang. Atribut yang digunakan adalah

```
<SCRIPT SRC= "namafile.js">...</SCRIPT>
```

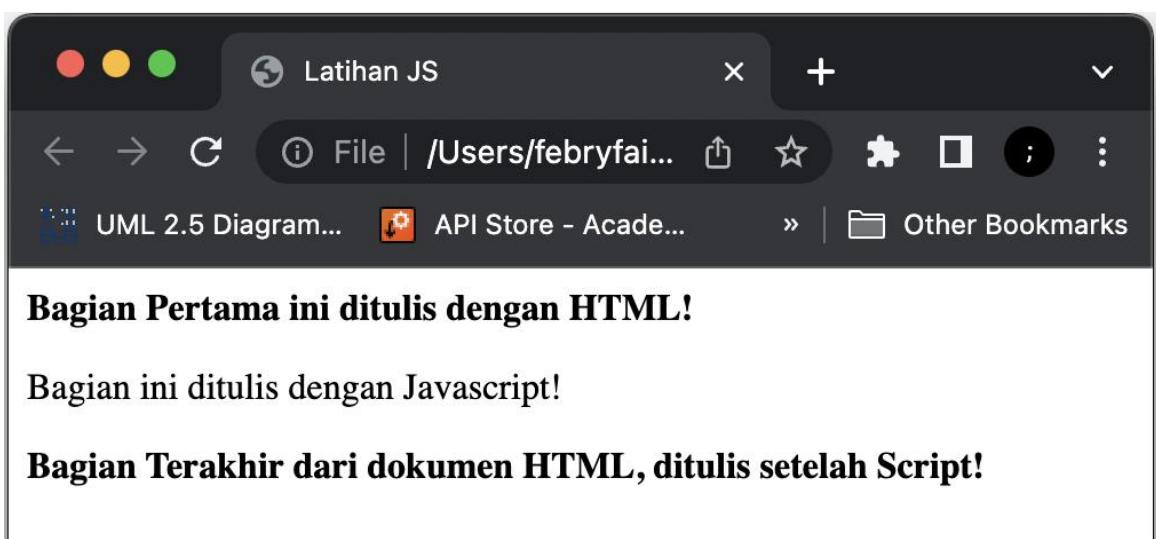
Diantara tag `<SCRIPT.....>` dan `<SCRIPT>` tidak diperlukan lagi kode Javascriptnya karena sudah dibuat dalam file terpisah. File yang mengandung kode Javascript berekstensi `.js` Pada bagian ini kita akan membuat program dengan menggunakan Javascript. Program ini akan menampilkan teks “ Belajar Pemrograman Javascript”

```
<HTML>
<HEAD>
<SCRIPT language="JavaScript">
<!--
document.write("Belajar Pemrograman Javascript!");
//-->
</SCRIPT>
</HEAD>
<BODY></BODY>
</HTML>
```



Berikut diberikan beberapa contoh program sederhana dengan menggunakan Javascript

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<B>Bagian Pertama ini ditulis dengan HTML!</B>
<P>
<SCRIPT language="JavaScript">
<!--
document.write("Bagian ini ditulis dengan Javascript!");
//-->
</SCRIPT>
<P>
<B>Bagian Terakhir dari dokumen HTML, ditulis setelah Script!</B>
</BODY>
</HTML>
```

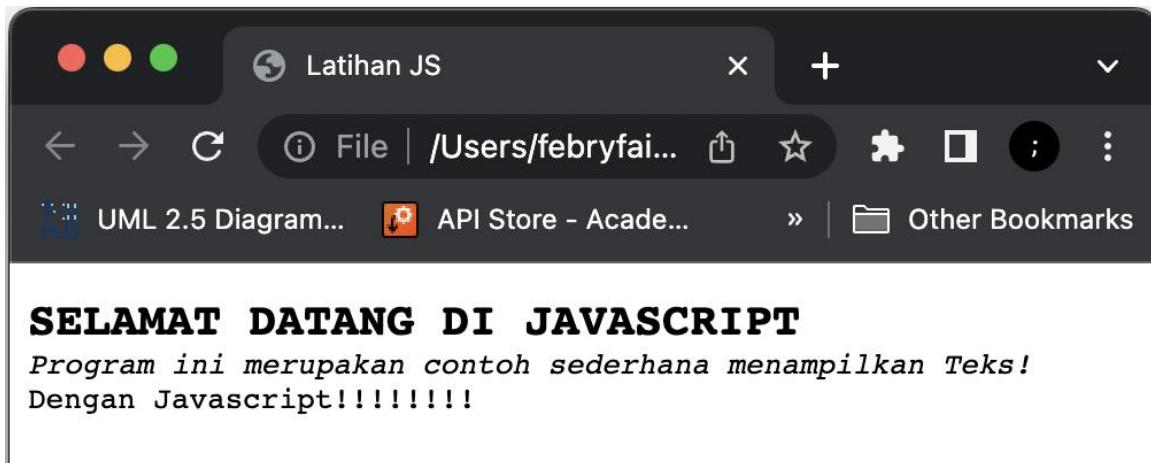


```
<HTML>
<HEAD><TITLE>BELajar Javascript Yuuuuuu</TITLE>
</HEAD>
<BODY BGCOLOR="gray">
<SCRIPT language="JavaScript">
```

```

<!--
document.writeln("<PRE>");
document.write("<B><FONT SIZE=5>");
document.writeln("SELAMAT DATANG DI JAVASCRIPT");
document.write("</B></FONT>"); document.write("<I>");
document.writeln("Program ini merupakan contoh sederhana menampilkan Teks!");
document.write("</I>");
document.writeln("Dengan Javascript!!!!!!!");
//-->
</SCRIPT>
</BODY>
</HTML>

```



Objek `document` mempunyai dua metode untuk menuliskan teks, yaitu **write** dan **writeln**. Metode **write** digunakan untuk menuliskan teks tanpa ganti baris, sedangkan metode **writeln** digunakan untuk menuliskan teks dengan ganti baris.

Berikut diberikan contoh Program Javascript yang diletakkan di file lain dan dipanggil melalui suatu file HTML

File external.js

```

document.writeln("<CENTER>");
document.writeln("<HR WIDTH=600 COLOR=Black>");
document.writeln("<H1>INSTITUTE BISNIS DAN INFORMATIK KESATUAN BOGOR</H1>");
document.writeln("<H2>FAKULTAS INFORMATIKA DAN PARIWISATA<H2>"); 
document.writeln("<H3>JURUSAN TEKNOLOGI INFORMASI<H3>"); 
document.writeln("<HR WIDTH=600 COLOR=Black>"); 
document.writeln("</CENTER>"); 

```

File

HTML

```

<HTML>
<HEAD>
<SCRIPT language="JavaScript" SRC=".//external.js"></SCRIPT>
</HEAD>
<BODY>
<B>Contoh ini menggunakan Javascript yang diambil dari File lain..</B>
</BODY>
</HTML>

```



Sama seperti bahasa pemrograman lain. Javascript juga menyediakan fasilitas untuk menuliskan komentar, komentar ini berguna bila nantinya anda atau orang lain membaca program. Pemberian komentar dalam Javascript dapat dilakukan dengan dua cara yaitu dengan menuliskan komentar setelah tanda garis miring dua kali, contoh :

//ini komentar

atau

/*ini juga komentar */

3.2.3 *Variables*

Variabel adalah tempat dimana kita menyimpan nilai-nilai atau informasi-informasi pada JavaScript. Variabel yang dideklarasikan dapat diisi dengan nilai apa saja.

Dalam JavaScript pendeklarasian sebuah variabel sifatnya opsional, artinya anda boleh mendeklarasikan atau tidak hal tersebut tidak menjadi masalah. Jika anda memberi nilai pada variabel, maka dalam JavaScript dianggap bahwa anda telah mendeklarasikan variabel tersebut.

Aturan penamaan variabel :

- Harus diawali dengan karakter (huruf atau baris bawah)
- Tidak boleh menggunakan spasi
- Huruf Kapital dan kecil memiliki arti yang berbeda
- Tidak boleh menggunakan kata-kata yang merupakan perintah dalam JavaScript.

Deklarasi Variabel

Var nama_variabel = nilai

Atau

Nama_variabel = nilai

Contoh :

```
var nama;           Nama = "Bunga Lestari"  
var nama = "Zaskia Mecca"    X = 1990;  
var X = 1998; Y = 08170223513 var Y;
```

3.2.4 Tipe Data

Tidak seperti bahasa pemrograman lainnya, JavaScript tidak memiliki tipe data secara eksplisit. Hal ini dapat dilihat dari beberapa contoh variabel diatas. Anda mendeklarasikan variabel tapi tidak menentukan tipenya. Meskipun JavaScript tidak memiliki tipe data secara eksplisit. JavaScript mempunyai tipe data implisit. Terdapat empat macam tipe data implisit yang dimiliki oleh JavaScript yaitu:

- Numerik, seperti : 0222532531, 1000, 45, 3.146789 dsb
- String, seperti : "Hallo", "April", "Jl. Setiabudi No 17A", "Cece Kirani" dsb
- Boolean, bernilai true atau false
- Null, variabel yang tidak diinisialisasi

3.2.5 Tipe Numerik

Pada dasarnya JavaScript hanya mengenal dua macam tipe numerik, yaitu bilangan bulat (integer) dan bilangan pecahan(real/float).

Untuk bilangan bulat, kita dapat merepresentasikan dengan basis desimal, oktal atau heksadesimal.

Contoh :

```
var A = 100;
```

```
var B = 0x2F;
```

untuk pendeklarasian tipe bilangan real, dapat menggunakan tanda titik atau notasi ilmiah (notasi E).

Contoh :

```
var a = 3.14533567;
```

```
var b = 1.23456E+3;
```

3.2.6 Tipe String

Untuk mendeklarasikan tipe string dapat dilakukan dengan cara menuliskan string di antara tanda petik tunggal ('') atau tanda petik ganda ("")

```
Contoh : var str ='Contoh deklarasi string';
var str1 = "cara ini juga bisa untuk menulis string";
```

3.2.7 *Tipe Boolean*

Tipe boolean hanya mempunyai nilai True atau False. Tipe ini biasanya digunakan untuk mengecek suatu kondisi atau keadaan.

Contoh :

```
var X = (Y > 90); contoh diatas menunjukkan bahwa jika Y lebih besar dari 90 maka X akan bernilai True.
```

3.2.8 *Tipe Null*

Tipe Null digunakan untuk merepresentasikan variabel yang tidak diberi nilai awal (inisialisasi).

3.2.9 *Operator*

Operator pada JavaScript terbagi menjadi enam, yaitu :

- Aritmatika
- Pemberian nilai (Assign)
- Pemanipulasi bit (bitwise)
- Pembanding
- Logika
- String

a. Operator Aritmatika

Digunakan untuk operan bertipe numerik. Ada dua macam operator aritmatik, yaitu operator numerik tunggal dan operator aritmatik biner. Perbedaan kedua operator terletak pada jumlah operan yang harus dioperasikan.

Operator	Tunggal/Biner	Keterangan
+	Biner	Penjumlahan
-	Biner	Pengurangan
*	Biner	Perkalian
/	Biner	Pembagian
%	Biner	Modulus
-	Tunggal	Negasi

++	Tunggal	Penambahan dengan satu
--	Tunggal	Pengurangan dengan satu

b. Operator Pemberian Nilai

Digunakan untuk memberikan nilai ke suatu operan atau mengubah nilai suatu operan.

Operator	keterangan	Contoh	Ekuivalen
=	Sama dengan	X=Y	
+=	Ditambah dengan	X+=Y	X=X+Y
-=	Dikurangi dengan	X-=Y	X=X-Y
=	Dikali dengan	X=Y	X=X*Y
/=	Dibagi dengan	X/=Y	X=X/Y
%=	Modulus dengan	X%=Y	X=X%Y
&=	Bit AND dengan	X&Y	X=X&Y
=	Bit OR	X =Y	X=X Y

c. Operator Manipulasi Bit

Operasi ini berhubungan dengan pemanipulasi bit pada operan bertipe bilangan bulat.

Operator	Keterangan
&	Bit AND
	Bit OR
^	Bit XOR
~	Bit NOT
<<	Geser ke kiri
>>	Geser ke Kanan
>>>	Geser ke kanan dengan diisi nol

Contoh : var A = 12; // A = 1100b var B = 10; // B = 1010b var C = A & B maka akan dihasilkan bilangan seperti berikut :

1100b

1010b AND

1000b

var A = 12; var C = A<<
2

var D = A >> 1

maka variabel C akan bernilai 48(0011 0000b)
variabel D akan bernilai 6 (0110b)

d. Operator Pembanding

Digunakan untuk membandingkan dua buah operand. Operan yang dikenal operator ini dapat bertipe string, numerik, maupun ekspresi lain.

Operator	Keterangan
==	Sama dengan
!=	Tidak sama dengan
>	Lebih besar
<	Lebih kecil
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan

e. Operator Logika

Digunakan untuk mengoperasikan operan yang bertipe boolean.

Operator	Keterangan
&&	Operator logika AND
	Operator Logika OR
!	Operator logika NOT

Contoh :

```
var A = true; var B = false;
var C = A && B; //menghasilkan false
```

```
var D = A || B ; // false
```

```
var E = !A; //false
```

f. Operator String

Selain operator pembanding, operator string pada JavaScript juga mengenal satu operator lagi yang bernama PENGGABUNGAN. Operator ini digunakan untuk menggabungkan beberapa string menjadi sebuah string yang lebih panjang.

Contoh : nama = "Java" + "Script";

akan menghasilkan "JavaScript" pada variabel nama

Contoh Program JavaScript

```
<HTML>
```

```
<HEAD><TITLE>Operasi Aritmatika</TITLE></HEAD>
```

```
<BODY>
```

```
<P><SCRIPT language="JavaScript">
```

```
<!--
```

```
document.writeln("<PRE>");
```

```
document.writeln("<H1>Operasi Aritmatik</H1>"); var A = "100"; var B =  
"200"; var C = 300; var D = 400; var E = A + B;
```

```
document.writeln("100" + "200" = ' + E); E =  
B + C;
```

```
document.writeln("200" + 300 = ' + E); E =  
C + D;
```

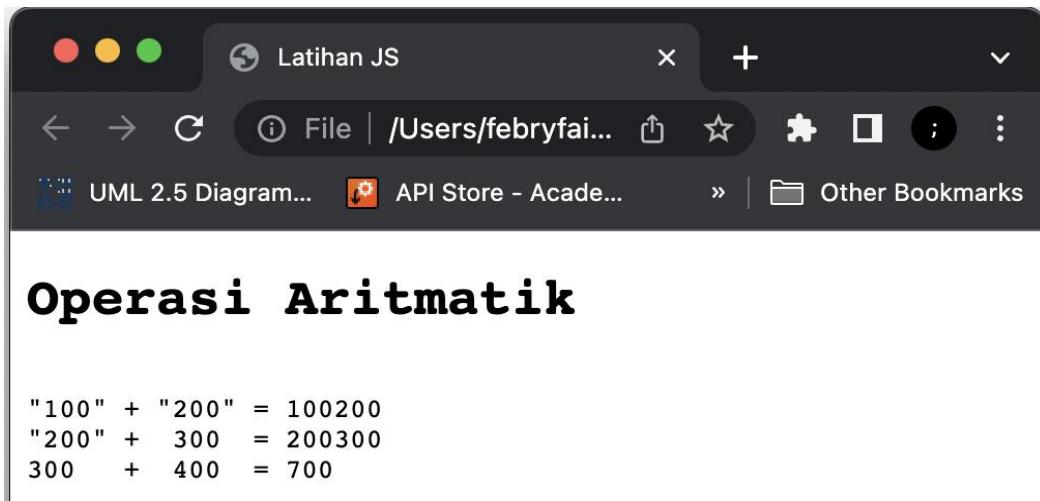
```
document.writeln('300 + 400 = ' + E); document.writeln("<PRE>");
```

```
/-->
```

```
</SCRIPT></P>
```

```
</BODY>
```

```
</HTML>
```



3.2.10 Masukan Data

Untuk memasukkan data dari keyboard dapat dilakukan dengan menggunakan perintah **input**.

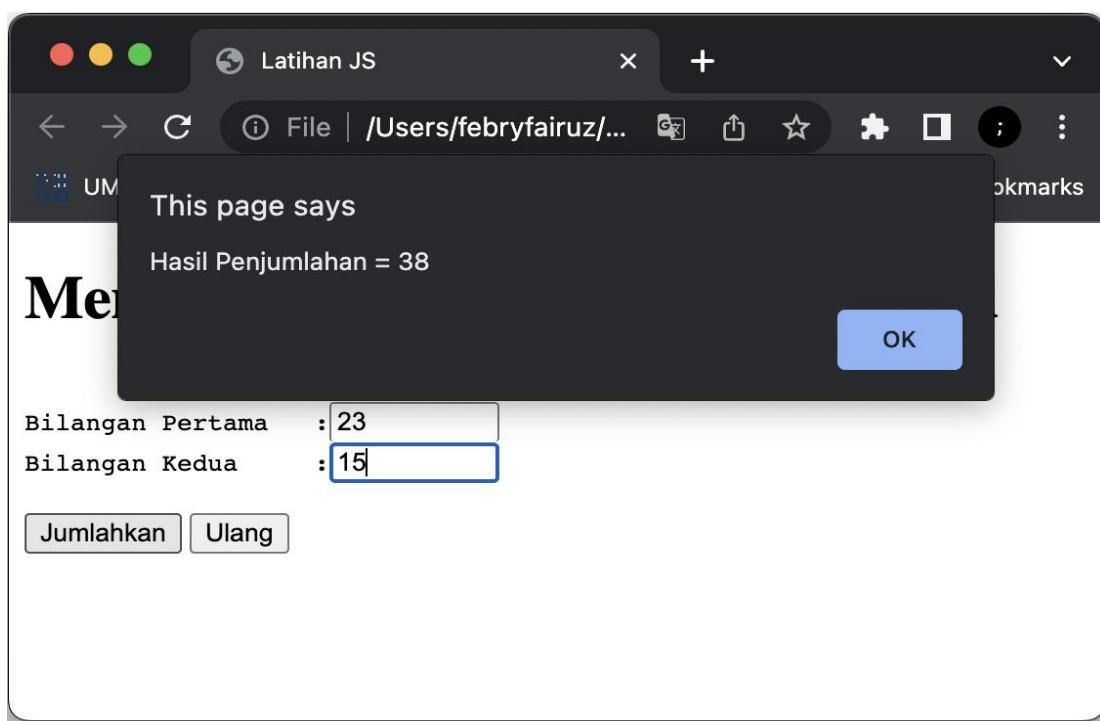
Contoh Program JavaScript

```
<HTML>
<HEAD><TITLE>Memasukkan Bilangan</TITLE></HEAD>
<BODY>
<P><SCRIPT language="JavaScript">
<!--
function jumlah()
{
    var bil1 = parseFloat(document.fform.bilangan1.value);    if
(isNaN (bil1)) bil1=0.0;
    var bil2 = parseFloat(document.fform.bilangan2.value);    if
(isNaN (bil2)) bil2=0.0;    var hasil = bil1 + bil2;
    alert ("Hasil Penjumlahan = " + hasil);
}
//--></SCRIPT></P>
<FORM NAME ="fform">
<H1><BR>Memasukkan Data Lewat Keyboard</H1>
<PRE>
```

```

Bilangan Pertama :<input type="text" size="11" name="bilangan1">
Bilangan Kedua  :<input type="text" size="11" name="bilangan2">
</PRE>
<P>
<INPUT TYPE="button" value="Jumlahkan" onclick="jumlah()">
<INPUT TYPE="reset" value="Ulang">
</FORM>
</BODY>
</HTML>

```



3.2.11 Object Javascript

- Objek Untuk memasukkan Data

Terdapat beberapa objek yang dapat digunakan untuk memasukkan data. Objek Objek tersebut biasanya terdapat dalam suatu form. Adapun objek-objek tersebut meliputi Objek Text, Objek Radio, Objek Checkbox, Objek Textarea, dan Objek Select.

- Objek Text

Untuk menginputkan data kita dapat menggunakan komponen/objek text. Contoh penggunaannya dapat kita lihat pada contoh berikut :

```

<html>
<head><title>Latihan Dengan Objek Text</title></head>
<body>
<script language ="JavaScript">
<!--
function tekan()
{
    var namastr = (document.form.nama.value);    var
alamatstr      =      (document.form.alamat.value);
document.form.onama.value      =      namastr;
document.form.oalamat.value = alamatstr;
}
//-->
</script>
<form name ="fform">
<H1> Memasukkan Data Dengan Objek Teks</H1><hr>
<PRE>
Nama Anda :<input type="text" size="11" name="nama">
Alamat :<input type="text" size="25" name="alamat">
</PRE>
<BR>
<input type="button" value="kirim" onclick="tekan()">
<input type="reset" value="ulang">
<H3>Output</H3>
<PRE>
Jadi Nama Anda adalah :<input type="text" size="11" name="onama">
Alamat Anda di :<input type="text" size="25" name="oalamat">
</form>
</body>
</html>

```

Latihan Dengan Objek Text

Nama Anda :

Alamat :

kirim **ulang**

Output

Jadi Nama Anda adalah :
 Alamat Anda di :

- Objek Radio

Objek radio adalah komponen yang digunakan untuk melakukan suatu pemilihan data. Karena selalu berupa Array , untuk mengakses satu tombol radio digunakan radio[indeks]. Disamping itu objek radio juga mempunyai nilai True jika dipilih dan False jika tidak. Untuk memilih suatu objek radio menggunakan properti Checked.

```
<html>
<head><title>Latihan Dengan Objek Radio</title></head>
<body>
<script language ="JavaScript">
<!--
function radio_box(form)
{
  var ket =
  "";
  if (form.wanita.checked == true)
  {
    ket = "Wanita";
  }
  else
  {
    ket = "Laki-laki";
  }
  document.write(ket);
}
-->
</script>
<form>
<input type="radio" name="wanita" checked="checked"/> Wanita
<input type="radio" name="wanita"/> Laki-laki
</form>

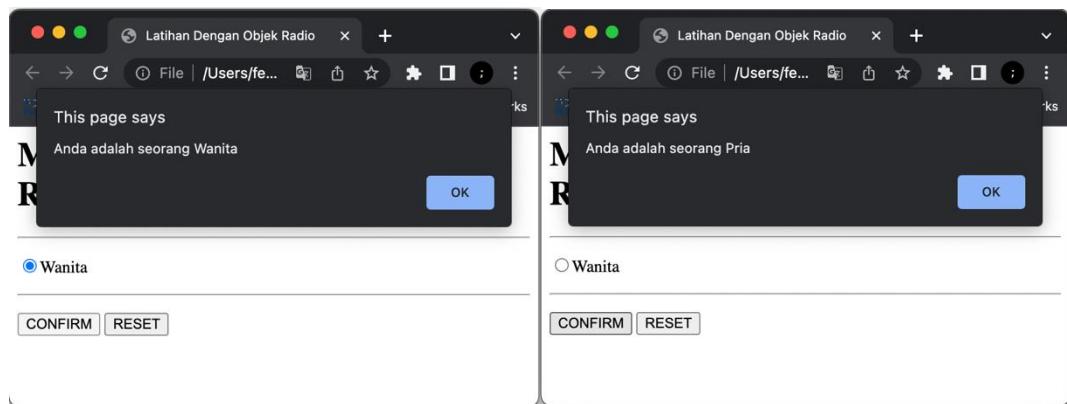
```

```

        }
    else
    {
        ket = "Pria";
    }
    alert('Anda adalah seorang ' +ket);
}
//-->
</script>
<form>

<H1> Memasukkan Data Dengan Objek Radio</H1><hr>
<p><input type="radio" value="wanita" name="wanita">Wanita</p>
<hr>
<p><input type="button" value="CONFIRM" onclick="radio_box(this.form)"> <input type="reset" value="RESET"></p>
</form>
</body>
</html>

```



- Objek Checkbox

Objek checkbox menyimpan informasi tentang elemen form yang berupa kotak cek. Penggunaannya hampir sama seperti objek radio.

```

<html>
<head><title>Latihan Dengan Objek Checkbox</title></head>

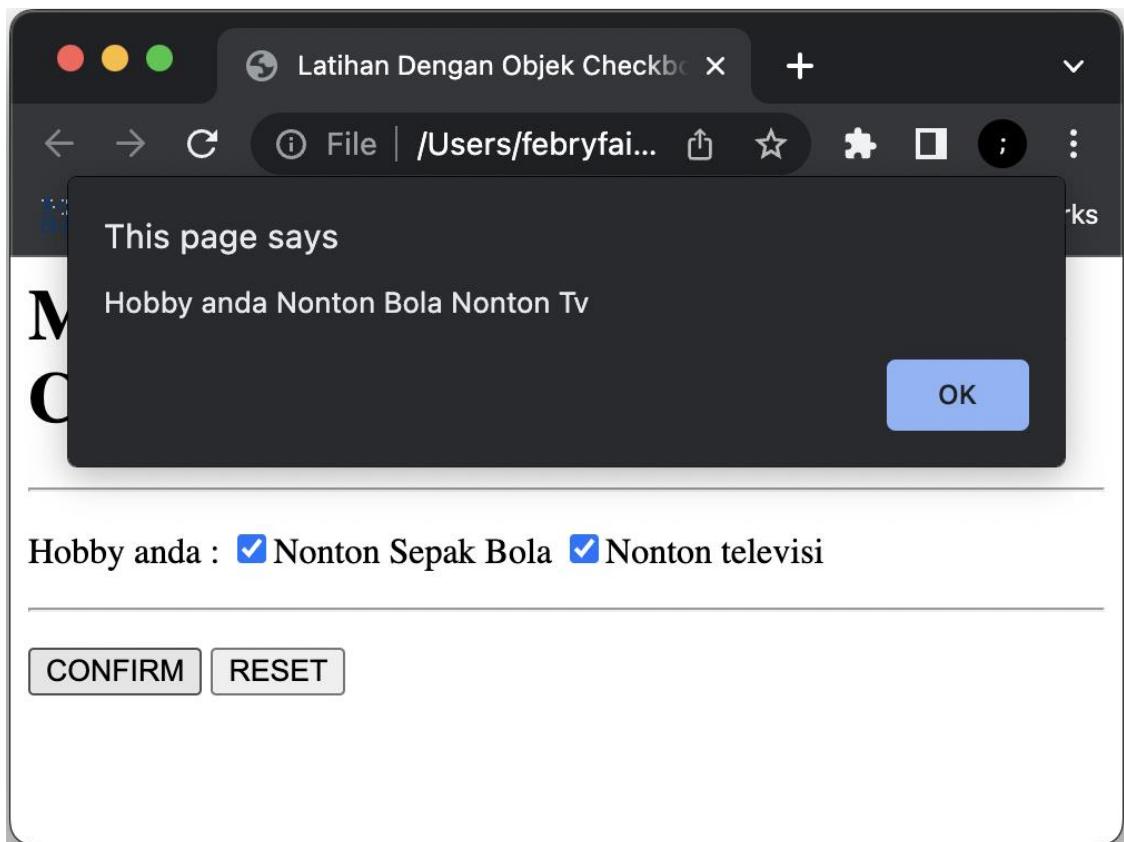
```

```

<body>
<script language ="JavaScript">
<!--
function radio_box(form)
{
  var ket =
""; var ket1
="";
if (form.bola.checked == true)
{
  ket = "Nonton Bola";
}

if (form.tv.checked == true)
{
  ket1 = " Nonton Tv";
}
alert('Hobby anda ' +ket+''+ket1);
}
//-->
</script>
<form>
<H1> Memasukkan Data Dengan Objek Checkbox</H1><hr>
<p>Hobby anda :
<input type="checkbox" value="ON" name="bola">Nonton Sepak Bola
<input type="checkbox" value="ON" name="tv">Nonton televisi</p>
<hr>
<p><input type="button"
value="CONFIRM"
onclick="radio_box(this.form)"> <input
type="reset" value="RESET"></p>
</form>
</body>
</html>

```

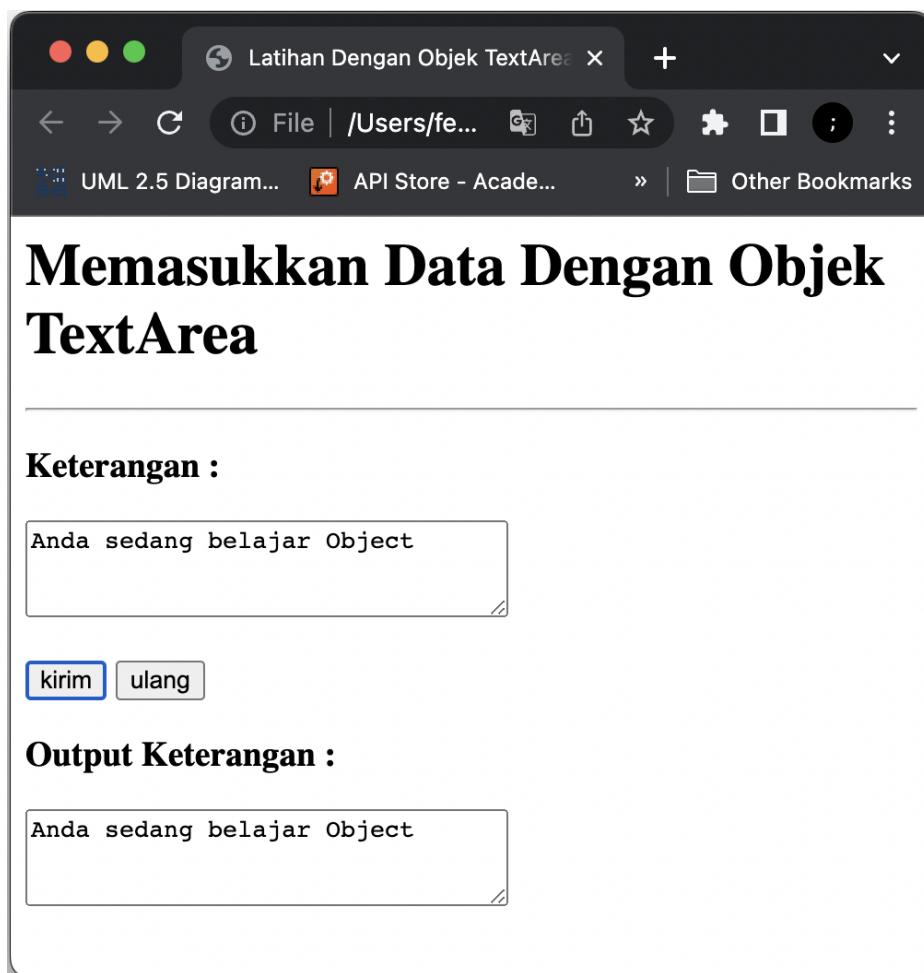


- Objek TextArea

Objek textarea menyimpan informasi tentang elemen form yang berupa kotak teks dengan banyak baris.

```
<html>
<head><title>Latihan Dengan Objek TextArea</title></head>
<body>
<script language ="JavaScript">
<!--
function tekan()
{
    var ketstr = (document.form.Ket.value); document.form.Oket.value = ketstr;
}
//-->
</script>
```

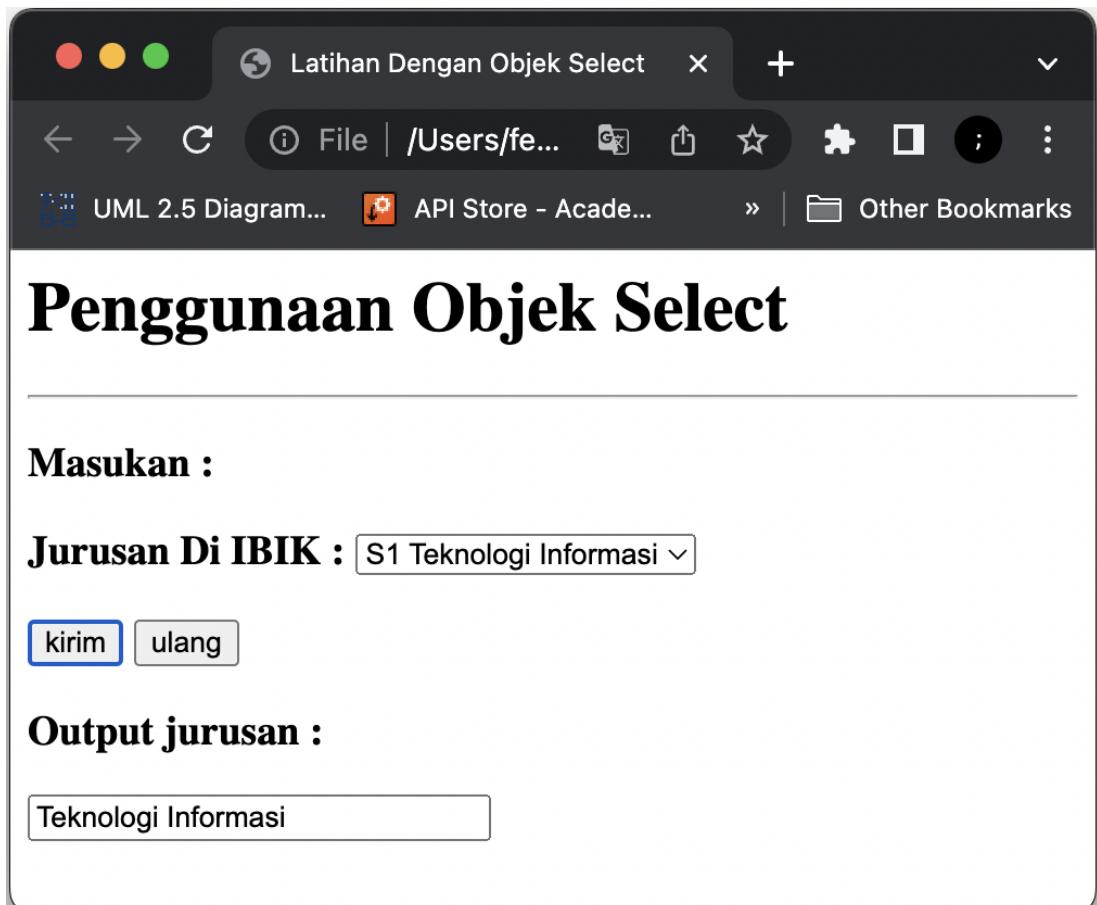
```
<form name ="fform">  
<H1> Memasukkan Data Dengan Objek TextArea</H1><hr>  
<h3>Keterangan :<h3><br>  
<textarea name="Ket" rows="3" cols="30"></textarea>  
<BR><BR>  
<input type="button" value="kirim" onclick="tekan()">  
<input type="reset" value="ulang">  
<H3>Output Keterangan :</H3>  
<textarea name="Oket" rows="3" cols="30"></textarea>  
</form>  
</body>  
</html>
```



- Objek Select

Objek Select menyimpan informasi tentang elemen form yang berupa kotak daftar. Objek select berguna apabila di dalam form terdapat banyak pilihan yang telah mempunyai nilai tertentu.

```
<html>
<head><title>Latihan Dengan Objek Select</title></head>
<body>
<script language ="JavaScript">
<!--
function tekan()
{
    var jurusanstr = (document.form.Jurusan.value); document.form.Ojurusan.value
= jurusanstr;
}
//-->
</script>
<form name ="fform">
<H1> Penggunaan Objek Select</H1><hr>
<h3>Masukan :<h3>
Jurusani Di IBIK :
<select name="Jurusani" Size="1">
<option value ="Teknologi Informasi"> S1 Teknologi Informasi </option>
<option value ="Pariwisata"> S1 Pariwisata</option>
<option value ="Teknik Industri"> Teknik Industri </option>
</select>
<p><input type="button" value="kirim" onclick="tekan()">
<input type="reset" value="ulang">
<H3>Output jurusan :</H3>
<input type="text" name="Ojurusan" size="30">
</form>
</body>
</html>
```



3.3 Kontrol Program

3.3.1 Percabangan

Untuk membuat suatu halaman yang dinamis dan interaktif, perancang halaman Web membutuhkan perintah-perintah yang dapat mengatur aliran dari informasi. Berdasarkan hasil komputasi yang telah dilakukan, JavaScript akan membuat keputusan jalur mana yang akan dieksekusi. Pada dasarnya dalam JavaScript terdapat dua macam pernyataan percabangan yaitu if..else dan switch

- **IF..ELSE**

Pernyataan ini digunakan untuk menguji sebuah kondisi dan kemudian mengeksekusi pernyataan tertentu bila kondisi tersebut terpenuhi, dan mengeksekusi pernyataan lain bila kondisi tersebut tidak terpenuhi.

```
if (kondisi) {  
    //pernyataan 1 dieksekusi  
    //bila kondisi terpenuhi  
} else {  
    //pernyataan2 dieksekusi
```

```
//bila kondisi tidak terpenuhi  
}
```

kondisi adalah ekspresi JavaScript yang mana hasil evaluasinya memiliki nilai Boolean **true** atau **false**

Untuk kasus yang melibatkan lebih banyak kondisi, maka kita dapat meletakkan pernyataan if lain setelah else

```
if (kondisi1) {  
    //pernyataan 1 dieksekusi  
    //bila kondisi1 terpenuhi  
} else if (kondisi2) {  
    //pernyataan2 dieksekusi  
    //bila kodisi1 tidak terpenuhi  
} else {  
    //pernyataan3 dieksekusi  
    //bila kodisi2 tidak terpenuhi  
}
```

Contoh program

```
<HTML>  
<HEAD><TITLE>Percabangan IF-ELSE</TITLE></HEAD>  
<BODY>  
<SCRIPT language="JavaScript">  
<!--  
function tanyabilangan(){  
    var bil = parseFloat(document.form.bilangan.value);  
    var jenis = " ";  
    if(isNaN(bil)) {  
        alert("Anda Belum memasukkan Bilangan");  
    } else {  
        if (bil > 0) {  
            jenis = " Adalah bilangan Positif";  
        } else {  
            jenis = " Adalah bilangan Negatif";  
        }  
    }  
    document.form.jenis.value = jenis;  
}</SCRIPT>  
</BODY>
```

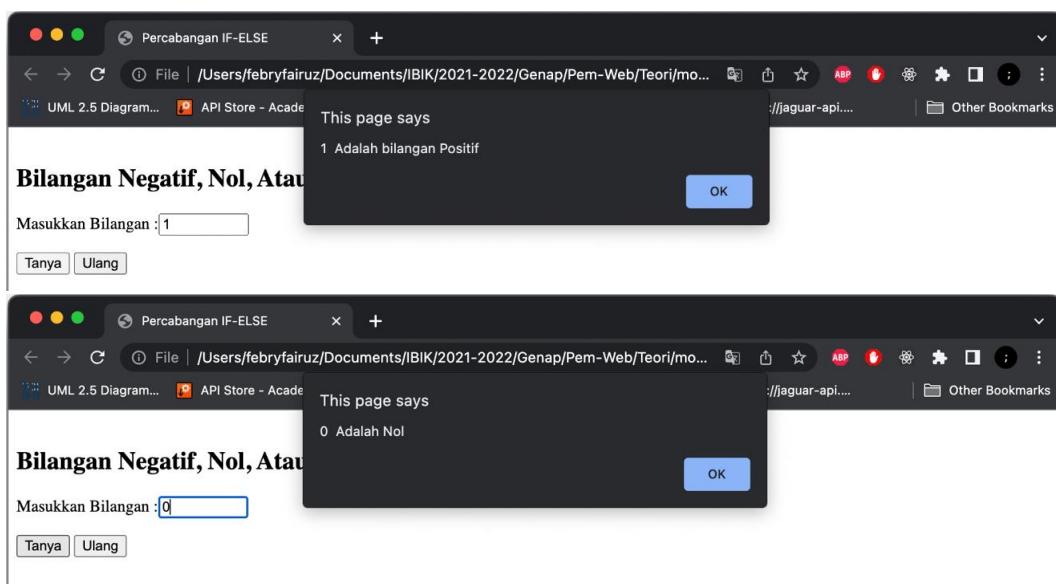
```

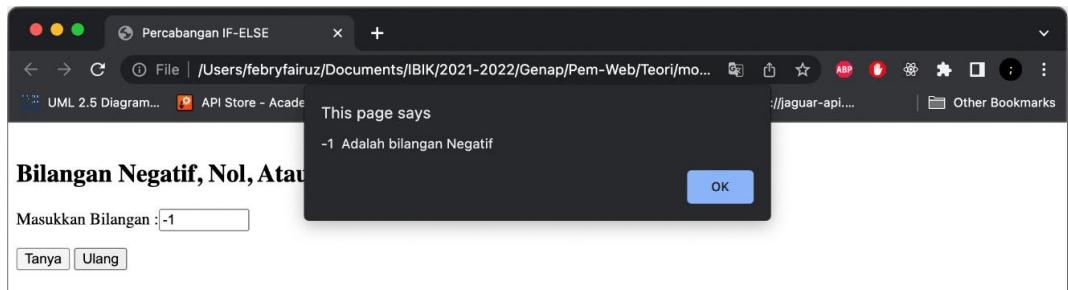
}else if (bil < 0) {
    jenis = " Adalah bilangan Negatif";
}else{
    jenis = " Adalah Nol";
}
alert (bil+" "+jenis);
}

//--></SCRIPT>

<FORM NAME ="fform">
<H2><BR>Bilangan Negatif, Nol, Atau Positif ???</H2>
Masukkan Bilangan :<input type="text" size="11" name="bilangan"> <P>
<INPUT TYPE="button" value="Tanya" onclick="tanyabilangan()">
<INPUT TYPE="reset" value="Ulang"> </p>
</FORM>
</BODY>
</HTML>

```





- **IF.. ELSE IF.. ELSE**

Percabangan majemuk adalah suatu percabangan yang dapat melibatkan lebih dari 1 kondisi di dalam percabangannya. Biasanya percabangan seperti ini menggunakan operator tambahan seperti AND, OR dan sebagainya.

<HTML>

```
<HEAD><TITLE>Percabangan IF-ELSE 3</TITLE></HEAD>
<BODY>
<SCRIPT language="JavaScript">
<!--
function hitungip(){
    var quis = parseFloat(document.fform.iquis.value);
    var tugas = parseFloat(document.fform.itugas.value);
    var uts = parseFloat(document.fform.iuts.value);
    var uas = parseFloat(document.fform.iuas.value);
    var ip =" "; var ket=" ";
    var na = (0.10*quis)+(0.20*tugas)+(0.30*uts)+(0.40*uas);

    if ((na > 80) && (na <=100)) {
        ip ="A";
        ket="Lulus dengan Sangat Baik";
    } else if ((na > 68) && (na <=80)) {
        ip ="B";
        ket="Lulus dengan Baik";
    }else if ((na > 55) && (na <=68)) {
        ip ="C";
        ket="Lulus dengan Cukup";
    } else if ((na > 38) && (na <=55)) {

```

```

ip ="D";
ket="Lulus dengan Kurang";
} else {
    ip ="E"; ket="Tidak Lulus";
}
document.form.oip.value=ip; document.form.oket.value=ket;
// gunakan untuk mengecek alert (ip""+na);
}

//-->

</SCRIPT>

<FORM NAME ="fform">

<table border="1" width="100%" ALIGN="center" >
<tr>

<td width="100%" colspan="4"><H2 ALIGN="center">Menghitung Indeks
Prestasi</H2></td>

</tr>
<tr>

<td width="25%">Quis (10%) :<input type="text" size="10" name="iquis">
</td>

<td width="25%">Tugas (20%):<input type="text" size="10"
name="itugas"> </td>

<td width="25%">UTS (30%):<input type="text" size="10" name="iuts">
</td>

<td width="25%">UAS (40%) :<input type="text" size="10" name="iuas">
</td>

</tr>
<tr>

<td width="100%" colspan="4"><P Align="center">

<INPUT TYPE="button" value="Hitung" onclick="hitungip()">
<INPUT TYPE="reset" value="Ulang"> </p></td>

</tr>
<tr>

```

```

<td width="100%" colspan="4" align="center">
Indeks Prestasi :<input type="text" size="5" name="oip">
Keterangan :<input type="text" size="25" name="oket"></td>
</tr>
</table>
</FORM>
</BODY>
</HTML>

```

Menghitung Indeks Prestasi

Quis (10%) :	Tugas (20%):	UTS (30%):	UAS (40%) :
70	80	75	88
<input type="button" value="Hitung"/>		<input type="button" value="Ulang"/>	
Indeks Prestasi :	A	Keterangan :	Lulus dengan Sangat Baik

- **SWITCH**

Selain menggunakan if..else, percabangan juga dapat ditangani dengan perintah switch. Dengan kata lain pernyataan switch digunakan untuk menyederhanakan pernyataan if..else yang terlalu banyak.

<HTML>

```

<HEAD><TITLE>Percabangan Switch</TITLE></HEAD>
<BODY>
<SCRIPT language="JavaScript">
<!--
function tanyabulan(){
var bulan = parseFloat(document.form.ibulan.value);
var namabulan=" ";
switch (bulan) {
case 1 : namabulan="Bulan ke 1 adalah = Januari";break;
case 2 : namabulan="Bulan ke 2 adalah = Februari";break;

```

```

case 3 : namabulan="Bulan ke 3 adalah = Maret";break;
case 4 : namabulan="Bulan ke 4 adalah = April";break;
case 5 : namabulan="Bulan ke 5 adalah = Mei";break;
case 6 : namabulan="Bulan ke 6 adalah = Juni";break;
case 7 : namabulan="Bulan ke 7 adalah = Juli";break;
case 8 : namabulan="Bulan ke 8 adalah = Agustus";break;
case 9 : namabulan="Bulan ke 9 adalah = September";break;
case 10 : namabulan="Bulan ke 10 adalah = Oktober";break;
case 11 : namabulan="Bulan ke 11 adalah = November";break;
case 12 : namabulan="Bulan ke 12 adalah = Desember";break;
default : namabulan="Anda salah mengisi";
}
alert(namabulan);
}
//--></SCRIPT>

<FORM NAME ="fform">

<H2>Penggunaan Percabangan Switch</H2><HR>

<PRE>

Masukkan Nomor Bulan [1-12] :<input type="text" size="2" name="ibulan">

<INPUT TYPE="button" value="Hitung" onclick="tanyabulan()"><INPUT
TYPE="reset" value="Ulang">

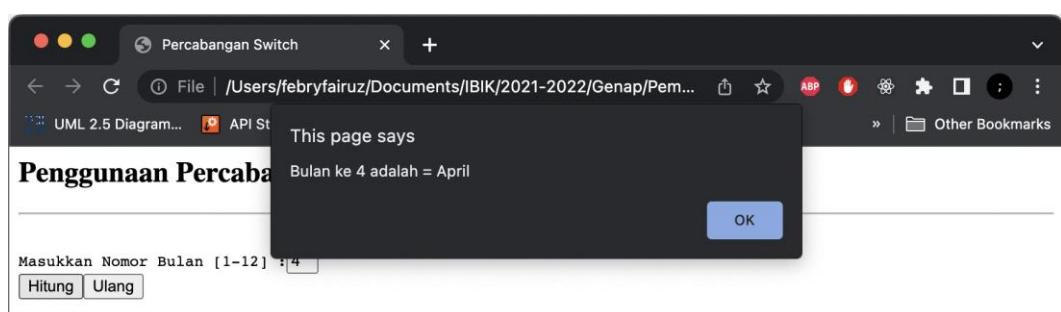
</PRE>

</FORM>

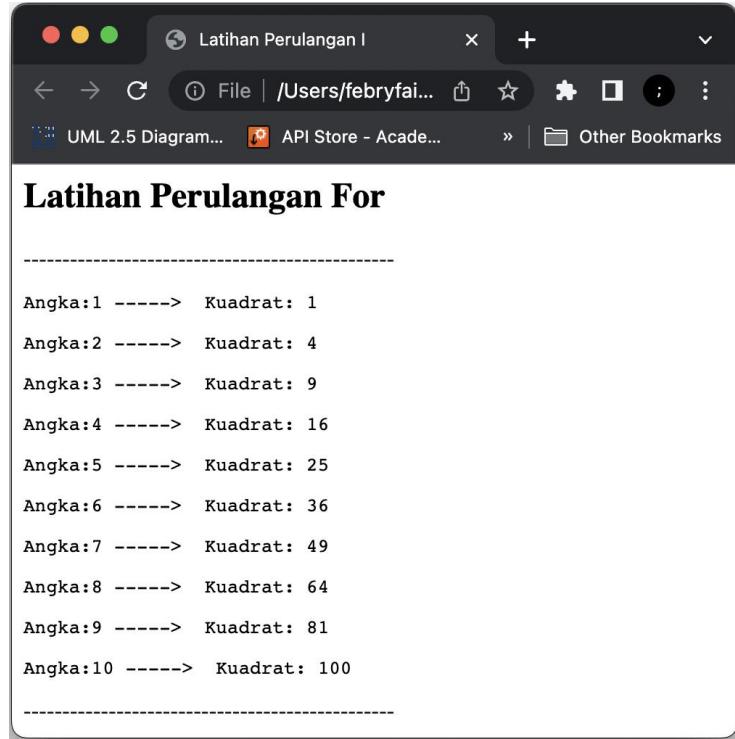
</BODY>

</HTML>

```



3.3.2 Perulangan



A screenshot of a web browser window titled "Latihan Perulangan I". The address bar shows the path "/Users/febryfai...". Below the address bar, there are several icons for bookmarks and tabs. The main content area displays the heading "Latihan Perulangan For" followed by a list of 10 lines of text, each showing a number from 1 to 10 followed by its square value separated by a double arrow: "Angka:1 -----> Kuadrat: 1", "Angka:2 -----> Kuadrat: 4", "Angka:3 -----> Kuadrat: 9", "Angka:4 -----> Kuadrat: 16", "Angka:5 -----> Kuadrat: 25", "Angka:6 -----> Kuadrat: 36", "Angka:7 -----> Kuadrat: 49", "Angka:8 -----> Kuadrat: 64", "Angka:9 -----> Kuadrat: 81", and "Angka:10 -----> Kuadrat: 100".

- **Perulangan While**

Perulangan lain yang dapat digunakan adalah dengan menggunakan perintah While. Perintah while digunakan untuk perulangan yang tidak diketahui berapa kali proses perulangannya. Perintah while terus mengulangi loop selama kondisi memiliki nilai true. Syntax untuk perintah while adalah sebagai berikut :

```
while (kondisi){ ulang pernyataan ini; }
```

Contoh Program

```
<HTML>

<HEAD><TITLE>Latihan Perulangan II</TITLE></HEAD>

<SCRIPT LANGUAGE="JavaScript">

var deret = prompt('Masukkan Jumlah Deret :,');

document.writeln("<H2>Latihan Perulangan While</H2>");

document.writeln("-----");

document.writeln("<BR>");

var jml = 0.0; var angka = 1;

while (angka <= deret) {

    jml= jml+angka;

    angka++
```

```

}

document.writeln("<BR>");

document.writeln("Jumlah Deret dari 1 sampai "+deret+" adalah = "+jml);

document.writeln("<BR>");

document.writeln("-----");

</SCRIPT>

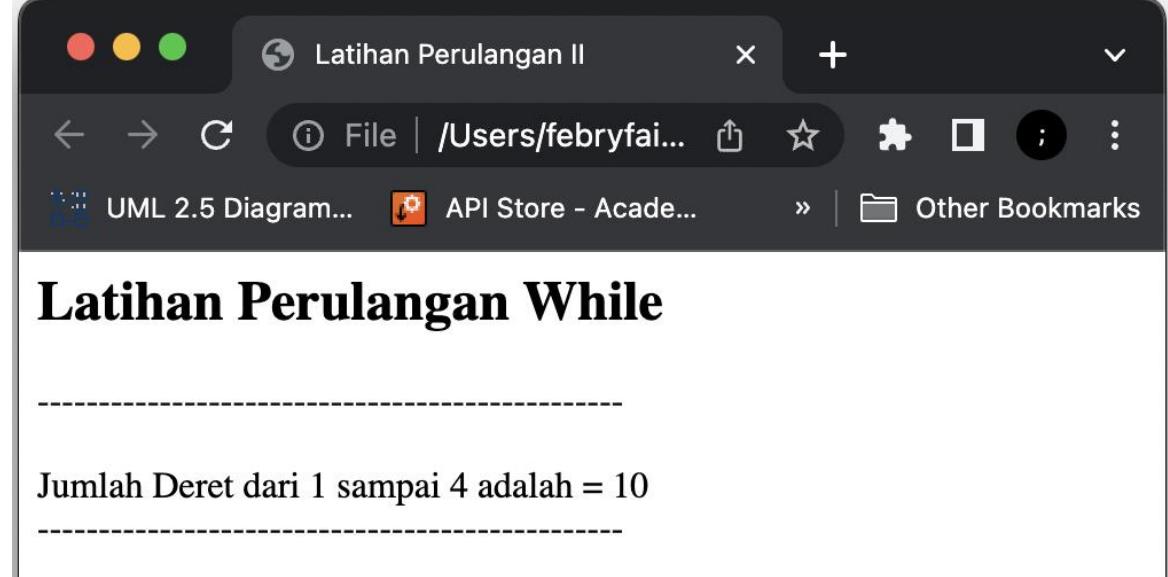
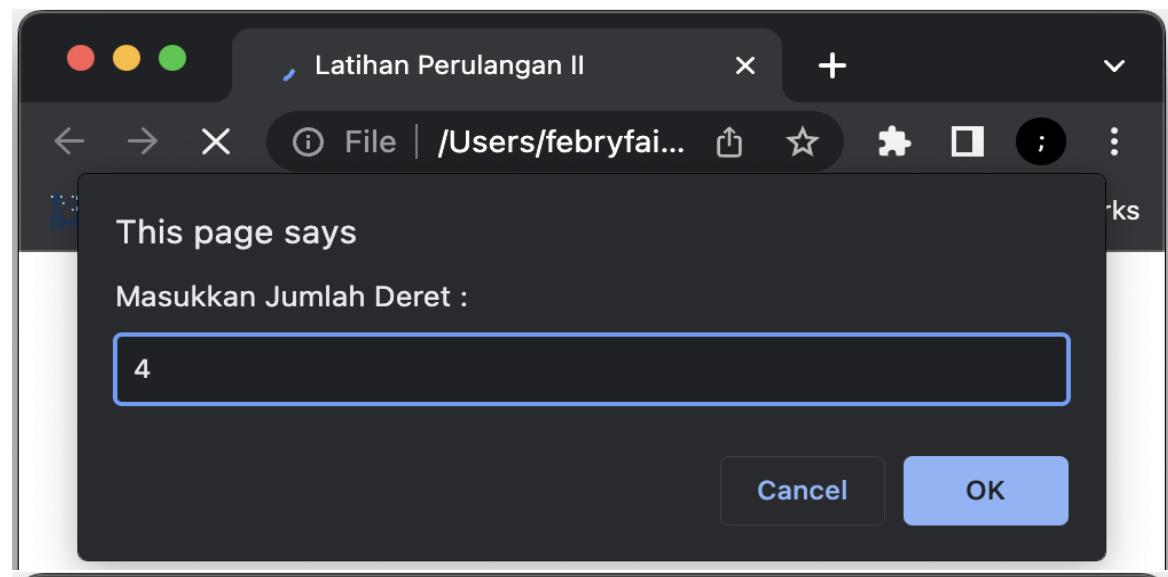
</HEAD>

<BODY>

</BODY>

</HTML>

```



- **Perulangan Do While**

Perulangan ini hampir sama seperti while, digunakan apabila kita belum tahu berapa kali perulangan harus dilakukan. Bedanya pernyataan do..while pengujiannya dilakukan di akhir pernyataan.

```
do{  
    //pernyataan1 dieksekusi  
}while (kondisi);
```

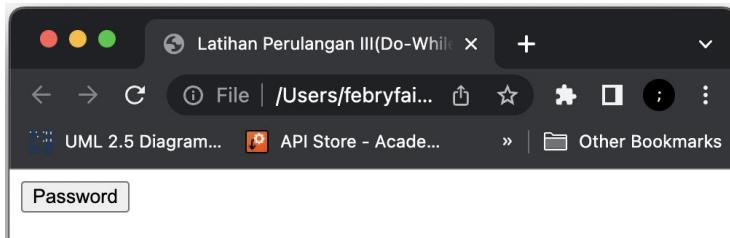
Contoh Program

```
<HTML>  
<HEAD><TITLE>Latihan Perulangan III(Do-While)</TITLE></HEAD>  
<BODY>  
<SCRIPT LANGUAGE="JavaScript">  
function pass(){  
    var coba = 1;  
do {  
    p = prompt("Tuliskan password dengan benar","");
    if (p=="IBIKJAYA"){  
        alert("Selamat Datang Friends");  
        window.open("Latiahn-1.html");  
        break;  
    } else {  
        alert("Password Salah !!! Ulangi lagi.");  
    } if  
(coba==3)  
{  
    alert("maaf, kesempatan anda hanya 3 kali"); history.go(-1);  
}  
coba=coba+  
1;  
} while  
(coba<=3);  
}</SCRIPT>
```

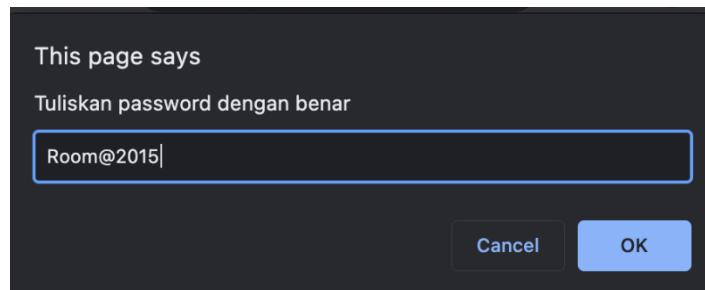
```
<FORM METHOD="post">  
<INPUT TYPE="button" VALUE="Password" ONCLICK="pass()">  
</FORM>  
</BODY>  
</HTML>
```

Adapun hasil yang diperoleh adalah sebagai berikut :

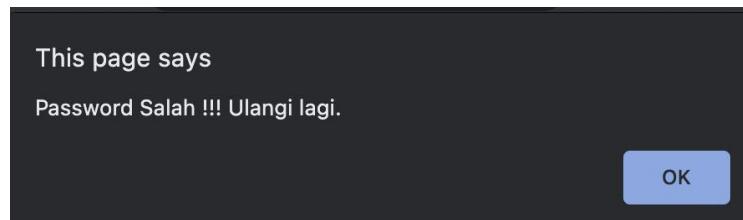
1. Tampilan awal



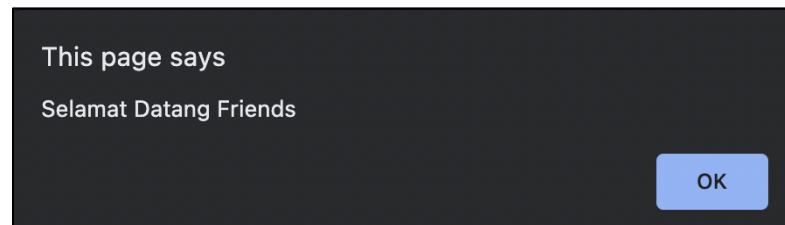
2. Prompt password



3. Alert apabila terjadi kesalahan password



4. Apabila password benar



3.4 Latihan Pembelajaran

1. Buatlah halaman seperti berikut, yang mencakup semua materi yang terdapat pada buku ini.
 - Input berupa : NIM, NAMA, JENIS KELAMIN, AGAMA, STATUS, JURUSAN, KOMENTAR. (SESUAIKAN OBJEK YANG DIPAKAI)
 - Proses terjadi di tombol KIRIM
 - Output berupa isian yang sama dengan soal nomor 1.
 - Tugas individu, dikumpulkan minggu ke 5, dalam bentuk print out KODE HTMLnya serta Tampilannya
2. Buatlah sebuah halaman form inputan, yang mencakup semua materi yang terdapat pada buku ini.

Perusahaan Travel IBIK Abadi memiliki armada dengan tujuan Jakarta, Solo dan Surabaya. Setiap tujuan memiliki kelas Eksekutif, Bisnis dan Ekonomi.

	Eksekutif	Bisnis	Ekonomi
Jakarta	70000	40000	10000
Solo	80000	50000	20000
Surabaya	90000	60000	30000

Diskon 10% diberikan apabila pemesan tiket merupakan anggota Travel Bintang Abadi.

Input : Nama Pemesan, Tujuan, Kelas, Banyak Tiket dan Status Member/Bukan
Output : Harga Tiket, Subtotal, Diskon dan Total Bayar

3. Buat program untuk menentukan faktorial dan jumlah deret sampai ke N.

PERHITUNGAN		
FAKTORIAL dan DERET		
N	<input type="text"/>	
Hasil		
Faktorial	<input type="text"/>	Submit Reset
Jumlah	<input type="text"/>	

4. Buat program untuk menghitung saldo akhir dari suatu tabungan dengan bunga dan jangka waktu tertentu :

Input :

Saldo : 100000

Bunga : 10

Waktu : 3

Ketika user menekan tombol Hitung maka akan muncul hasil seperti berikut :

Saldo Bulan 1 = Rp. 110000

Saldo Bulan 2 = Rp. 121000

Saldo Bulan 3 = Rp. 133100

BAB 4

JAVASCRIPT 2

4.1 Tujuan Pembelajaran

Mahasiswa memahami bentuk-bentuk dari berbagai macam object yang disediakan oleh Javascript untuk membantu membuat tampilan web lebih menarik dengan berbagai animasi.

4.2 Dasar Teori

4.2.1. Object Array

Array adalah suatu variabel yang dapat memuat beberapa nilai secara berurutan atau seri. Artinya variabel yang dideklarasikan sebagai array isinya tidak satu. Berikut adalah pendeklarasian untuk array.

```
nama = new Array(3)
```

Pernyataan diatas menunjukan bahwa variabel nama memiliki 3 elemen. Ketiga elemen tersebut akan memiliki nilai masing-masing 0, 1, 2. Nilai pertama = 0. Untuk mengisikan ketiga elemen tersebut kita dapat melakukan dengan cara :

```
nama[0] = "Febri", nama[1] = "Damatraseta", nama[2] = "Fairuz"
```

Selain cara diatas kita juga bisa mendeklarasikan array sekaligus dengan mengisikan elemen-elemennya.

```
nama = new Array("Febri", "Damatraseta", "Fairuz")
```

Contoh Program

<pre><HTML> <HEAD> <TITLE> LATihan Objek Array</TITLE> <BODY> <h3>Latihan Objek Array I</H3> Nama pada data ke 3 adalah : <SCRIPT LANGUAGE="JavaScript"> function cobaarray(){ nama = new Array("Febri", "Damatraseta", "Fairuz");</pre>	<pre><HTML> <HEAD> <TITLE> LATihan Objek Array</TITLE> <BODY> <h3>Latihan Objek Array II</H3> Pemanggilan data Array dengan Perintah Perulangan For
 <SCRIPT LANGUAGE="JavaScript"> var nilai = new Array(3);</pre>
--	---

<pre> document.write(nama[2]); } </SCRIPT> <SCRIPT LANGUAGE="JavaScript"> cobaarray(); </SCRIPT> </BODY> </HTML> </pre>	<pre> nilai[0]="A"; nilai[1]="B"; nilai[2]="C"; for (a=0;a<3;++a) { document.writeln("Nilai ke "+ [a+1] + " : "+nilai[a]+"
"); } </SCRIPT> <SCRIPT LANGUAGE="JavaScript"> </SCRIPT> </BODY> </HTML> </pre>
---	---

4.2.2. Object Date

Objek ini digunakan untuk memanipulasi tanggal dan waktu pada JavaScript. Untuk pendeklarasiannya adalah sebagai berikut :

tanggal = new Date()

Pernyataan diatas menyatakan bahwa variabel tanggal mengandung unsur tanggal dan waktu.

Metode-metode untuk Objek Date

Metode	Kegunaan
getDate()	Menghasilkan tanggal (integer) mulai 1 – 31.
getDay()	Menghasilkan hari(integer) mulai 0-6. Minggu = 0, Senin = 1,.....
getMonth()	Menghasilkan bulan(integer) mulai 0-11. Januari=0, Feb=1,.....
getFullYear()	Menampilkan tahun menjadi 4 digit
getHours()	Menghasilkan jam mulai 0-23
getMinutes()	Menghasilkan menit mulai 0-59
getSeconds()	Menghasilkan detik mulai 0-59

Contoh Program

```
<HTML>
<HEAD>
<TITLE> LATihan Objek Date</TITLE>
<BODY>
<h3 align="center">Latihan Objek Date/Tanggal</H3>
<SCRIPT LANGUAGE="JavaScript">
var hari = new
Array("Senin","Selasa","Rabu","Kamis","Jumat","Sabtu","Minggu");
var bulan = new Array("Januari","Februari","Maret","April",
"Mei","Juni","Juli","Agustus",
"September","Oktober","November","Desember");
var t = new Date();
var hari_ini=hari[t.getDay()-1];
var tanggal=t.getDate();
var bulan_ini=bulan[t.getMonth()];
var tahun=t.getYear();
var jam =t.getHours();
var menit =t.getMinutes();
var detik =t.getSeconds();
document.write("<font size=5 face=arial>");
document.write("<b><center>Sekarang adalah hari :" +hari_ini+", tanggal: "+
tanggal +" "+ bulan_ini +" " +tahun);
document.write("<hr width=700>"); document.write("</font>");
document.write("<font size=3 face=arial>");
document.write("<b><center>Jam sekarang = "+ jam +":"+menit+":"+detik);
document.write("</font>");
</SCRIPT>
<SCRIPT LANGUAGE="JavaScript">
</SCRIPT>
</BODY>
</HTML>
```

Latihan Objek Date/Tanggal

Sekarang adalah hari :Senin,tanggal: 11 Juli 122

Jam sekarang = 17:50:39

4.2.3. Object Math

Math digunakan untuk menangani perhitungan matematis yang rumit. Bentuk penulisan:

Math.metode(nilai)

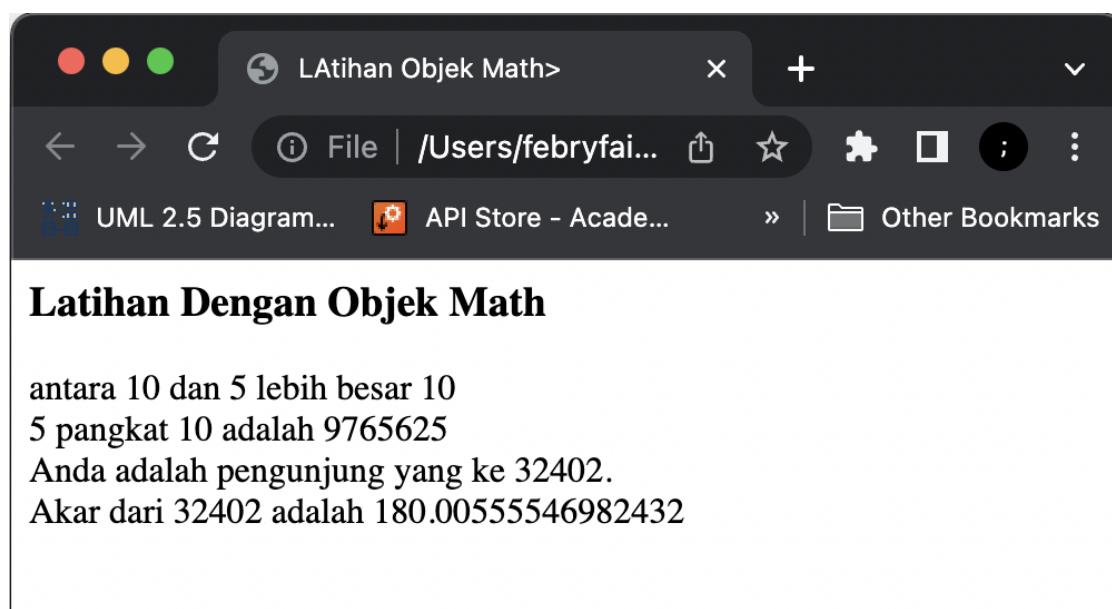
Metode Untuk Objek Math

Metode	Keterangan
abs(a)	Nilai absolut dari a
acos(a)	Nilai arc-kosinus dari a
asin(a)	Nilai arc-sinus dari a
atan(a)	Nilai arc-tan dari a
ceil(a)	Membulatkan nilai ke integer diatasnya
cos(a)	Nilai kosinus dari a
exp(a)	Nilai E pangkat a
log(a)	Nilai logaritma dari a
max(a,d)	Nilai terbesar dari a dan d
min(a,d)	Nilai terkecil dari a dan d
pow(a,d)	Nilai dari a pangkat d
random(a)	Nilai acak antara 0 dan 1
round(a)	Membulatkan nilai a ke integer terdekat
sqrt(a)	Nilai akar dari kuadrat a
sin(a)	Nilai sinus dari a

tan(a)	Nilai tangen dari a
--------	---------------------

Contoh Program

```
<HTML>
<HEAD>
<TITLE> LATihan Objek Math</TITLE>
<BODY>
<h3>Latihan Dengan Objek Math</h3>
<SCRIPT LANGUAGE="JavaScript">
var a=10; var b=5; besar=Math.max(a,b);
document.write("antara " + a + " dan " + b + " lebih besar "+besar+"<br>");
pangkat=Math.pow(b,a);
document.write(b+ " pangkat " + a + " adalah "+pangkat+"<br>");
var ran;
ran = Math.round(Math.random()*50000);
document.write("Anda adalah pengunjung yang ke " + ran + ".<br>");
var akar; akar =Math.sqrt(ran);
document.write ("Akar dari " + ran + " adalah "+akar);
</SCRIPT>
</BODY>
</HTML>
```



4.2.4. Object String

String adalah suatu objek yang merupakan kumpulan dari elemen karakter-karakter. Dalam Javascript string atau karakter harus diapit dengan tanda petik ganda(“) atau tanda petik tunggal(‘).

Contoh pendeklarasian Objek String :

Nama = “Shafana Vevica”

Panjang = Nama.length; // Panjang akan berisi 14

Length adalah properti yang sering digunakan dalam objek string yang digunakan Untuk mengetahui banyaknya karakter dalam suatu string.

Objek String juga memiliki method yang dapat digunakan untuk memanipulasi string tersebut. Adapun Method yang dapat digunakan meliputi :

Method	Fungsi
big()	Tercetak lebih besar
blink()	Efek berkedip aktif pada browerNetscape
bold()	Tercetak tebal
charAt(n)	Mengambil karakter ke -n dari string. Index string dimulai dari 0
fixed()	Tercetak fixed-pitch
fontcolor('warna')	Tercetak sesuai warna yang didefinisikan
indexOf('char')	Mengambil nilai indeks dari suatu karakter
italic()	Tercetak miring
link('url')	Menjadikan string hyperlink
small()	Tercetak lebih kecil
strike()	Tercetak dengan coretan
sub()	Tercetak subscript
substring(a,b)	Mengambil karakter dari posisi a sampai b-1
sup()	Tercetak superscript
toLowerCase()	Tercetak huruf kecil
toUpperCase()	Tercetak huruf besar

split('')	Menjadikan string diuraikan/dipisahkan berdasarkan tanda (''). Hasil dari split akan dihasilkan sebuah array dengan indeks 0 untuk string ke 1 dan seterusnya.
-----------	--

Contoh Program

```

<HTML>
<BODY>
<H3>Latihan Objek String</H3>
<SCRIPT LANGUAGE="Javascript"> nama ="Shafana Vevica";
panjang=nama.length; n=nama.substring(1,4);
besar=nama.toUpperCase();
namakulink=nama.link('shafa.html');
document.writeln('Namaku adalah =' + nama + '<BR>');
document.writeln('Panjang namaku adalah '+ panjang + ' karakter <BR>');
document.writeln('method BIG =' + nama.big() + '<BR>');
document.writeln('method SMALL =' + nama.small() + '<BR>');
document.writeln('method SUB =' + nama.sub() + '<BR>');
document.writeln('method SUP =' + nama.sup() + '<BR>');
document.writeln('method BOLD =' + nama.bold() + '<BR>');
document.writeln('method ITALIC =' + nama.italics() + '<BR>');
document.writeln('method FONTCOLOR = '+' + nama.fontcolor('red') + '<BR>');
document.writeln('method LOWERCASE = '+' + nama.toLowerCase() + '<BR>');
document.writeln('method UPPERCASE =' + besar + '<BR>');
document.writeln('method SUBSTRING =' + n + '<BR>');
document.writeln('method STRIKE =' + nama.strike() + '<BR>');
document.writeln('method CharAT =' + nama.charAt(3) + '<BR>');
document.writeln('method Link =' + namakulink + '<BR>');

```

```

document.writeln('Index Huruf c = '+ nama.indexOf("c") +'<BR>');

awal=nama.indexOf('V'); akhir=nama.length;

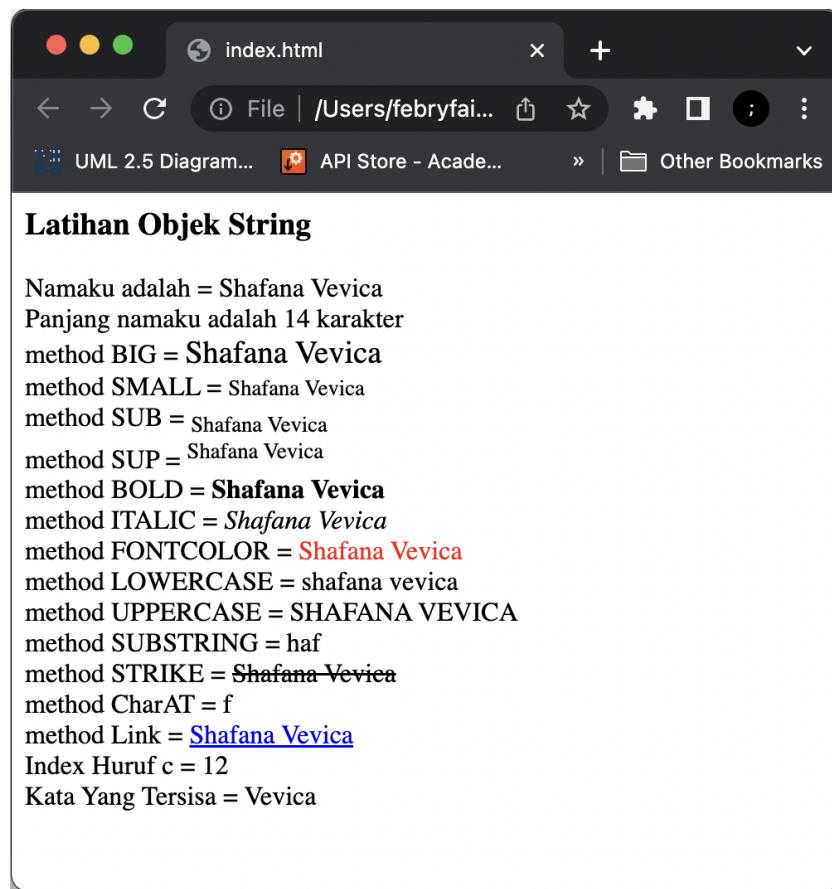
document.writeln('Kata Yang Tersisa = '+ nama.substring(awal,akhir)
+'<BR>');

</SCRIPT>

</BODY>

</HTML>

```



4.2.5. Object Document

Objek ini digunakan untuk mengakses informasi mengenai dokumen HTML, tampilan output dan memanipulasinya.

Property dari objek document meliputi :

Property	Fungsi
bgColor	Memberikan warna latar belakang

fgColor	Memberikan warna foreground atau warna huruf
link[]	Mengakses objek anchor/link(dapat digunakan nama objek anchor/link)
linkColor=warna	Memberikan warna link
alinkColor=warna	Memberikan warna pada active link
vlinkColor=warna	Memberikan warna pada visited link
title=judul window	Memberikan judul/title window
image[]	Mengakses objek image(dapat digunakan nama objek anchor/link)
forms[]	Mengakses objek form(dapat digunakan nama objek form)

Method dari objek document meliputi :

Method	Fungsi
open()	Menciptakan/membuka document HTML
close()	Mengakhiri document HTML
write(output)	Memberikan output ke browser
writeln(output)	Memberikan output ke browser dengan menyertakan perpindahan baris

Khusus untuk output ke browser ada beberapa hal yang perlu diperhatikan :

- Diisi dengan string(“”) atau (‘’)
- Dapat diberikan tag HTML
- Dapat digunakan untuk menampilkan isi dari variable
- Terdapat karakter spesial :

\b = untuk backspace

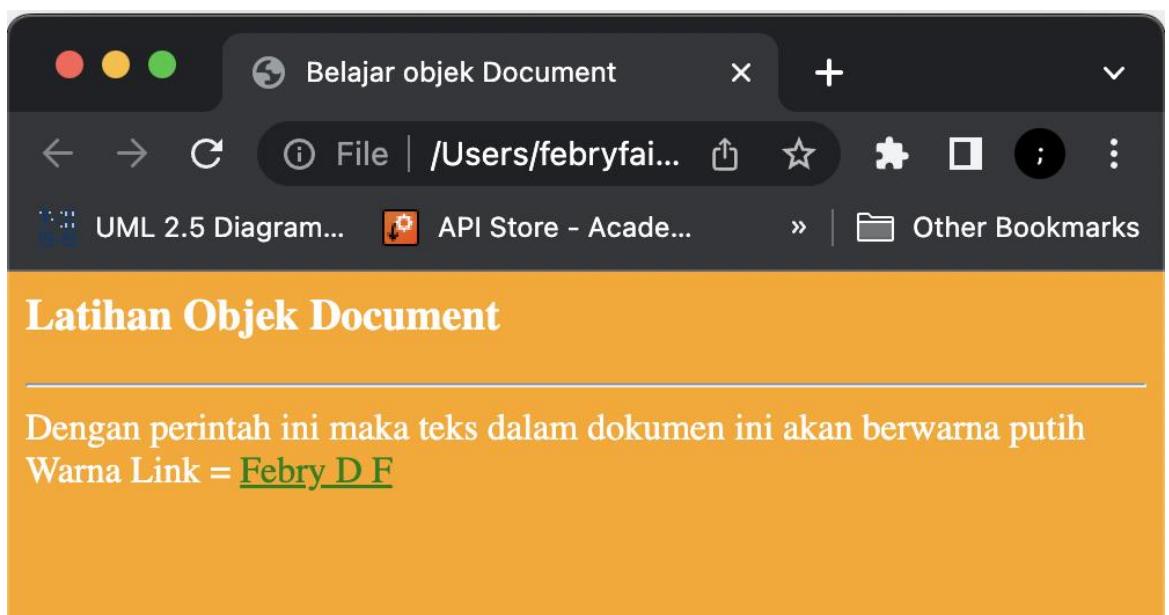
\f = untuk form feed \n = untuk
baris baru

\r = untuk carriage return

\t = untuk tab

Contoh Program

```
<HTML>
<BODY>
<H3>Latihan Objek Document</H3><hr>
Dengan perintah ini maka teks dalam dokumen ini akan berwarna
putih<BR>
<SCRIPT LANGUAGE="Javascript">
nama="Febry D F";
document.bgColor="orange";
document.fgColor="white";
document.title="Belajar objek Document";
document.linkColor="red";
document.vlinkColor="green";
document.alinkColor="white";
namakulink=nama.link('https://www.linkedin.com/in/febri-d-2b9537b1/');
document.writeln('Warna Link = '+ namakulink +'<BR>');
</SCRIPT>
</BODY>
</HTML>
```



4.2.5. Object Window

Objek window merupakan objek tertinggi dalam objek Javascript. Objek ini digunakan untuk memanipulasi tampilan jendela dari document HTML.

Properti pada Objek window

Property	Fungsi
length	Mengetahui jumlah frame pada window
location.href	Mengakses objek location untuk melakukan redirect atau berpindah ke alamat tertentu.
Status=nilai_status	Memberikan nilai status window

Metode-metode untuk Objek window

Method	Fungsi
alert(pesan)	Memunculkan messagebox sebuah pesan kesalahan
confirm(pesan)	Memunculkan pesan konfirmasi. Method ini akan menghasilkan dua nilai kembalian yaitu true untuk Ok dan false untuk Cancel
prompt(pesan,nilai default)	Memunculkan pesan yang menunggu sebuah input
close()	Menutup jendela aktif
open(url file,windo wname ,feature)	Membuka jendela baru dengan feature meliputi : toolbar=yes no mengaktifkan toolbar status=yes no mengaktifkan window status menubar=yes no mengaktifkan menu bar scrollbars=yes no mengaktifkan scroll bar resizable=yes no jendela resizable
	width = ukuran lebar jendela height = ukuran tinggi jendela
print()	Membuka jendela dialog print

Contoh Program

```
<HTML>
```

```

<BODY>

<H3>Latihan Objek
Window</H3><hr>

<SCRIPT LANGUAGE="Javascript">
window.status="Welcome";
window.alert("Selamat Datang");
angka=window.prompt("Inputkan
Angka?",0);

document.write("Angka favorit anda adalah =<strong>" +angka+
"</strong><br>");

tampung=window.confirm("jenis kelamin anda Pria
?"); if(tampung) {

document.write("Boleh Kenalan donk");

}

e
l
s
e
{

document.write("Ok dech");

}

window.close();

</SCRIPT>

</BODY>

</HTML>

```

Contoh penggunaan perintah window.open dan window.location untuk membuka halaman web lain.

Contoh Program

```

<HTML>

<BODY>

<CENTER><H3>Latihan Objek Document</H3><hr>

```

Membuka Web Page dengan Perintah Window.Open dan Window.Location

```

</CENTER>

<SCRIPT
LANGUAGE="Javascript">

function konek1()
{
    window.open("utsb.HTML");
}

function konek2()
{
    window.location="kunci_jawaban UTS.HTML";
}

</SCRIPT>

<FORM METHOD="post">
<P><CENTER>

<INPUT TYPE="button" VALUE="Kunci Jawaban UTS A"
ONCLICK="konek1()"

<INPUT TYPE="button" VALUE="Kunci Jawaban UTS B"
ONCLICK="konek2()"

</FORM></CENTER>

</BODY>

</HTML>

```

Contoh penggunaan objek window.location.href untuk membuka halaman web yang lain.

Contoh Program

```

<HTML>

<BODY>

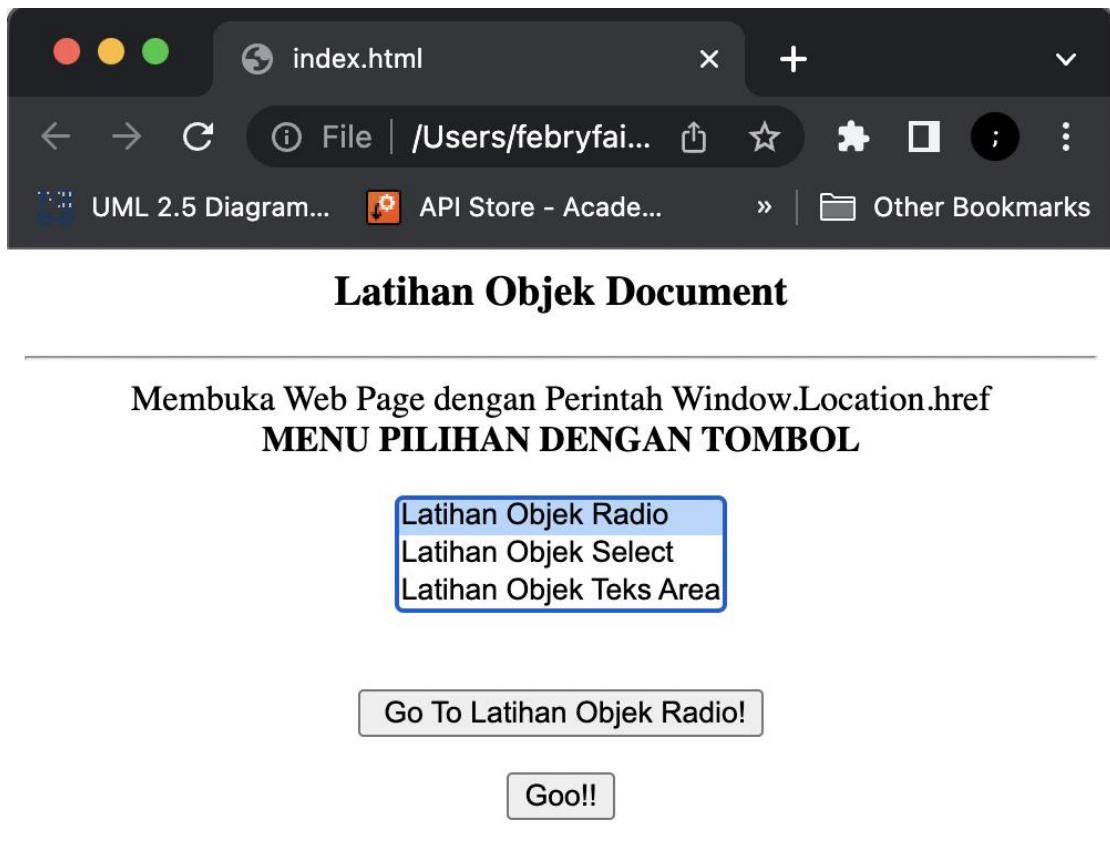
```

```

<CENTER><H3>Latihan Objek Document</H3><hr>
Membuka Web Page dengan Perintah Window.Location.href
</CENTER>
<SCRIPT LANGUAGE="Javascript">
function konek1()
{
if(document.pilihan.pilih.options[0].selected)
{
window.location.href="latobjekradio.HTML";
}
else if (document.pilihan.pilih.options[1].selected)
{
window.location.href="latobjekselect.HTML";
}
else if (document.pilihan.pilih.options[2].selected)
{
window.location.href="latobjekteksarea.HTML";
}
return true;
}
function konek2()
{
var pilihint; var pilihstr;
pilihint=document.pilihan.pilih.selectedIndex;
pilihstr=document.pilihan.pilih.options[pilihint].text;
document.pilihan.pilihteks.value=" Go To " + pilihstr + "!" ;
}
</SCRIPT>
<CENTER>
<FORM NAME="pilihan">
<B>MENU PILIHAN DENGAN TOMBOL</B>

```

```
<P><SELECT NAME="pilih" ONCHANGE="konek2()" MULTIPLE  
SIZE="3">  
    <OPTION>Latihan Objek Radio</OPTION>  
    <OPTION>Latihan Objek Select</OPTION>  
    <OPTION>Latihan Objek Teks Area</OPTION>  
</SELECT>  
</P>  
<P><BR>  
<INPUT TYPE="button" name="pilih" value="" size="40"  
maxlength="40">  
</P>  
<P>  
<INPUT TYPE="button" NAME="Gobutton" VALUE="Goo!!"  
ONCLICK="konek1()">  
</P>  
</FORM></CENTER>  
</BODY>  
</HTML>
```



4.3 Event Handler

Even adalah sesuatu yang terjadi pada halaman HTML. Berikut ini terdapat beberapa bentuk kejadian yaitu jika pengguna memuat dokumen, pengguna memasukkan data, pengguna mengklik tombol dan sebagainya. Hal-hal tersebut diatur oleh even. Semua kejadian pada Javascript dapat anda tangani dengan menentukan kejadiannya. Biasanya kejadian(event) adalah sebuah fungsi, tetapi pada beberapa kasus, kita dapat menuliskan pernyataan-pernyataannya secara langsung. Berikut ini adalah daftar kejadian(event) pada JavaScript :

Event	Keterangan
onClick	Kejadian yang dibangkitkan bila pengguna mengklik sebuah elemen form atau link.
onChange	Dibangkitkan bila informasi masukan pada sebuah elemen form (text, textarea, select) diubah oleh pengguna.
onBlur	Dibangkitkan ketika suatu elemen kehilangan focus masukan, yaitu ketika pengguna menekan tombol <tab> atau mengklik elemen lain form lainnya.

onFocus	Dibangkitkan bila sebuah elemen form menerima focus masukan; yaitu bila pengguna mengklik elemen form tersebut atau menekan tombol <tab> sehingga focus masukan berpindah ke elemen ini.
onAbort	Dibangkitkan bila pengguna menghentikan pemuatan citra (tag) yaitu bila pengguna menekan tombol stop atau mengklik link.
onError	Dibangkitkan bila terjadi kesalahan saat browser memuat dokumen atau citra.
onLoad	Dibangkitkan bila browser selesai memuat document
onUnload	Dibangkitkan bila pengguna keluar dari dokumen
onMouseOver	Dibangkitkan bila kursor mouse berada di atas sebuah link.
onMouseOut	Dibangkitkan bila kursor mouse keluar dari daerah link atau peta citra.
onReset	Dibangkitkan bila pengguna menekan tombol reset
onSelect	Kejadian yang dibangkitkan bila pengguna memilih sebagian atau seluruh teks pada elemen form yang berupa kotak teks.
onSubmit	Dibangkitkan ketika pengguna menekan tombol submit.

4.3.1 Penanganan Kejadian (Event)

Berikut ini akan diberikan beberapa contoh program-program yang menggunakan kejadian-kejadian dalam aplikasinya.

Contoh program yang menggunakan event **OnClick** :

```
<HTML>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
function warna(pilihan){
    alert("Anda Memilih Warna " + pilihan); document.bgColor=pilihan;
}
```

```

</SCRIPT>

<h1 align="center">Latihan Event OnClick</h1>

<hr width="500" color="black" noshade>

<h3 align="center">Pilih warna favorit anda.</h3>

<CENTER>

<FORM>

<INPUT TYPE="button" VALUE="Biru" onClick="warna('lightblue')">
<INPUT TYPE="button" VALUE="Pink" onClick="warna('pink')">
<INPUT TYPE="button" VALUE="Coklat" onClick="warna('burlywood')">
<INPUT TYPE="button" VALUE="Kelabu" onClick="warna('darkgray')">
<INPUT TYPE="button" VALUE="Oranye" onClick="warna('peachpuff')">
<INPUT TYPE="button" VALUE="Putih" onClick="warna('white')">

</FORM>

<FORM>

<IMG NAME="coolfan" SRC="https://www.ibik.ac.id/wp-content/uploads/2020/08/logo-ibik-web.png" height=72><BR><BR>
<INPUT TYPE=BUTTON VALUE=" Off " onClick="coolfan.src =
'https://www.ibik.ac.id/wp-content/uploads/2019/10/logo-ibik-web.png'">
<INPUT TYPE=BUTTON VALUE=" On " onClick="coolfan.src =
'https://www.ibik.ac.id/wp-content/uploads/2020/08/logo-ibik-web.png'">

</FORM>

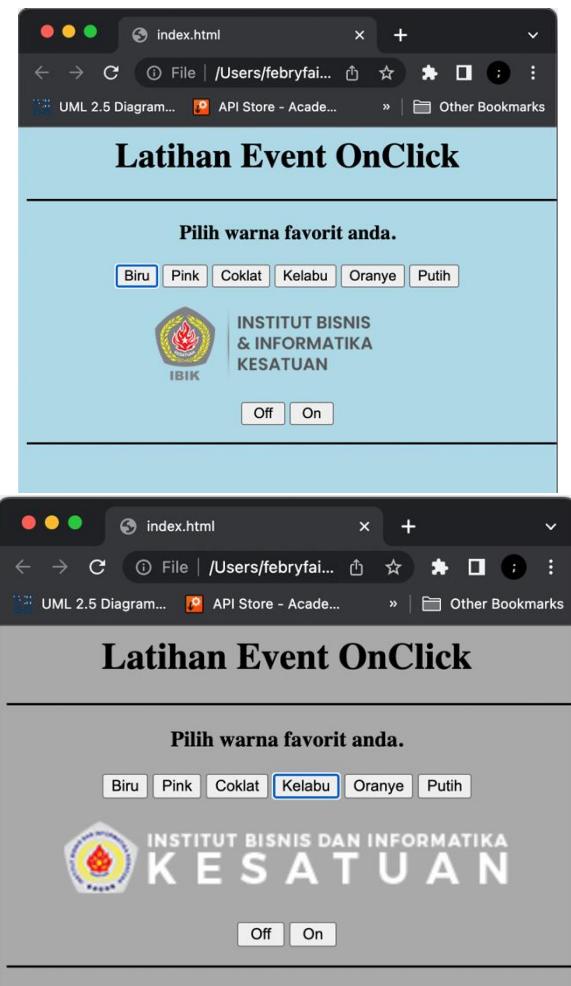
</CENTER>

<hr width="500" color="black" noshade>

</BODY>

</HTML>

```



Contoh program yang menggunakan even **OnBlur** dan **onFocus** :

```
<HTML>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
function masukannim() {
if (document.f.inim.value=="")
{
alert("anda belum memasukkan nim");
}
function masukannama()
{
if (document.f.inama.value=="")
{
alert("anda belum memasukkan nama");
}
function masukanalamat()
{
```

```

if (document.f.alamat.value=="")
{
    alert("anda belum memasukkan alamat");
}
}

function terimakasih()
{
if ((document.f.inim.value!="")&&(document.f.inama.value!="")&&
(document.f.alamat.value!=""))
{
    alert("Terima Kasih Telah mengisi Data");
} else
    alert("Mohon Data Dilengkapi");
}

</SCRIPT>

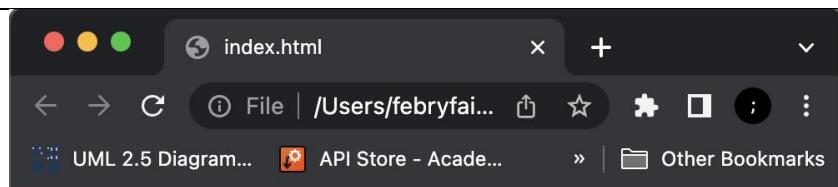
<h1 align="center">Latihan Event OnFocus dan OnBlur</h1>
<hr width="600" color="black" noshade size="10">
<font face="arial">
<h3 align="center">R E G I S T R A S I</h3>
<hr width="600" color="black" noshade size="2">
<CENTER>
<form name="f" method="get">
<TABLE>
    <tr>
        <td width="31%">NIM</td>
        <td width="7%">:</td>
        <td width="62%"><input type="text" name="inim" size="9" onFocus="window.status='Silahkan Mengisi NIM Anda';" onBlur="masukannim()"></td>
    </tr>
    <tr>
        <td width="31%">NAMA</td>
        <td width="7%">:</td>
        <td width="62%"><input type="text" name="inama" size="23" onFocus="window.status='Silahkan Mengisi Nama Anda';" onBlur="masukannama()"></td>
    </tr>
    <tr>
        <td width="31%">ALAMAT</td>
        <td width="7%">:</td>

```

```

<td width="62%><input type="text" name="ialamat" size="34"
onFocus="window.status='Silahkan Mengisi Alamat Anda';"
onBlur="masukanalamat()"></td>
</tr>
</table>
<hr width="600" color="black" noshade size="2">
<p><input type="BUTTON" value="Kirim" Onclick="terimakasih()">
<input type="reset" value="Reset"></p>
</form>
</center>
</BODY>
</HTML>

```



Latihan Event OnFocus dan OnBlur

R E G I S T R A S I

NIM	:	<input type="text"/>
NAMA	:	<input type="text"/>
ALAMAT	:	<input type="text"/>

Contoh program yang menggunakan **event onLoad** dan **onUnload**:

```

<HTML>

<BODY Onload="tanggal()" OnUnload="tutup()">

<SCRIPT LANGUAGE="JavaScript">

function tanggal() {
    var d = new Date();
    var y = d.getFullYear();
    var m = d.getMonth() + 1;

```

```

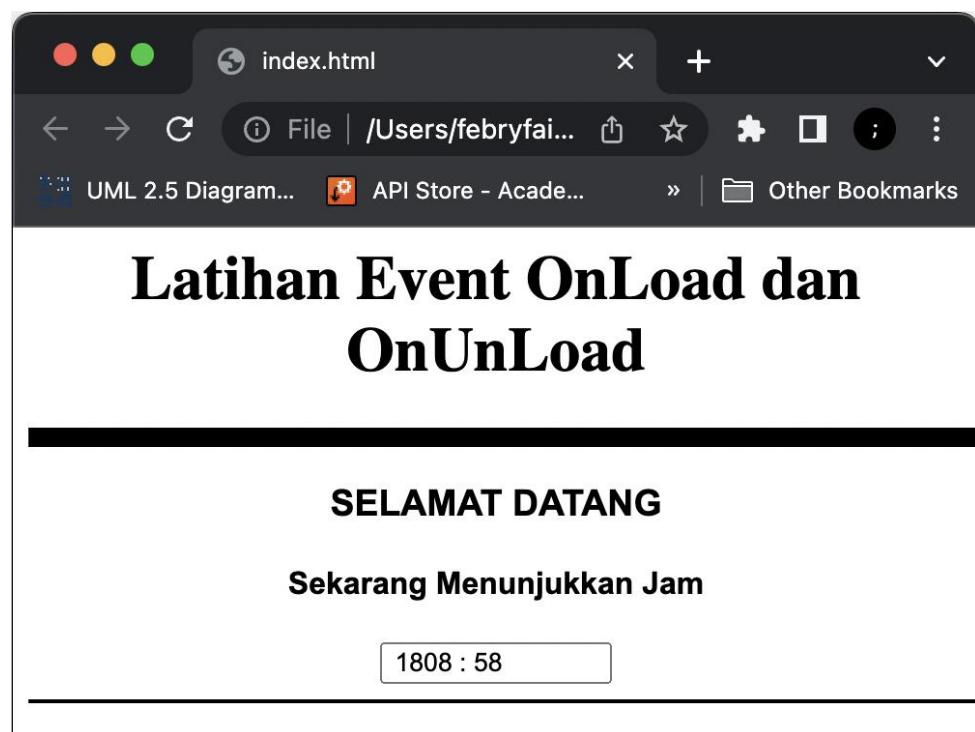
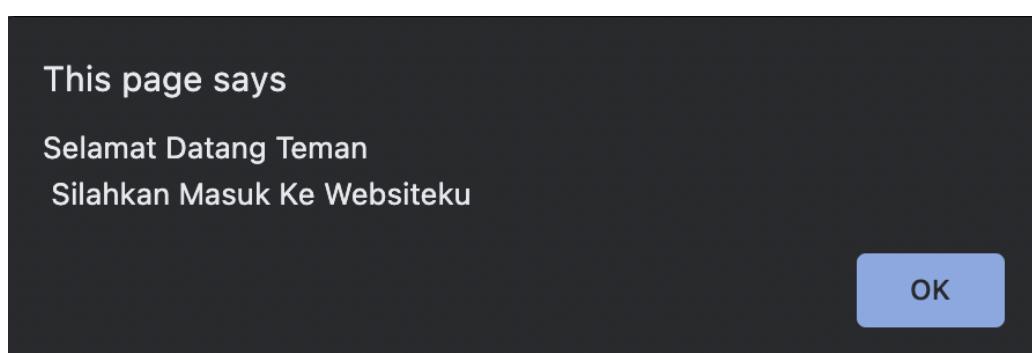
var d = d.getDate();
var t = d + '/' + m + '/' + y + '';
defaultStatus = "Anda datang pada tanggal " + t + ".";
alert("Selamat Datang Teman \n Silahkan Masuk Ke Websiteku");
}

function timer() {
    var d = new Date();
    var jam = d.getHours();
    var menit = d.getMinutes();
    var detik = d.getSeconds();
    var strwaktu = (jam<10)?"0"+jam:jam;
    strwaktu +=(menit<10)? "0"+menit:" "+menit;
    strwaktu +=(detik<10)? "0"+detik:" "+detik;
    document.f.txtwaktu.value= " "+strwaktu;
    setTimeout("timer()",200);
}
function tutup(){
    window.alert("Terimakasih Telah Berkunjung\nJangan lupa Datang kembali Ya..");
}
// end hiding -->
</SCRIPT>

<h1 align="center">Latihan Event OnLoad dan OnUnLoad</h1>
<hr width="600" color="black" noshade size="10">
<font face="arial">
<h3 align="center">SELAMAT DATANG</h3>
<CENTER>
<form name="f">

```

```
<h4 align="center">Sekarang Menunjukkan Jam</h4>
<input type="Text" size="16" name="txtwaktu">
<hr width="600" color="black" noshade size="2">
</form>
<SCRIPT LANGUAGE="JavaScript">timer()</SCRIPT>
</center>
</BODY>
</HTML>
```



Contoh program yang menggunakan event **onMouseOut** dan **onMouseOver**:

```
<HTML>
```

```

<BODY bgcolor="lightblue">
<SCRIPT LANGUAGE="JavaScript">
function g10{
    document.f.imgLogo.src="https://www.ibik.ac.id/wp-
content/uploads/2020/08/logo-ibik-web.png"
}
function g20{
    document.f.imgLogo.src="https://www.ibik.ac.id/wp-
content/uploads/2019/10/logo-ibik-web.png"
}
</SCRIPT>
<h1 align="center">Latihan Event OnMouseOver dan OnMouseOut</h1>
<hr width="600" color="black" noshade size="10">
<font face="arial">
<h3 align="center">SELAMAT DATANG</h3>
<CENTER>
<form name="f">
<Img      Name="logo"      height="72"      src="https://www.ibik.ac.id/wp-
content/uploads/2020/08/logo-ibik-web.png"
onmouseover="document.logo.src='https://www.ibik.ac.id/wp-
content/uploads/2019/10/logo-ibik-web.png';window.status='Ibik 1'"
onmouseout="document.logo.src='https://www.ibik.ac.id/wp-
content/uploads/2020/08/logo-ibik-web.png';window.status='Ibik 2'">
<h4 align="center">Kalo Kepanasan Tunjuk Kipas Angin Saja</h4>
<A HREF="latevenonclick.html" onMouseOver="g1()" onMouseOut="g2()">
<IMAGE      NAME="imgLogo"      SRC="https://www.ibik.ac.id/wp-
content/uploads/2020/08/logo-ibik-web.png" HEIGHT=72 BORDER=0>
</A>
<p>
Contoh Link Dengan Even
</p>

```

```

<A HREF="http://www.google.com" onMouseOver="document.bgColor='#ffcc00';
onMouseOver=window.status='Ke      Website      Google.Com';      return
true">Google.Com</A><BR>

<A                  HREF="https://www.ibik.ac.id/s1-teknologi-informasi/"
onMouseOver="window.status='Oh, jangan deket-deket..';
return true" onMouseOut="alert('Nah gitu dong'); return true">Teknologi Informasi
IBIK </A>

<hr width="600" color="black" noshade size="2">

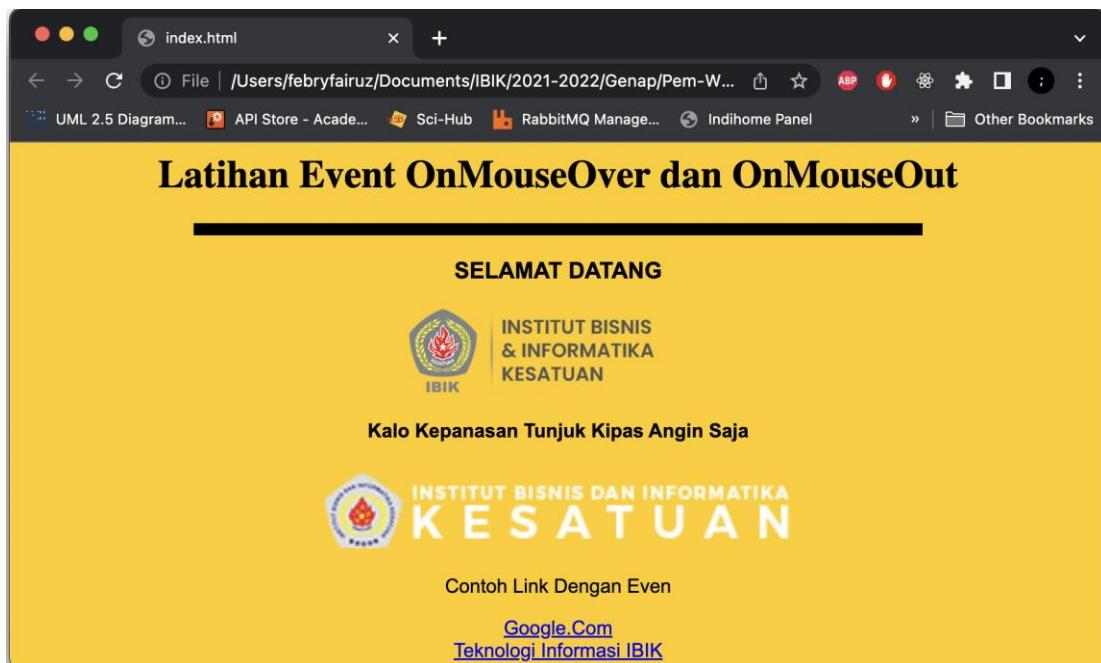
</form>

</center>

</BODY>

</HTML>

```



4.3.2 Event on Submit

Event on submit akan dibangkitkan apabila seorang user menekan tombol submit. Dengan event ini data yang diinputkan akan dikirimkan ke tempat lain (email, file teks atau ke dalam suatu tabel).

Contoh program

```

<Html>
<Body>

```

```

<SCRIPT LANGUAGE="JavaScript">

function cari() {

var kata = document.formcari.keyword.value; var hasil =
"http://www.google.com/search?q=" + kata ; window.open(hasil,
'google', config='height=500,width=750 scrollbars=yes
location=yes')

}

</SCRIPT>

<FORM NAME="formcari" onSubmit="cari()">

<center>

<table>

<tr>

<td colspan="4" bgcolor="orange"><h1 align="center">S e a r c h -
E n g i n e</h1><hr color="black" size="4" ></td>

</tr>

<tr>

<td><b>Cari pakai</b></td>

<td></td>

<td><INPUT NAME="keyword" SIZE="40" TYPE="text"></td>

<td><INPUT TYPE="submit" VALUE="Cari .. !!"><input type="reset"
Value="Ulang"></td>

</tr>

<tr>

<td colspan="4" bgcolor="orange"><hr color="black" size="4"
></td>

</tr>

</table>

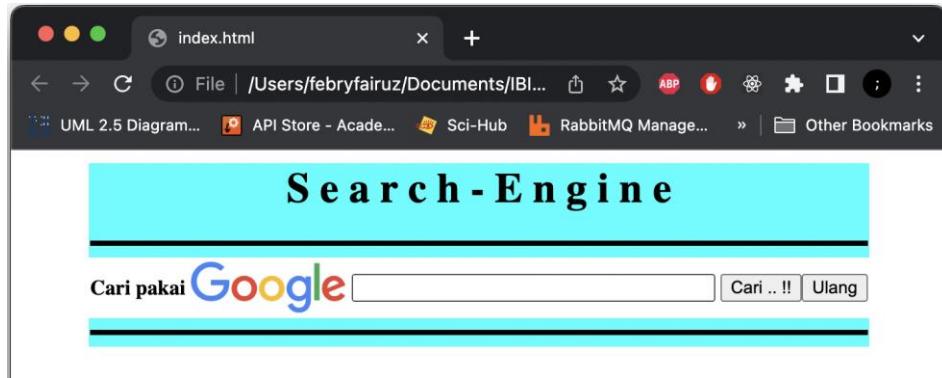
</center>

</FORM>

</body>

```

```
</html>
```



Berikut diberikan contoh event submit yang terjadi pada form pengisian data guestbook yang hasilnya akan dikirim ke suatu email.

```
<Html>
<Body>
<SCRIPT LANGUAGE="JavaScript">
function isiform(form) {
  isinama(form);
  isiemail(form);
  isikomentar(form);
  kosongkan(form);
}
function kosongkan(form){
  if((isinama(form) && isiemail(form) && isikomentar(form)))
  {
    form.submit;
  }
  if((isinama(form)== false || isiemail(form)== false || isikomentar(form)== false))
  {
    salahisi(form);
  }
}
function salahisi(form)
{
  var teks ="Ada Kesalahan Isian :"; if (isinama(form)== false) {teks +="\nNama Anda";} if (isiemail(form)==false) {teks +="\nEmail Anda";} if (isikomentar(form)==false) {teks +="\nKomentar Anda";} alert(teks);
}
function isinama(form)
{
```

```

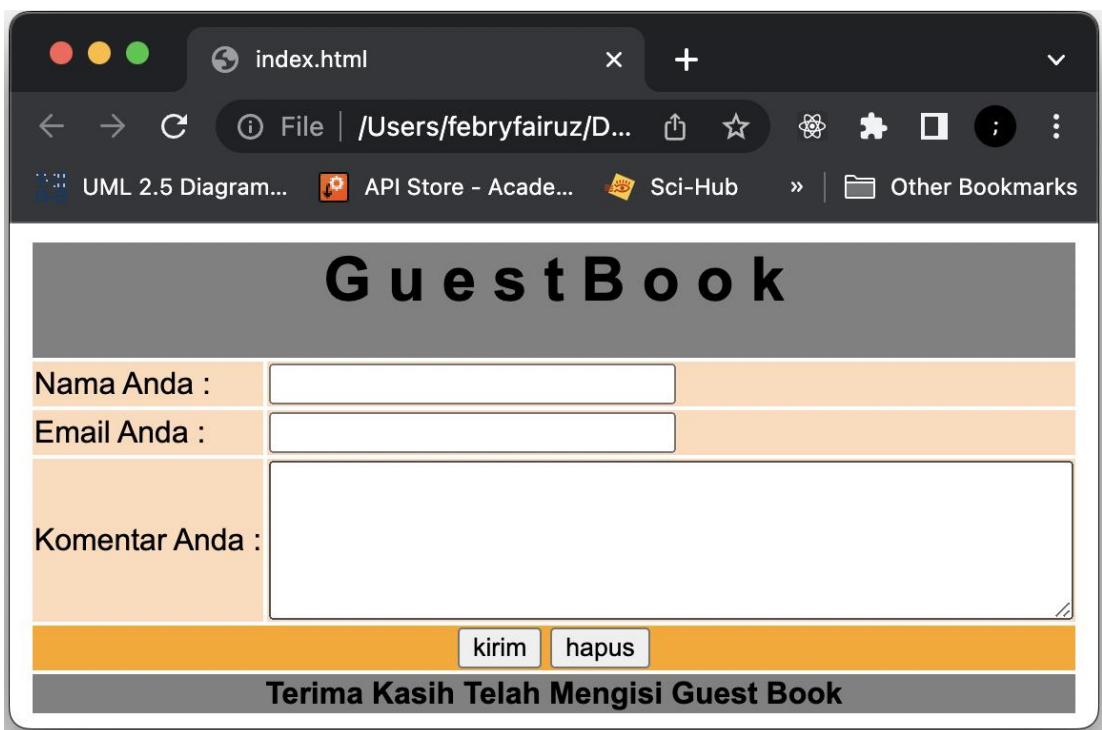
if (form.nama.value=="") {return false;} else {return true;}
}
function isiemail(form)
{
if((form.email.value==""      ||      form.email.value.indexOf('@',0)==-1)      ||
form.email.value.indexOf('.')==-1)
{return false;} else {return true;}
}
function isikomentar(form)
{
if(form.cs.value=="") {return false;} else
{return true;}
}
</script>
<form           name="form-guest"           method="post"
action="mailto:febrid@ibik.ac.id?subject=FormJS">
<font face="Arial">
<table align="center">
<tr bgcolor="gray">
<td colspan="2" align="center"><h1>G u e s t B o o k</h1></td>
</tr>
<tr bgcolor="peachpuff">
<td>Nama Anda :</td>
<td><input type="text" value="" name="nama" size="30"</td>
</tr>
<tr bgcolor="peachpuff">
<td>Email Anda :</td>
<td><input type="text" value="" name="email" size="30"</td>
</tr>
<tr bgcolor="peachpuff">
<td>Komentar Anda :</td>
<td><textarea name="cs" rows="5" cols="50"></textarea></td>
</tr>
<tr bgcolor="orange">
<td colspan="2" align="center">
<input      type="button"      name="thesubmit"      value="kirim"
onClick="isiform(this.form)">
<input type="reset" value="hapus">
</td>

```

```

</tr>
<tr bgcolor="gray">
<td colspan="2" align="center"><b>Terima Kasih Telah Mengisi
Guest Book</b> </td>
</tr>
</table>
</font>
</form>
</body>
</html>

```



Untuk memuat suatu image, pada Javascript terdapat objek Image. Untuk membuat objek tersebut pendeklarasiannya adalah sebagai berikut :

```
img1 = new Image()
```

```
img1.src = "pic1.gif"
```

Artinya membuat objek image dengan isinya adalah image pic1.gif berikut akan diberikan contoh mengenai objek image :

Contoh Program

```

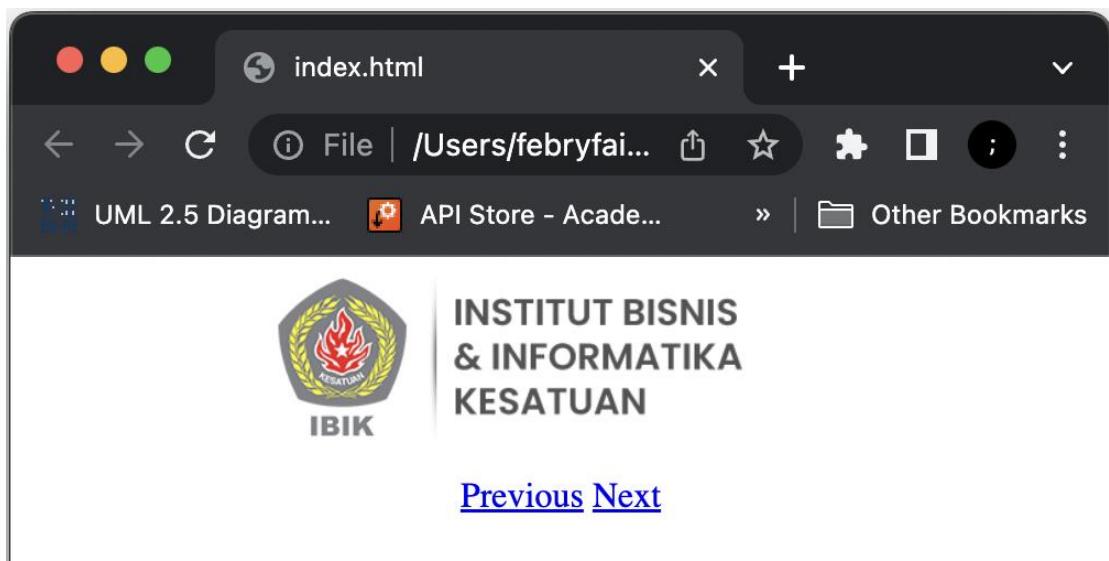
<HTML>
<BODY>
<SCRIPT LANGUAGE="JavaScript">

```

```

var num=1 ;
img1 = new Image();
img1.src = "pic1.gif";
img2 = new Image();
img2.src = "pic2.gif";
img3 = new Image();
img3.src = "pic3.gif";
img4 = new Image();
img4.src = "pic4.gif";
img5 = new Image();
img5.src = "pic5.gif";
img6 = new Image();
img6.src = "pic6.gif";
function slideshow(x){
num=num+x;
if (num==7) {
    num=1
}
if (num==0) {
    num=6
}
documentmypic.src=eval("img"+num+".src")
}
</SCRIPT>
<CENTER>
<IMG SRC="pic1.gif" NAME="mypic" BORDER=0 height="200"
width="150"><p>
<A HREF="JavaScript:slideshow(-1)">Previous</A>
<A HREF="JavaScript:slideshow(1)">Next</A>
</CENTER>
</BODY>
</HTML>

```



Contoh Program selanjutnya

```
<html>
<head>
<script language="javascript">
var image1=new Image();
image1.src="1.gif";
var image2=new Image();
image2.src="2.gif";
var image3=new Image();
image3.src="3.gif"
</script>
</head>
<body>
<center>
<h2>Penggunaan Objek Image Untuk Membuat SlideShow</H2><hr size=5
color="black">

<script>
function slideit(){
    document.images.slide.src=eval("image"+step+".src");
}
```

```

if (step<3) step++ else step=1 ;
}

slideit();

</script>

<hr size=5 color="black">

</body>

</html>

```



4.4 Latihan Pembelajaran

1. Buatlah program dengan javascript untuk memunculkan alert Selamat Pagi, Selamat Siang dan Selamat Malam. Sesuai dengan waktu yang tertera di komputer !!!!!

2. Buat program untuk menghitung/mencari akar-akar dari suatu persamaan $F(x) = ax^2+bx+c$
Rumus mencari akar x_1 dan x_2 adalah :

BAB 5

MONOLITH – LARAVEL

5.1 Tujuan Pembelajaran

Mahasiswa memahami dan menguasai pengembangan aplikasi web menggunakan kerangka kerja Laravel dalam arsitektur monolitik

5.2 Dasar Teori

5.2.1. Monolith

Arsitektur Monolith adalah sebuah arsitektur perangkat lunak yang terdiri dari satu aplikasi besar yang dibangun dan diimplementasikan sebagai satu unit tunggal. Aplikasi monolith biasanya dibangun menggunakan satu bahasa pemrograman dan berjalan pada satu platform atau lingkungan yang sama.

Secara umum, arsitektur monolith terdiri dari tiga layer utama: layer presentasi, layer bisnis, dan layer data. Berikut adalah penjelasan rinci tentang setiap layer:

1. Layer Presentasi

Layer presentasi bertanggung jawab untuk menampilkan informasi ke pengguna akhir dan menerima input dari mereka. Layer ini biasanya terdiri dari antarmuka pengguna (UI) dan logika presentasi. UI dapat berupa halaman web, aplikasi seluler, atau antarmuka pengguna desktop, tergantung pada kebutuhan aplikasi.

2. Layer Bisnis

Layer bisnis adalah pusat dari aplikasi monolith. Ini terdiri dari logika bisnis, seperti algoritma pengolahan data dan aturan bisnis yang mengatur perilaku aplikasi. Layer ini juga dapat mencakup lapisan logika jaringan atau layanan yang terhubung ke aplikasi monolith.

3. Layer Data

Layer data adalah tempat penyimpanan data aplikasi. Ini bisa menjadi database relasional, database NoSQL, atau sistem file. Layer data juga berisi logika akses data yang digunakan untuk membaca atau menulis data dari atau ke sumber data.

Kekuatan dari arsitektur monolith adalah mudah diimplementasikan, dipelihara, dan dikelola. Namun, ketika aplikasi tumbuh dan kompleksitasnya meningkat, arsitektur monolith dapat menyebabkan masalah. Perubahan pada satu bagian aplikasi dapat memengaruhi bagian lainnya, yang dapat menyebabkan kesulitan dalam pengujian dan pemeliharaan. Itu sebabnya, beberapa organisasi beralih ke arsitektur microservice untuk mengatasi masalah tersebut.

5.2.2 Laravel

Laravel adalah kerangka kerja aplikasi web PHP sumber terbuka yang didesain untuk memudahkan proses pengembangan aplikasi web yang elegan dan ekspresif. Laravel menyediakan seperangkat alat dan fitur yang memudahkan para pengembang

untuk membangun aplikasi web dengan pola arsitektur Model-View-Controller (MVC) yang terstruktur, mudah diubah, dan memiliki performa yang tinggi.

Laravel menyediakan berbagai fitur yang mempermudah pengembangan aplikasi web, seperti sistem routing yang fleksibel, sistem template tampilan yang efisien (blade templating engine), sistem migrasi basis data, sistem kueri basis data yang ekspresif (Eloquent ORM), sistem autentikasi dan otorisasi yang mudah digunakan, dan banyak lagi.

Selain itu, Laravel juga memiliki kemampuan untuk terintegrasi dengan berbagai layanan dan sistem lainnya, seperti sistem cache, sistem email, sistem queue, dan banyak lagi. Dengan dukungan dari komunitas pengembang yang besar dan aktif, Laravel terus berkembang dan menjadi salah satu kerangka kerja aplikasi web PHP yang paling populer dan banyak digunakan saat ini.

5.2.3. *Environment Development*

Sebelum membuat project Laravel, harus memastikan bahwa telah menginstal PHP dan Composer. Jika menggunakan macOS, PHP dan Composer dapat diinstall melalui Homebrew. Selain itu, laravel merekomendasikan untuk menginstal Node dan NPM.

XAMPP

Agar lebih mudah, gunakan software All-in-one Packages yang didalamnya sudah terdapat PHP yaitu XAMPP. Download file installer melalui link berikut <https://www.apachefriends.org/download.html> kemudian lakukan penginstalan seperti biasa. Untuk mengecek versi PHP ketik syntax berikut pada terminal

```
C:\Users\Irvan>php -v
PHP 8.0.23 (cli) (built: Aug 30 2022 12:56:19) ( ZTS Visual C++ 2019 x64 )
Copyright (c) The PHP Group
Zend Engine v4.0.23, Copyright (c) Zend Technologies
```

Composer

Download file composer installer melalui link <https://getcomposer.org/download/> kemudian install. Kemudian cek apakah sudah terinstall dengan benar dengan mengetik syntax berikut pada terminal

Laravel Project

Setelah Anda menginstal PHP dan Composer, Anda dapat membuat proyek Laravel baru melalui perintah `create-project` Composer :

```
composer create-project laravel/laravel example-app
```

Atau, dapat membuat project Laravel baru dengan menginstal Laravel secara global melalui Composer:

```
composer global require laravel/installer  
  
laravel new example-app
```

Setelah project dibuat, start Laravel local development server menggunakan perintah Servis Artisan CLI Laravel:

```
cd example-app  
  
php artisan serve
```

5.2.4. Architecture Concepts

Konsep dasar Laravel meliputi Routing, Views, Eloquent ORM, Controllers, Middleware, Models & Migrations, Requests, dan Responses.

Routing

Routing memungkinkan pengembang Laravel untuk merutekan permintaan aplikasi ke controller-nya. Ini adalah proses di mana semua permintaan diuraikan dengan bantuan rute. Perutean Laravel dibagi menjadi tiga kategori yaitu Basic routing, Route parameters, and Named routes..

Controller

Framework Laravel mengikuti pola arsitektur MVC di mana 'C' adalah singkatan dari Controller yang bertanggung jawab untuk menyampaikan informasi antara Model dan Tampilan. Jika Anda ingin membuat Controller, buka prompt perintah pada sistem operasi yang Anda gunakan dan ketik perintah berikut: `make [Name]Controller`

Views

Peran Views adalah untuk memisahkan logika presentasi dan logika bisnis. Ini digunakan untuk UI aplikasi dan terdiri dari HTML yang akan disajikan oleh aplikasi. Umumnya, file tampilan berisi informasi yang disajikan kepada pengguna. Juga, itu bisa berupa seluruh halaman web atau bagian dari halaman seperti footer atau header. Tampilan disimpan di direktori views/resources yang tepat.

Models & Migrations

Model adalah media untuk mengelola logika bisnis dalam aplikasi yang dikaitkan dengan kerangka kerja MVC. Sebagian besar digunakan untuk berinteraksi dengan database dengan memungkinkan untuk memasukkan, memperbarui, dan mengambil informasi dalam tabel data.

Di sisi lain, Migrasi memberikan cara untuk mencatat perubahan yang dibuat di tabel data selama proses pengembangan. Selain itu, Migrasi menawarkan pendekatan yang bermanfaat untuk pengembangan kolaboratif untuk menjaga sinkronisasi database tanpa perlu menghapus dan membuat ulang database setiap kali terjadi perubahan.

Request and Responses

Request di Laravel berfokus pada penanganan semua permintaan HTTP dari ujung klien. Perusahaan pengembangan Laravel dapat mengakses semua input klien dengan komponen ini. Kita dapat menggunakan Permintaan di mana saja seperti permintaan yang ditentukan()>field_name.

Objek Response adalah tindakan pantulan ke browser pengguna, setiap kali pengguna mengirimkan permintaan. Semua pengontrol dan rute harus mengembalikan beberapa respons sehubungan dengan permintaan tersebut.

Middleware

Middleware menjembatani gap antara permintaan dan tanggapan. Ini adalah semacam mekanisme pemfilteran yang memverifikasi identitas pengguna aplikasi dan memeriksa autentikasinya. Jika pengguna diautentikasi, Middleware mengarahkannya ke beranda jika tidak, dia akan dialihkan ke halaman login. Untuk menjalankan Middleware, kita dapat mengikuti perintah:

PHP artisan make:middleware

Eloquent ORM

Laravel memiliki ORM (Object-relational Mapping) yang paling kuat. Ini adalah teknik pemrograman yang memungkinkan Implementasi Catatan Aktif PHP yang mudah. ORM yang fasih memberdayakan pengembang Laravel untuk meningkatkan kueri atau masalah basis data dengan sintaks PHP yang disederhanakan menghilangkan penulisan kode yang rumit dalam kode SQL. Ini menciptakan interaksi yang lancar bagi para pengembang dengan tabel database dengan menyediakan model yang sesuai untuk setiap tabel.

Ini adalah konsep inti Laravel yang harus dikuasai oleh setiap developer Laravel dalam pengembangan aplikasi web yang gesit. Laravel telah mengalahkan kerangka kerja PHP lainnya di setiap bidang pengembangan front-end.

5.3 Latihan Pembelajaran

Buatlah tampilan website yang berisi data diri menggunakan framework laravel

BAB 6

LARAVEL MVC, ROUTE DAN BLADE FRAGMENT

6.1 Tujuan Pembelajaran

1. Memisahkan logika bisnis (Model), tampilan (View), dan pengendali (Controller) dalam sebuah aplikasi web.
2. Mengatur bagaimana URL dari aplikasi kita akan berinteraksi dengan kode di belakangnya.
3. Memecah tampilan (UI) menjadi bagian-bagian yang dapat digunakan ulang.

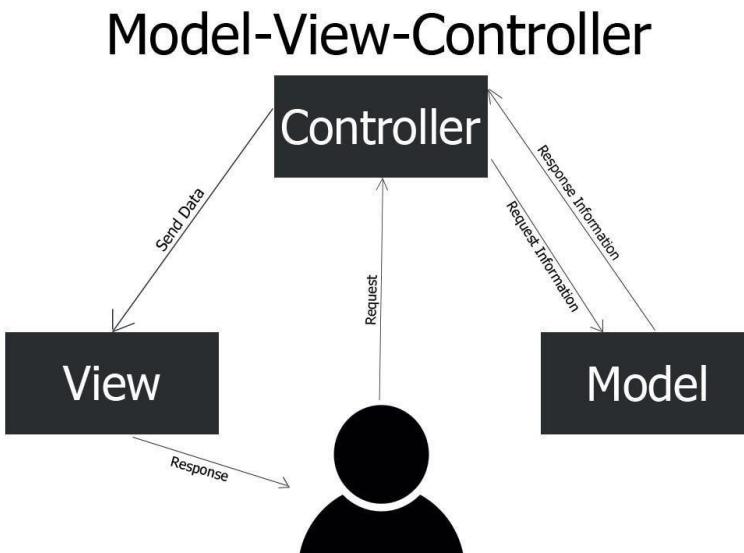
6.2 Dasar Teori

6.2.1 MVC

MVC adalah sebuah arsitektur perancangan kode program. Tujuannya untuk memecah kode program utama menjadi 3 komponen terpisah dengan tugas yang spesifik. Ketiga komponen tersebut adalah:

- Pengaksesan database, disebut sebagai **Model**.
- Tampilan design (user interface), disebut sebagai **View**.
- Alur logika program, disebut sebagai **Controller**.

Ide awal dari perlunya konsep MVC adalah agar aplikasi yang dibuat bisa mudah dikelola dan dikembangkan, terutama untuk aplikasi besar. Arsitektur MVC ini ditawarkan oleh Bahasa pemrograman PHP sebagai pengganti bentuk Object Oriented Programming. Karena PHP tidak dapat mengimplementasikan OOP secara baik.



Gambar 6.2.1.1. Arsitektur MVC pada Laravel

Ini adalah contoh alur yang terjadi dalam sebuah aplikasi yang menerapkan konsep MVC. Kita berangkat dari user yang sedang membuka sebuah web browser.

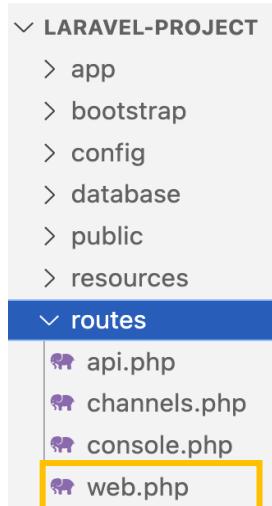
Setiap interaksi yang dilakukan user akan ditangani oleh Controller. Misalnya ketika user mengetik alamat situs www.kis.ibik.ac.id, maka sebuah controller di server KIS akan menangkap “request” tersebut. Atau ketika user selesai mengisi form register dan men-klik tombol submit, file controller akan menerima sebuah proses.

Controller pada dasarnya berisi logika program. Seandainya perlu mengambil data dari database, maka controller akan memanggil Model. Model inilah yang bertanggung jawab mengakses database lalu mengembalikan hasilnya kembali ke controller.

Setelah data dari model diterima kembali, controller kemudian meneruskan data tersebut ke dalam View. Data ini kemudian diproses sebagai kode HTML dan CSS di dalam view. Inilah yang dilihat oleh user di dalam web browser.

6.2.2. Laravel Route

Route berperan sebagai penghubung antara user dengan keseluruhan framework. Dalam Laravel, setiap alamat web yang kita ketik di web browser akan melewati route terlebih dahulu. Route lah yang menentukan kemana proses akan dibawa, apakah ke Controller atau ke View. Untuk mengakses file route masuklah kedalam project anda carilah folder routes dan buka file web.php.



Gambar 6.2.2.1. Structure folder Route

1. Basic Route

Berikut ini beberapa contoh penggunaan Route secara basic untuk menampilkan sebuah text berdasarkan alamat website:

Web.php

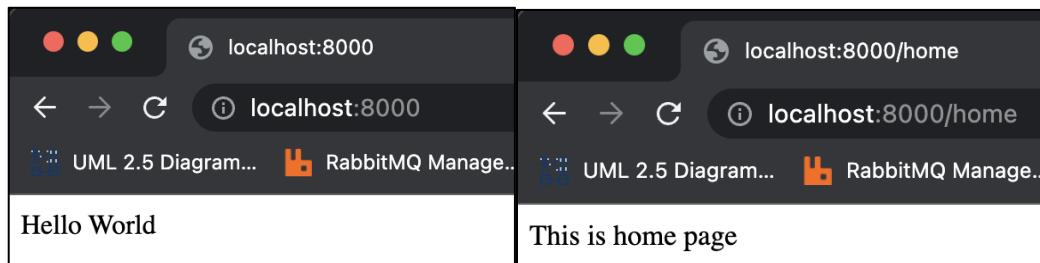
```
Route::get('/', function () {
    return "Hello World";
});
```

```
Route::get('/home', function () {
    return "This is home page";
});
```

Inilah kode program yang dipanggil untuk memproses alamat `http://localhost:8000`. Secara sederhana, penulisan route Laravel mengikuti format berikut:

```
Route::<jenis method>(<alamat URL>,<proses yang dijalankan>)
```

Tampilan alamat website berdasarkan end-point yang telah didaftarkan di `Web.php`:



Gambar 1.1. Contoh dasar inisialisasi route

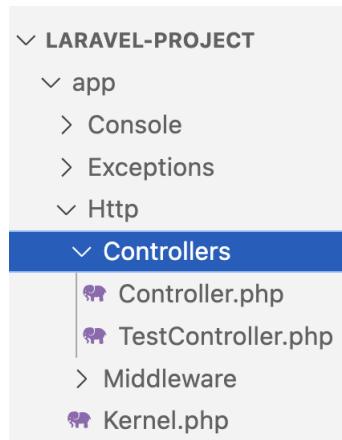
Pertama, kita harus menulis perintah ‘`Route::`’. Di dalam konsep MVC PHP, tanda `::` merupakan perintah untuk mengakses sebuah static method (atau bisa juga static property) kepunyaan sebuah class, yang dalam contoh ini adalah milik class `Route`.

Selanjutnya diikuti dengan penulisan `<jenis method>`. Jenis method adalah ‘cara sebuah URL diakses’. ‘`get`’ merupakan salah satu jenis method yang akan dijalankan ketika alamat URL diakses secara normal.

Maksud “normal” di sini adalah dengan mengetik langsung alamat web di address bar web browser atau ketika kita mengklik sebuah link. Nantinya terdapat jenis method lain seperti `post`, `put`, dan `delete`.

2. Route dengan basic Controller

Pada penggunaan route kali ini akan memanfaatkan Controller sebagai wadah untuk menampung logika program. Controller merupakan suatu bagian penting dari pemrograman MVC yang berfungsi sebagai penghubung antara View dan model. Didalam controller akan terdapat banyak logika-logika pemrograman untuk menyusun fungsi tertentu. Berbagai pemrosesan juga pada umumnya dilakukan di dalam controller. Secara bawaan file controller akan disimpan pada folder `app/Http/Controllers`.



Gambar 2.1. Lokasi directory Controller

Untuk membuat sebuah controller kita dapat membuatnya dengan cara memasukan syntax dibawah ini pada terminal/cmd project Laravel anda:

```
php artisan make:controller [namafile]
```

```
febryfairuz@Febrys-MacBook-Air laravel-project % php artisan make:controller ProfileController
```

```
INFO Controller [app/Http/Controllers/ProfileController.php] created successfully.
```

Pada contoh syntax diatas, akan membuat sebuah controller bernama ProfileController. Jika berhasil seperti pada contoh diatas maka akan membuat file seperti pada direktori gambar 2.2.1 bernama *ProfileController*. Pada file ProfileController silakan anda membuat sebuah function dengan nama index, identity, dan family.

ProfileController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProfileController extends Controller
{
    public function index(){
        return "Welcome to Profile page";
    }

    public function identity(){
        return "Welcome to Profile page sub menu identity";
    }

    public function family(){
        return "Welcome to Profile page sub menu family";
    }
}
```

Selanjutnya definisikanlah 3 buah route bernama profile yang memanggil controller ProfileController:

```
use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;

Route::get('/profile', [ProfileController::class, 'index']);
Route::get('/profile/identity', [ProfileController::class, 'identity']);
Route::get('/profile/family', [ProfileController::class, 'family']);
```

Setiap memanggil sebuah controller pada file route anda perlu memanggil file tersebut. Contoh pada script diatas memanggil sebuah file bernama ProfileController untuk ketiga buah end-point. File controller tersebut wajib dipanggil dengan menambahkan:

```
use App\Http\Controllers\ProfileController;
```

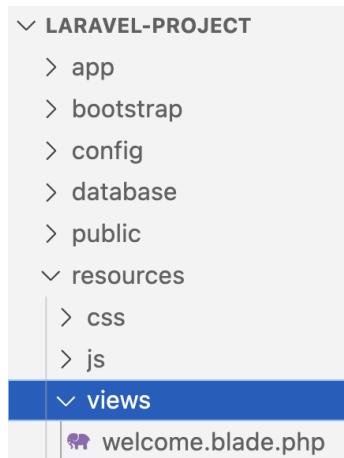
Berikan output dari halaman ini, dan simpan sebagai bentuk **Latihan soal nomor 1.**

3. Route dengan Controller dan View

Laravel sudah memiliki 1 view bawaan yang terlihat saat mengakses halaman homepage atau halaman root Laravel.

```
Route::get('/', function () {
    return view('welcome');
});
```

Kode ini bisa dibaca: “*Jika halaman root ‘/’ diakses, tampilkan view bernama welcome*”. Di dalam Laravel, nama sebuah view mewakili nama suatu file, artinya harus terdapat file bernama ‘welcome’ di dalam folder instalasi Laravel. Letak direktori penyimpanan view di dalam folder resources\views.



Gambar 3.1. Direktori file views laravel

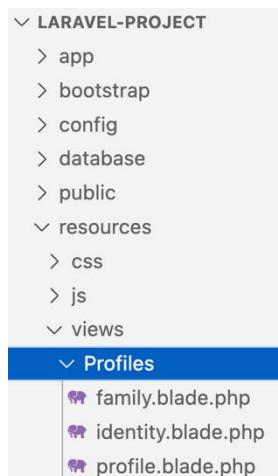
Dalam Laravel, setiap view harus ditulis dengan format berikut:

```
<nama_file>.blade.php
```

Sehingga jika kita ingin membuat view baru untuk *profile*, maka nama filenya adalah **profile.blade.php**.

Blade merupakan sebuah templating engine bawaan Laravel. Bahasan lebih lanjut tentang blade akan dibahas dalam section setelah ini. Perlu dipahami bahwa semua file view Laravel **harus** diakhiri dengan **.blade.php**. Selain itu file view juga harus berada di dalam folder resources\views.

Melanjutkan contoh kasus route pada ProfileController, bukalah file tersebut dan tambahkan function view untuk memanggil sebuah file dari masing-masing end-point yang telah didaftarkan.



Buatlah 3 buah file view blade yang berada didalam folder Profiles:

- **profile.blade.php**
- **identity.blade.php**, dan
- **family.blade.php**

Setelah membuat ketiga file view blade silakan anda isi file tersebut dengan syntax HTML untuk menampilkan sebuah informasi secara berbeda-beda.

Pada file ProfileController, silakan anda ubah script return output pada setiap function dengan memanggil file view blade:

ProfileController.php

```
<?php

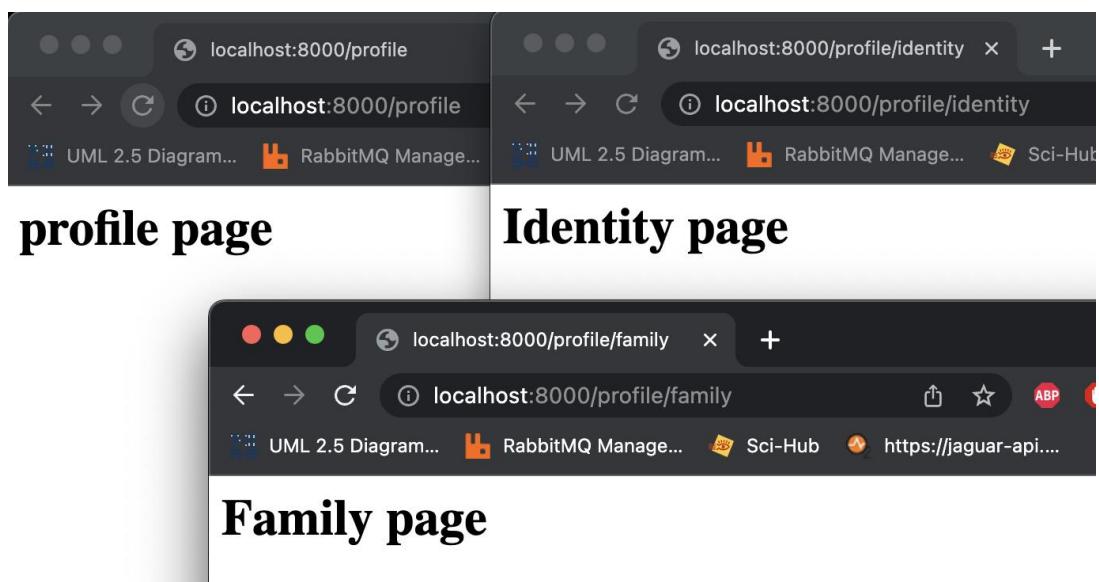
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProfileController extends Controller
{
    public function index(){
        return view('Profiles.profile');
    }

    public function identity(){
        return view('Profiles.identity');
    }

    public function family(){
        return view('Profiles.family');
    }
}
```



Gambar 3.2 Tampilan route dengan controller dan view

6.2.3. Laravel Blade

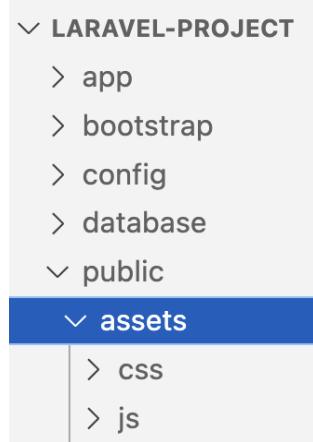
1. Blade Layout

Jika pada bab 1-3 anda telah mendesain sebuah website dengan bentuk web DYNAMIC, pada bagian ini kita akan membuat sebuah website dengan bentuk STATIC.

Disini akan menggunakan Framework CSS – Bootstrap v.5 untuk mempermudah mengatur layout pada website.

Silahkan anda download bundle package bootstrap di: <https://getbootstrap.com/docs/5.3/getting-started/download/>

Setelah berhasil anda download extraklah file tersebut dan pindahkan ke dalam project Laravel anda di folder **public** dan **rename** folder '**bootstrap-5.3.0-alpha2-dist**' menjadi '**assets**'. Jika anda memiliki file media lainnya seperti gambar, video ataupun memiliki file custom CSS atau JS anda bisa menyimpannya didalam folder assets.

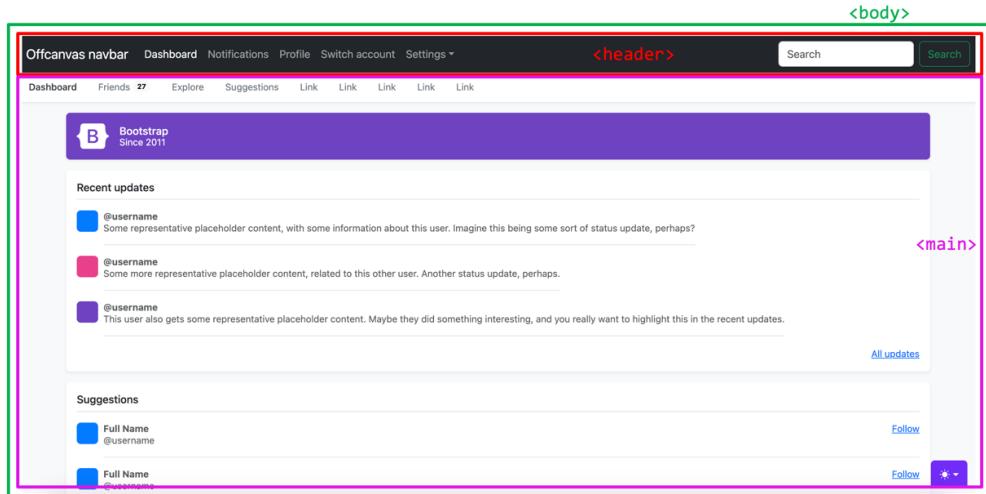


Gambar 1.1 Direktori folder media assets

A. Mengatur Blade Layout

Pada contoh kasus ini akan mengambil sebuah bentuk template website milik bootstrap yang dapat dilihat pada:

<https://getbootstrap.com/docs/5.3/examples/offcanvas-navbar/>



Gambar Layout template bootstrap

Jika kita breakdown bentuk layout diatas maka didapati seperti dibawah ini:

```

<html>
<head>
  <title>PWL - IBIK</title>
</head>
<body>
  <header>
    ...
  </header>
  <main>
    ...
  </main>

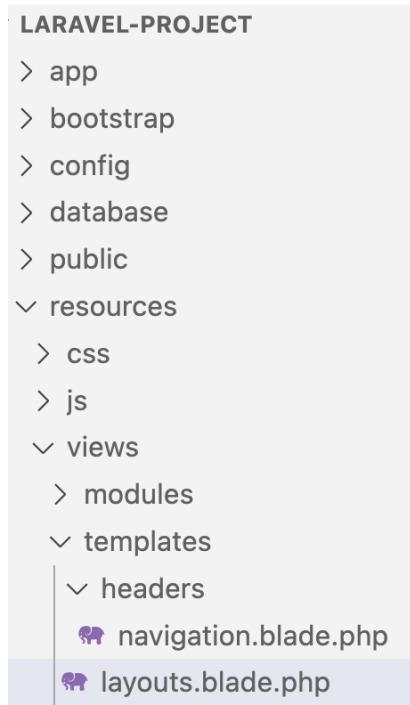
```

```

</main>
<footer>
...
</footer>
</body>
</html>

```

Pada project Laravel anda, buatlah structure folder pada view seperti gambar dibawah ini:



Gambar Structure folder blade layout

Pada gambar diatas folder templates diperuntukan bagi file-file yang berhubungan dengan template UI. Sedangkan folder modules berisi file-file mentah yang akan dijadikan komponen main web static.

Pada file layouts.blade.php akan berisi structure html seperti yang telah di break down sebelumnya.

Layouts.blade.php

```

<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>PW-IBIK</title>
<link rel="stylesheet" href="{{ url('./assets/css/bootstrap.min.css') }}>
<link rel="stylesheet" href="{{ url('./assets/bootstrap-icons/font/bootstrap-icons.css') }}>

</head>

<body class="bg-body-tertiary">
<header>

```

```

@extends('templates.headers.navigation')

```

```

</header>

<main class="container">
    @yield('content')
</main>

```

```

<footer class="container mt-5">
    <p class="fs-7">Copyright © 2023</p>
</footer>

```

```

<script src="{{ url('./assets/js/bootstrap.bundle.min.js') }}"></script>

```

```

</body>

```

```

</html>

```

Note:

`@extends` [] penanda bahwa akan memanggil file bernama navigation.blade.php yang berada didalam folder `templates/headers`

`@yield` [] menampilkan tag html yang telah ditandai sebagai bentuk komponen main static

`{{ url() }}` [] `url()`menandakan bentuk dari base url (exp: localhost:8000)

navigation.blade.php

```

<nav class="navbar navbar-expand-lg fixed-top navbar-dark bg-dark" aria-label="Main
navigation">
    <div class="container-fluid">
        <a class="navbar-brand" href="{{ url('/') }}>PWL</a>
        <button class="navbar-toggler p-0 border-0" type="button" id="navbarSideCollapse" aria-
label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>

        <div class="navbar-collapse offcanvas-collapse" id="navbarsExampleDefault">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                    <a class="nav-link active" aria-current="page" href="{{ url('/home') }}>Home</a>
                </li>
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#" data-bs-toggle="dropdown"
                        aria-expanded="false">Profile</a>
                    <ul class="dropdown-menu">
                        <li><a class="dropdown-item" href="{{ url('/profile/identity') }}>Identity</a></li>
                        <li><a class="dropdown-item" href="{{ url('/family') }}>Family</a></li>
                    </ul>
                </li>
            </ul>
            <form class="d-flex" role="search">
                <input class="form-control me-2" type="search" placeholder="Search" aria-
label="Search">
                <button class="btn btn-outline-success" type="submit">Search</button>
            </form>
        </div>
    </div>
</nav>

```

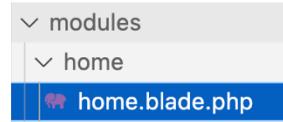
Untuk source code navigasi anda dapat melihat macam bentuknya di:

<https://getbootstrap.com/docs/5.3/components/navbar/#nav>

setelah berhasil membuat blade layout, selanjutnya akan mengatur isi komponen static yang telah ditandai oleh `@yield`

B. Mengatur file main blade view sebagai module

Disini kita akan mengarahkan sebuah alamat website dengan route /home sebagai bagian dari bentuk blade layout. Sehingga website yang kita bangun akan menjadi bentuk web static.



Pada gambar 3.1.2. terdapat folder modules, buatlah folder baru untuk route view home. Dan isi dari file home.blade.php anda dapat mengisinya dengan desain HTML, CSS, dan JS secara bebas.

home.blade.php

```
@extends('templates.layouts')

@section('content')

<div class="d-flex align-items-center p-3 my-3 text-white bg-purple rounded shadow-sm">
    
    <div class="lh-1">
        <h1 class="h6 mb-0 text-white lh-1">Bootstrap</h1>
        <small>Since 2011</small>
    </div>
</div>

<div class="card">
    <div class="card-header">
        <h3 class="card-title">This is a HOME page</h3>
    </div>
    <div class="card-body">
        Welcome folks..
    </div>
</div>

@endsection
```

Notes:

`@extends('templates.layouts')` menandakan bahwa file [home.blade.php](#) ini memanggil file blade layouts

`@section('content')` menandakan bahwa pada file blade layout isi component HTML dibawah fragment section dengan nama content ini akan melakukan ‘append’ di dalam `@yield('content')`.

Selanjutnya yang perlu anda lakukan adalah menambahkan route baru dengan nama home dan route ini memanggil file controller dan mengembalikan bentuk file UI home.blade.php.

Web.php

```

use App\Http\Controllers\HomeController;
use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;

Route::get('/', [HomeController::class, 'index']);
Route::get('/home', [HomeController::class, 'index']);

Route::get('/profile', [ProfileController::class, 'index']);
Route::get('/profile/identity', [ProfileController::class, 'identity']);
Route::get('/profile/family', [ProfileController::class, 'family']);

```

HomeController.php

```

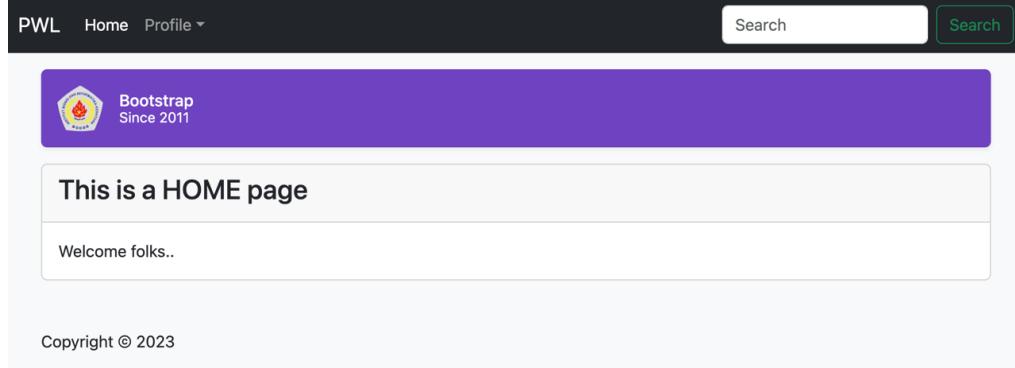
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
class HomeController extends Controller
{
    public function index(){
        return view('modules.home.home');
    }
}

```

Maka output yang didapatkan akan seperti ini:



Gambar Tampilan halaman Home dengan blade layout

2. Blade Fragment

Ada beberapa blade fragment yang akan dibahas pada bagian ini, dari mulai cara merender sebuah data dan melakukan operation statement.

A. Rendering View

Blade merupakan pengaturan tampilan dengan menggunakan HTML markup, dengan penambahan beberapa directive dari Laravel.

Pada contoh sebelumnya terlihat directive pada bagian @section dan @yield. Berikut adalah beberapa markup HTML dalam bentuk blade view:

- Directive `@section` mendefinisikan sebuah bagian (section) dari isi halaman web
- Directive `@yield` digunakan untuk menampilkan isi dari bagian tersebut.

B. Rendering variable

Jika didalam controller ingin mengirimkan sebuah variabel dan ditampilkan ke dalam view blade maka anda dapat menggunakan cara sebagai berikut:

HomeController.php

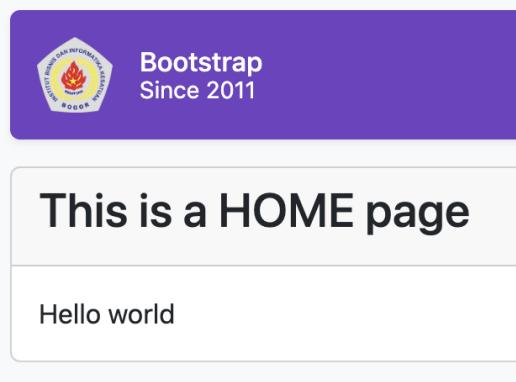
```
<?php  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
  
class HomeController extends Controller  
{  
    public function index()  
    {  
        return view('modules.home.home',['title'=>"Hello world"]);  
    }  
}
```

Pada method view terdapat `["title"=>"Hello world"]` ini adalah contoh pengiriman sebuah variabel dari controller ke dalam blade view. Setiap data yang ingin dikirimkan harus disimpan dalam bentuk ‘key’. Contoh pada script diatas key title memiliki sebuah value dalam bentuk STRING yang berisi tulisan “Hello World”. Sehingga untuk menampilkan key title pada blade view cukup memanggilnya dengan cara: `{{ $title }}`

Contoh pada file home.blade.php:

```
<div class="card">  
    <div class="card-header">  
        <h3 class="card-title">This is a HOME page</h3>  
    </div>  
    <div class="card-body">  
        {{ $title }}  
    </div>  
</div>
```

Maka hasil yang didapatkannya yaitu:



Atau anda juga dapat menggunakan method with() untuk mengirimkan sebuah data kedalam blade view.

```
return view('modules.home.home')->with("title", "Hello world");
```

Lalu jika data yang anda ingin kirimkan dalam bentuk multiple atau lebih dari satu bagaimana mengirimkannya ?

Anda dapat memanfaatkan bentuk array sebagai bagian dari pengiriman data secara multiple dan merendernya dengan menggunakan blade directive.

C. Directive

Selain konsep pewarisan template (blade layout) dan menampilkan data, Blade juga menyediakan struktur kontrol umum PHP, seperti pernyataan bersyarat dan pengulangan.

- Selection: IF

HomeController
<pre>class HomeController extends Controller { 2 references 0 overrides public function index(){ \$data = array(); \$data['title'] = "Sample IF ELSE"; \$data['npm'] = 212310029; return view('modules.home.home')->with("data",\$data); } }</pre>
home.blade.php
<pre><div class="card"> <div class="card-header"> <h3 class="card-title">{{ \$data['title'] }}</h3> </div> <div class="card-body"> <p>NPM {{ \$data['npm'] }} termasuk bilangan @if (\$data['npm'] % 2 === 0) GENAP @else GANJIL @endif </p> </div> </div></pre>



Selection: Switch case

```
<p>NPM {{ $data['npm'] }} termasuk bilangan
  @switch($data['npm'] % 2)
    @case(0)
      <span class="text-primary">GENAP</span>
      @break

    @default
      <span class="text-info">GANJIL</span>
      @break

  @endswitch
</p>
```

- Repetition: FOR

```
<div class="skills">
  <h4 class="text-uppercase">Skill:</h4>
  <div class="d-flex flex-row justify-content-between">
    <p class="text-dark">PHP</p>
    <div>
      @for ($i = 0; $i < 5; $i++)
        <span class="me-1 text-warning">
          | <i class="bi bi-star-fill"></i>
        </span>
      @endfor
    </div>
  </div>

  <div class="d-flex flex-row justify-content-between">
    <p class="text-dark">JAVA</p>
    <div>
      @for ($i = 0; $i < 5; $i++)
        <span class="me-1 text-warning">
          | <i class="bi bi-star{{ ($i > 2) ? '-fill' : '' }}"></i>
        </span>
      @endfor
    </div>
  </div>
</div>
```

Maka output dari script di atas sebagai berikut:

Blade Directive

NPM 212310029 termasuk bilangan **GANJIL**

NPM 212310029 termasuk bilangan **GANJIL**

SKILL:

PHP	★★★★★
JAVA	☆☆☆★★

- Repetition: Foreach
Pengulangan dengan menggunakan FOREACH biasanya hanya diperuntukan jika memiliki nilai dalam bentuk multiple Array.

HomeController.php

```
public function index(){  
    $data = array();  
    $data['title'] = "Blade Directive";  
    $data['npm'] = 212310029;  
    $students[] = array("npm"=>212310004, "name"=>"Muhamad Agus Setiawan");  
    $students[] = array("npm"=>212310029, "name"=>"Muhamad Ilham Fachririzal");  
    $students[] = array("npm"=>212310044, "name"=>"Hana Yulia Rahmah");  
    $students[] = array("npm"=>212310027, "name"=>"MUHAMMAD ASKAH");  
    $students[] = array("npm"=>212310005, "name"=>"ADJIE SYERAFFI RAHMAT");  
    $data['students'] = $students;  
    return view('modules.home.home')->with("data",$data);  
}
```

home.blade.php

```
<div class="card-body">  
    <table class="table">  
        <thead>  
            <tr>  
                <th>No</th>  
                <th>NPM</th>  
                <th>Name</th>  
            </tr>  
        </thead>  
        <tbody>  
            @foreach ($data['students'] as $index => $result)  
            <tr>  
                <td>{{ $index+1 }}</td>  
                <td>{{ $result['npm'] }}</td>  
                <td>{{ $result['name'] }}</td>  
            </tr>  
        @endforeach  
    </tbody>  
    </table>  
</div>
```

Maka output dari script di atas sebagai berikut:

List of Student		
No	NPM	Name
1	212310004	Muhamad Agus Setiawan
2	212310029	Muhamad Ilham Fachririzal
3	212310044	Hana Yulia Rahmah
4	212310027	MUHAMMAD ASKAH
5	212310005	ADJIE SYERAFFI RAHMAT

6.3 Latihan Pembelajaran

1. Jawablah contoh kasus pada section 2.2
2. Jika pada backend memiliki sebuah data array dalam bentuk tabel dibawah ini, bagaimana mengubah bentuk tabel skil akademik menjadi bentuk multiple array ?

Course	Type	Rate
Matematika	Diskrit	4
Matematika	Aljabar Linear	3
Basis Data	DDL	2
Bhs Inggris	Speaker	1

3. Dari soal nomor 2, buatlah modifikasi pada contoh kasus pada gambar 2.3.1. Dimana Ketika mengakses route Profile maka akan menampilkan informasi soal nomor 2 (Route\controller\View), dimana bentuk array tersebut dikirimkan dari Controller dan ditampilkan ke blade profile. Gunakan blade layout pada contoh kasus ini.
4. Dari hasil kode nomor 3, silahkan anda buat tampilan hasil rendernya menjadi seperti gambar dibawah ini:

Course	Type	Rate
Matematika	Diskrit	★★★★★
	Aljabar Linear	★★★★★
Basis Data	DDL	★★★★★
Bhs Inggris	Speaker	★★★★★

BAB 7

DATABASES

7.1 Tujuan Pembelajaran

Mahasiswa memahami, mengelola, dan mengintegrasikan database ke dalam aplikasi web menggunakan framework Laravel

7.2 Dasar Teori

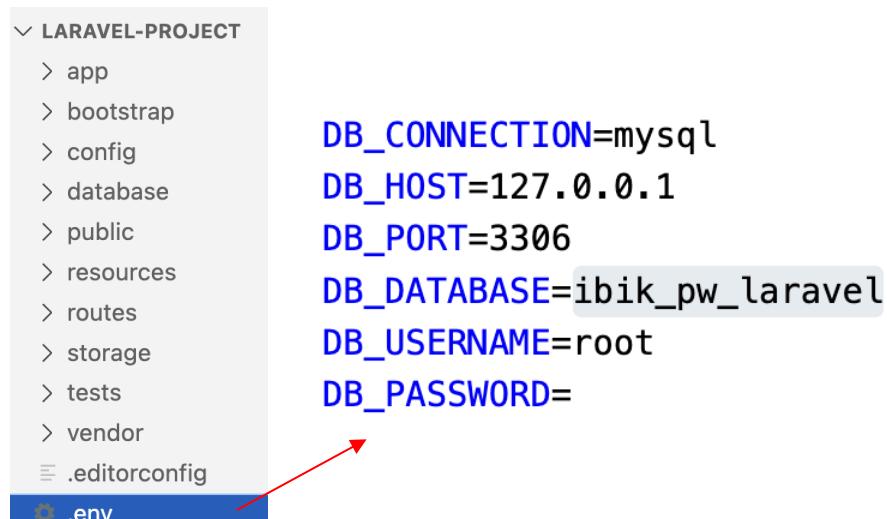
7.2.1 Migration

Migration seperti kontrol versi untuk database, memungkinkan untuk memodifikasi dan membagikan skema database aplikasi. Migrasi biasanya dipasangkan dengan pembuatan skema Laravel lainnya agar dapat membuat skema database aplikasi

Bagian depan Skema Laravel menyediakan dukungan agnostik database untuk membuat dan memanipulasi tabel di semua sistem database yang didukung Laravel.

1. Database configuration

Dalam membangun sebuah koneksi project Laravel dengan database terdapat hal-hal yang perlu disetting terlebih dahulu, pertama buatlah sebuah database bernama `ibik_pw_laravel`, bukalah file `.env` pada project Laravel dan ubahlah settingan environment database sesuai dengan koneksi MYSQL anda:



Gambar Setting database file koneksi `.env`

2. Generate Migration

Untuk membuat sebuah table pada database anda dapat memanfaatkan Teknik migration yang dimiliki oleh Laravel. Dimana anda dapat membuat beberapa factory berdasarkan tabel-tabel database anda. Contoh berikut ini adalah akan membuat sebuah table bernama `Products`. Untuk membuat sebuah table `Products` hal yang perlu dilakukan ialah membuat file migration, bukalah project Laravel anda dengan terminal dan memasukan syntax di bawah ini untuk membuat file migration table:

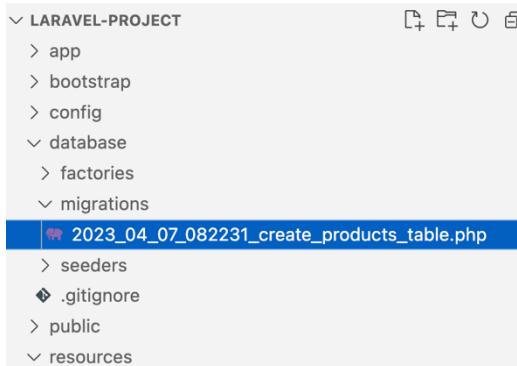
```
php artisan make:migration create_products_table
```

```

febryfairuz@Febrys-MacBook-Air laravel-project % php artisan make:migration create_products_table
[INFO] Migration [database/migrations/2023_04_07_082231_create_products_table.php] created successfully.
febryfairuz@Febrys-MacBook-Air laravel-project %

```

Setelah anda memasukan syntax migration tersebut maka akan menginformasikan letak directory path file migrations bernama *create_products_table*.



Gambar Directory path file migrations

Bukalah file migration yang telah anda buat dan silahkan anda buat field-field apa saja yang dibutuhkan, contohnya yaitu seperti dibawah ini:

```

public function up(): void
{
    Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->string('name',50);
        $table->text('description');
        $table->float('price',8,2); //=>bernilai DOUBLE
        $table->tinyInteger('is_active'); //=>tiny int
        $table->timestamps();
    });
}

```

Jika melihat script di atas menandakan bahwa table product akan memiliki beberapa field seperti:

products
id: bigint(20)
name: varchar(50)
description: text
price: double(8, 2)
is_active: tinyint(4)
created_at: timestamp
updated_at: timestamp

Field	Type	Value
id	bigint	20
name	varchar	50
description	text	
price	double	8,2
is active	tinyint	4

A. Tipe Data

Beberapa Tipe data yang dapat digunakan untuk mendeklarasikan field pada tabel:

```

# bigIncrements()           # decimal()
# bigInteger()              # enum()
# boolean()                 # float()
# char()                    # id()
# dateTime()                # increments()
# date()                    # integer()

```

B. Menjalankan Migration

Setelah menentukan field apa saja yang akan dibuat untuk mengeksekusi file migration anda dapat memasukan syntax dibawah ini pada terminal project Laravel anda:

```
php artisan migrate
```

Namun jika ada perubahan pada table anda, setelah merubah file migrasinya lalu jalankan perintah ini:

```
php artisan migrate:refresh
```

Silahkan anda cek kedalam database *ibik_pw_laravel* apakah table *products* yang telah anda buat dengan migration berhasil dibuat atau tidak.

7.2.2 Query Builder

Setelah berhasil membuat sebuah table products, contoh kasus kali ini akan membuat CRUD terhadap data-data yang akan dikelola oleh table products. Pada chapter sebelumnya, telah diterangkan bahwa konsep kerja dari PHP Framework adalah MVC. Dimana file Model akan mengeksekusi logika database, file Controller akan mengeksekusi logika business process dan View pada blade akan menampilkan hasil yang telah dikelola di model dan controller dalam bentuk UI.

No	Name	Description	Price	Status	Action
1	ABC Saus Asam Manis 195 ml - Twin Pack	2 botol ABC Saus Asam	Rp 90000	Yes	Delete Edit View
2	Kecap Kerbau	Dari kedele langsung	Rp 8000	Yes	Delete Edit View
3	Sendal Jepti	Bahan dari kulit ular	Rp 35000	Yes	Delete Edit View

Gambar 2. Tampilan project Laravel dengan CRUD

Setelah membuat file migration buatlah file Controller dan Model untuk Product dengan menggunakan syntax php artisan:

```
php artisan make:controller ProductsController --resource --model=Products
```

Syntax `php artisan` diatas akan membuat sebuah file controller pada `./app/Http/controllers/` bernama `ProductsController` dan file model pada `./app/Http/models/` bernama `Products` yang berada pada directory path project Laravel.

Untuk mengimplementasikan Query Builder pada Laravel, terdapat dua buah cara yaitu dengan *Eloquent Factory Model* dan PDO parameters. Berikut ini akan memberikan contoh dari kedua Teknik query builder pada Laravel untuk melakukan CRUD.

1. Create and Retrieve

Pada section ini akan mencontohkan bagaimana melakukan insert dan retrieve data kedalam sebuah database dengan konsep MVC dan Teknik query builder dengan *Eloquent Factory Model*.

Model: Products.php

```
<?php  
  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Support\Facades\DB;  
  
class Products extends Model  
{  
    use HasFactory;  
  
    protected $table = 'products';  
  
    protected $fillable = [  
        'id', 'name', 'description','price','is_active'  
    ];  
  
    public function storedData($data){  
        $results = Products::create($data);  
        return $results;  
    }  
}
```

Controller: ProductsController.php

```
<?php  
  
namespace App\Http\Controllers;  
  
use App\Http\Requests\UpdateProductsRequest;  
use App\Models\Products;  
use Illuminate\Http\Request;  
  
class ProductsController extends Controller  
{  
  
    public function index()  
    {  
        $products = Products::latest()->paginate(5);  
        return view('modules.master-data.products.index',compact('products'))  
            ->with('i',(request()->input('page', 1) - 1) * 5);  
    }  
}
```

Pada sebuah fungsi bernama `index` pada `ProductsController` diatas, variable `$products` menyimpan sebuah *Eloquent Factory* dari model products:

`Products::latest()->paginate(5);` mengambil record pada database dari urutan 5 terbaru (pengambilan data secara descending).

Lalu nilai variable `$products` dikirimkan ke file blade yang berada di `module.master-data.products.index`. Fungsi paginate merupakan bawaan eloquent factory dimana pada ui blade index sudah membentuk sebuah data dalam bentuk pagination dengan maximum nilai data sebanyak 5 buah.

Route: Web.php

```
<?php
```

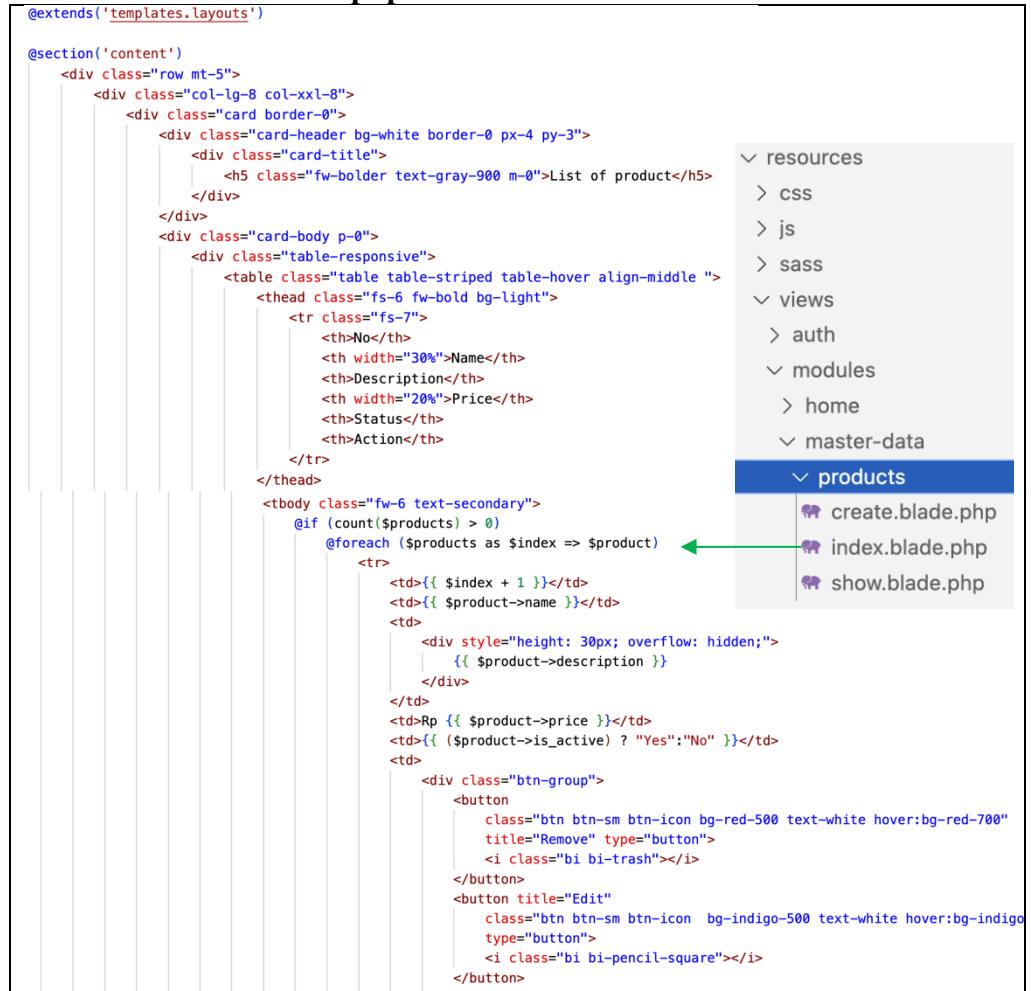
```
use App\Http\Controllers\HomeController;
use App\Http\Controllers\ProductsController;
use Illuminate\Support\Facades\Route;

Route::get('/', [HomeController::class, 'index']);
Route::get('/home', [HomeController::class, 'index']);

Route::controller(ProductsController::class) -> group(function () {
    Route::get('/master-data/products', 'index') -> name('m_products');
    Route::post('/master-data/products', 'store');
    Route::get('/master-data/products/{id}', 'show') -> name('m_products_detail');
    Route::get('/master-data/products/edit/{id}', 'edit') -> name('m_products_edit');
    Route::get('/master-data/products/remove/{id}', 'destroy') -> name('m_products_remove');
});

});
```

Blade View: index.blade.php



```
@extends('templates.layouts')

@section('content')
    <div class="row mt-5">
        <div class="col-lg-8 col-xxl-8">
            <div class="card border-0">
                <div class="card-header bg-white border-0 px-4 py-3">
                    <div class="card-title">
                        | <h5 class="fw-bolder text-gray-900 m-0">List of product</h5>
                    </div>
                </div>
                <div class="card-body p-0">
                    <div class="table-responsive">
                        <table class="table table-striped table-hover align-middle">
                            <thead class="fs-6 fw-bold bg-light">
                                <tr class="fs-7">
                                    <th>No</th>
                                    <th width="30%">Name</th>
                                    <th>Description</th>
                                    <th width="20%">Price</th>
                                    <th>Status</th>
                                    <th>Action</th>
                                </tr>
                            </thead>
                            <tbody class="fw-6 text-secondary">
                                @if (count($products) > 0)
                                    @foreach ($products as $index => $product)
                                        <tr>
                                            <td>{{ $index + 1 }}</td>
                                            <td>{{ $product->name }}</td>
                                            <td>
                                                <div style="height: 30px; overflow: hidden;">
                                                    | {{ $product->description }}</div>
                                            </td>
                                            <td>Rp {{ $product->price }}</td>
                                            <td>{{ ($product->is_active) ? "Yes" : "No" }}</td>
                                            <td>
                                                <div class="btn-group">
                                                    <button
                                                        class="btn btn-sm btn-icon bg-red-500 text-white hover:bg-red-700"
                                                        title="Remove" type="button">
                                                        <i class="bi bi-trash"></i>
                                                    </button>
                                                    <button title="Edit"
                                                        class="btn btn-sm btn-icon bg-indigo-500 text-white hover:bg-indigo-700"
                                                        type="button">
                                                        <i class="bi bi-pencil-square"></i>
                                                    </button>
                                                </div>
                                            </td>
                                        </tr>
                                    @endforeach
                                @endif
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

        <a title="Detail"
           class="btn btn-sm btn-icon bg-orange-500 text-white hover:bg-orange-700"
           type="button">
           <i class="bi bi-arrow-right-square"></i>
        </a>
      </div>
    </td>
  </tr>
  @endif
@else
  <tr>
    <td colspan="6">No record found</td>
  </tr>
@endif
</tbody>
</table>
{{ $products->links() }}

```

Paginate bawaan Laravel 10 dengan menggunakan CSS TailWind

Blade View: create.blade.php

```

<div class="card border-0">
  <div class="card-header bg-white border-0 px-4 py-3">
    <h5>Form Product</h5>
  </div>
  <div class="card-body pt-0">
    @if ($errors->any())
      <div class="alert alert-danger mb-5">
        <strong>Error!</strong> <br>
        <ul>
          @foreach ($errors->all() as $error)
            <li>{{ $error }}</li>
          @endforeach
        </ul>
      </div>
    @endif
    @if ($message = Session::get('success'))
      <div class="alert alert-success mb-5">
        <p>{{ $message }}</p>
      </div>
    @endif
    <form action="{{ url('/master-data/products') }}" method="post" autocomplete="off" id="form-product">
      @csrf
      <div class="mb-3">
        <label class="form-label">Name</label>
        <input type="hidden" class="form-control id" name="id" />
        <input type="text" class="form-control name" name="name" />
        <div class="form-text text-danger"></div>
      </div>
      <div class="mb-3">
        <label class="form-label">Description</label>
        <textarea class="form-control description" name="description"></textarea>
        <div class="form-text text-danger"></div>
      </div>
      <div class="mb-3">
        <label class="form-label">Price</label>
        <div class="input-group flexnowrap">
          <span class="input-group-text" id="addon-wrapping">Rp</span>
          <input type="text" class="form-control price" name="price" />
        </div>
        <div class="form-text text-danger"></div>
      </div>
    </form>
  </div>

```

```


<label class="form-label">Active</label>
    <div class="d-flex justify-content-start align-items-center">
        <label class="me-2">
            | <input type="radio" class="is_active_Y" name="is_active" value="1" /> Yes
        </label>
        <label class="me-2">
            | <input type="radio" class="is_active_N" name="is_active" value="0" /> No
        </label>
    </div>
    <div class="form-text text-danger"></div>
    <div class="text-end">
        <button class="py-2 px-4 btn-danger" type="reset">Clear</button>
        <button class="py-2 px-4 btn-primary" type="submit">Save changes</button>
    </div>
</form>
</div>


```

Output dari script diatas akan menghasilkan program seperti dibawah ini dengan beberapa kondisi, seperti validasi dan informasi data berhasil atau gagal dieksekusi untuk melakukan inserting ataupun menampilkan data pada database.

Form Product

Error!
 The name field is required.
 The description field is required.
 The price field is required.

Name

Description

Price

 Rp 45000

Active

 Yes No

Clear **Save changes**

Form Product

Name

Description

Price

 Rp 45000

Active

 Yes No

Clear **Save changes**

Gambar 2.1.1. Jika salah satu form isian field adalah kosong

Gambar 2.1.2. Tampilan mengisi sebuah data baru

The screenshot shows a web interface for managing products. On the left, there is a table titled "List of product" displaying five rows of product data. Each row includes columns for No, Name, Description, Price, Status, and Action (with icons for edit, delete, and details). Below the table, it says "Showing 1 to 5 of 6 results" and has a navigation bar with links for <, 1, 2, and >. On the right, there is a "Form Product" section with fields for Name, Description, Price (set to Rp), and Active status (with radio buttons for Yes and No). A success message "Product created successfully." is displayed above the form. At the bottom right of the form are "Clear" and "Save changes" buttons.

Gambar 2.1.3. Tampilan informasi berhasil menambahkan data baru

2. Retrieve 1 Data

Pada section ini akan menampilkan data dalam jumlah 1 atau mencari data berdasarkan kondisi. Disini akan mencontohkan penggunaan query builder dalam bentuk PDO parameters. Dimana untuk mengakses sebuah tabel harus didefinisikan dengan menggunakan `DB::table('table_name')`.

Model: Products.php

```
public function getByCondition($condition){
    $results = DB::table($this->table)->where($condition);
    return $results;
}
```

Controller: ProductsController.php

```
public function show(Request $request)
{
    $products = new Products();
    $data['product'] = $products->getByCondition(array('id'=>$request->id))->first();

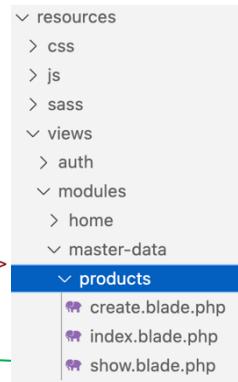
    return view('modules.master-data.products.show', $data);
}
```

Blade View: show.blade.php

```

@extends('templates.layouts')
@section('content')
    <div class="row">
        <div class="col-lg-6 col-xxl-6">
            <div class="info">
                <p class="text-4xl mb-3">{{ $product->name }}</p>
                <p class="text-4xl mb-2">Rp {{ $product->price }}</p>
                <div class="border-top border-bottom p-2 mt-3">
                    <span class="text-1xl">Description</span>
                </div>
                <div class="description p-2">{{ $product->description }}</div>
            </div>
        </div>
        <div class="col-lg-2 col-xxl-2">
            <div class="border rounded bg-white p-3">
                <div class="mb-3">
                    <p class="text-dark text-sm mb-2">Created at <br/>{{ $product->created_at }}</p>
                    <p class="text-dark text-sm">Last Updated at <br/>{{ $product->updated_at }}</p>
                </div>
                <div class="d-grid gap-2">
                    <a href="" class="btn btn-sm btn-warning">Edit</a>
                    <a href="" class="btn btn-sm btn-danger">Remove</a>
                </div>
            </div>
        </div>
    </div>
@endsection

```



Jika melihat gambar 2.1.3 pada table of product terdapat 3 buah button di kolom Action, jika mengklik tombol ketiga (button berwarna orange) yaitu detail maka akan dialihkan ke halaman dengan route end-point:

<http://localhost:8000/master-data/products/3>

jika melihat file route pada *Web.php* tipe link diatas akan diarahkan ke *ProductsController* dengan fungsi bernama *show()*.

ABC Saus Asam Manis 195 ml - Twin Pack

Rp 90000

Description

2 botol ABC Saus Asam Manis 195 ml Saus Spesial ABC mempunyai cita rasa yang lezat, membuat hidangan sehari-hari menjadi hidangan spesial. Lengkapi masakan Anda dengan lezatnya cita rasa ABC Saus Spesial dan buat hidangan sehari-hari menjadi hidangan spesial yang tidak kalah lezatnya dari hidangan restoran.

Tersedia dalam 4 varian: Saus Asam Manis, Saus Tiram, Saus Inggris, dan Minyak Wijen. Tersedia dalam 4 varian: Saus Asam Manis, Saus Tiram, Saus Inggris, dan Minyak Wijen. Pas untuk memasak berbagai macam makanan. Rasa yang cocok untuk lidah orang Indonesia. Air, gula, pasta tomat, garam, konsentrasi nanas, bawang, cuka, pati termodifikasi, pengawet (natrium benzoat dan natrium metabisulfit), penstabil nabati, rempah-rempah, perisa sintetik.

Gambar 2.2. Tampilan halaman detail

3. Update dan Delete

Pada contoh section kali ini akan mengkombinasikan blade view dengan bentuk render data dengan JAVASCRIPT untuk melakukan UPDATE dan DELETE.

Model: Products.php

```

public function updatedData($data){
    $isExist = $this->getByCondition(array('id'=>$data['id']))->first();
    if(!empty($isExist)){
        unset($data['_token']);
        $results = DB::table($this->table)->where(array('id'=>$data['id']))->update($data);
        return $results;
    }else{
        return null;
    }
}

public function removeByCondition($condition){
    $results = Products::where($condition)->delete();
    return $results;
}

```

Query Builder PDO

Query Builder Eloquent Factory

Controller: ProductsController.php

```

public function store(Request $request)
{
    $products = new Products();
    if(!empty($request->id)){
        $request->validate([
            'id' => 'required',
            'name' => 'required',
            'description' => 'required',
            'price'=> 'required'
        ]);

        $results = $products->updatedData($request->all());
        return redirect()->route('m_products')->with('success',
            ($results) ? 'Product saved.' : 'Product failed save.');
    }else{
        $request->validate([
            'name' => 'required',
            'description' => 'required',
            'price'=> 'required'
        ]);
        $results = $products->storedData($request->all());
        return redirect()->route('m_products')->with('success',
            ($results)? 'Product created successfully.': 'Product failed save.');
    }
}

```

Pada code sebelumnya di fase create, function store hanya menyimpan logika transaksi insert query saja. Namun kali ini dilakukan perubahan, jika data yang dilempar memiliki key bernama *id*, maka akan mengeksekusi logika transaksi update query sedangkan sebaliknya maka akan mengeksekusi logika transaksi insert query.

Pengecekan ini terjadi dikarenakan jika ingin melakukan update query data, memerlukan sebuah kondisi statement, pada contoh update disini akan mempergunakan key *id* sebagai salah satu update kondisi statement.

Blade View: index.blade.php

```
@extends('templates.layouts')
@section('content')

...
<div class="btn-group">
    <button
        class="btn btn-sm btn-icon bg-red-500 text-white hover:bg-red-700"
        title="Remove" type="button"
        onclick="RemoveItem({{ $product->id }})">
        <i class="bi bi-trash"></i>
    </button>
    <button title="Edit"
        class="btn btn-sm btn-icon bg-indigo-500 text-white hover:bg-indigo-700"
        type="button" onclick="EditItem({{ $product->id }})">
        <i class="bi bi-pencil-square"></i>
    </button>
    <a href="{{ route('m_products_detail', $product->id) }}" title="Detail"
        class="btn btn-sm btn-icon bg-orange-500 text-white hover:bg-orange-700"
        type="button">
        <i class="bi bi-arrow-right-square"></i>
    </a>
</div>
...
<script>
    const RemoveItem = (id) => {
        if (confirm("Are you sure wants to remove this item ?")) {
            const xmlhttp = new XMLHttpRequest();
            xmlhttp.onload = function() {
                var data = JSON.parse(this.response);
                alert(data.message);
                window.location.href = "{{ route('m_products') }}";
            }
            xmlhttp.open("GET", "{{ url('') }}/master-data/products/remove/" + id);
            xmlhttp.send();
        }
    }
    const EditItem = (id) => {
        var targetDiv = document.getElementById("form-product");
        let id_ = targetDiv.getElementsByClassName("id")[0];
        let name = targetDiv.getElementsByClassName("name")[0];
        let desc = targetDiv.getElementsByClassName("description")[0];
        let price = targetDiv.getElementsByClassName("price")[0];
        let is_active_Y = targetDiv.getElementsByClassName("is_active_Y")[0];
        let is_active_N = targetDiv.getElementsByClassName("is_active_N")[0];

        const xmlhttp = new XMLHttpRequest();
        xmlhttp.onload = function() {
            var data = JSON.parse(this.response);
            id_.value = data.id;
            name.value = data.name;
            desc.value = data.description;
            price.value = data.price;
            if(data.is_active === 1){
                is_active_Y.checked = true;
            }else{
                is_active_N.checked = true;
            }
        }
        xmlhttp.open("GET", "{{ url('') }}/master-data/products/edit/" + id);
        xmlhttp.send();
    }
</script>
@endsection
```

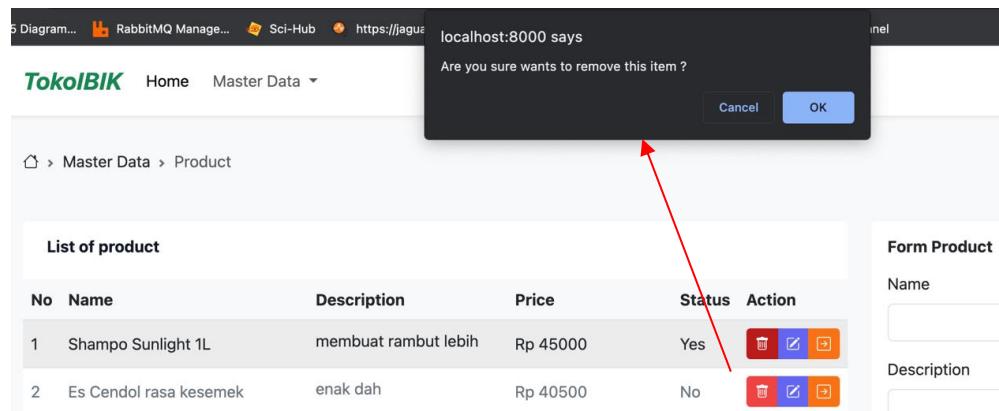
Hanya menambahkan event click

Menambahkan route link pada tombol

Menambahkan function action dari event click

Maka output dari proses transaksi logical diatas untuk UPDATE dan DELETE sebagai berikut:

Gambar 2.3. Tampilan update salah satu data.



Gambar 2.4. Tampilan delete salah satu data.

7.3 Latihan Pembelajaran

1. Buatlah package baru bernama pembelajaran-6
2. Buatlah sebuah CRUD untuk data **Users**, dimana pada tabelnya memiliki structure dibawah ini:

Field	Tipe	Value
id*	bigint	20
email	varchar	20
fullname	varchar	100
address	text	8,2
birthdate	date	4
gender	enum	('M','F')
phone	varchar	14

3. Berdasarkan soal nomor 2, buatlah skema table **Users** dengan menggunakan MIGRATION.
4. Setelah berhasil membuat table dengan soal nomor 3, buatlah Tampilan untuk melakukan maintenance data Users dalam bentuk CRUD dengan memanfaatkan kerangka kerja MVC

BAB 8

AUTHENTICATIONS

8.1 Tujuan Pembelajaran

Mahasiswa dapat menerapkan autentikasi yang kuat, dengan memastikan bahwa hanya pengguna yang sah yang memiliki akses ke sumber daya dan fitur tertentu di dalam aplikasi. Laravel menyediakan sistem autentikasi yang terintegrasi dan memiliki fitur-fitur keamanan yang bawaan.

8.2 Dasar Teori

Pada bagian ini akan membuat sebuah authentication pada sebuah website, namun sebelum mengolah authentication atau hak akses terhadap sebuah website, hal yang perlu dilakukan ialah menyiapkan sebuah halaman Sign In dan Database table USERS. Pada pertemuan sebelumnya telah dibahas bahwasannya ketika pertama kali membuat sebuah project Laravel, pada folder migration telah mengenerate beberapa table bawaan seperti pada gambar dibawah ini:



Gambar 8.2.a Directory folder database

Dari gambar tersebut, salah satu file bernama create_users_table.php merupakan table yang digunakan untuk menyimpan data users. Dimana file migrasi tersebut jika anda generate maka akan menciptakan sebuah table pada database terpilih sebagai berikut:

users	
🔑	id: bigint UNSIGNED
👤	name: varchar(255)
✉️	email: varchar(255)
	email_verified_at: timestamp
	password: varchar(255)
	remember_token: varchar(100)
	created_at: timestamp
	updated_at: timestamp

Gambar 8.2.b Table users bawaan Laravel

Untuk melakukan proses authentication kita dapat menggunakan atau memanfaatkan table bawaan tersebut, namun jika anda ingin melakukan pembaharuan data terhadap jenis

isian kolumn pada table users, silakan lakukan pembaharuan di dalam file tersebut. Setelah melakukan generate database table users, pada contoh kali ini akan membuat sebuah halaman Sign In dengan isian email dan password. Sebelum itu, pastikan table users telah memiliki beberapa data isian sehingga dapat dipergunakan untuk melakukan logical transaksi Sign In dengan mengecek kolumn email dan password.

id	name	email	password
1	M Subhan R A	212310020@student.ibik.ac.id	\$2y\$10\$J4xtXxD/JTbkSxEuKo94LeTDE.t0duOidqGkhzQAvv4Zy2.ArQrdC
2	Michael Teddy S	212310047@student.ibik.ac.id	password123

Gambar 8.2.c Browse table users

Selanjutnya ialah membuat sebuah route baru bernama `/signin` dimana alamat ini akan menunjukan halaman form isian Sign In. Halaman tersebut diakses melalui sebuah class controller bernama `Authentications`.

Routes - Web:

```
Route::controller(Authentication::class)->group(function () {
    Route::get('/signin', 'index')->name('signin');
});
```

Controllers - Authentication:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;

class Authentication extends Controller{
    public function index(){
        return view('authentications.signin');
    }
}
```

Blade – Sign In:

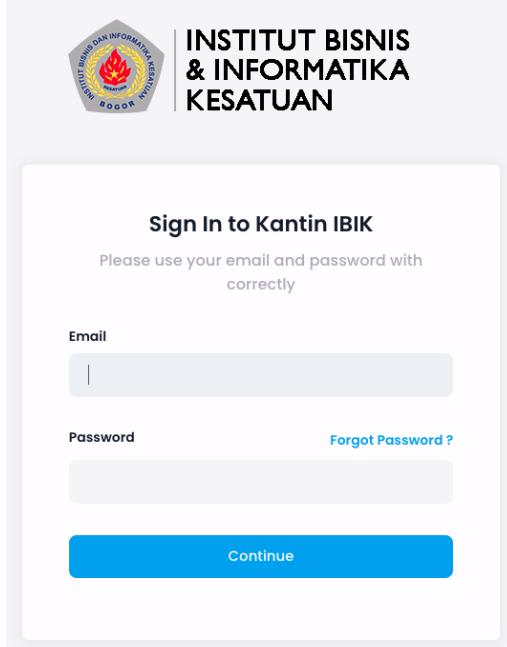
```
<form action="{{ url('/signin') }}" method="post" >
<div>
<h1>Sign In to Kantin IBIK</h1>
<div>Please use your email and password with correctly</div>
</div>
<div>
<label for="email">Email</label>
<input type="text" name="email" />
</div>
<div>
<label for="password">Password</label>
<a href="#">Forgot Password ?</a>
</div>
<input type="password" name="password" />
</div>
```

```

<button type="submit">Continue</button>
</div>
</form>

```

Output – Sign In:



Gambar 8.2.d Halaman Sign In

8.2.1 Facades

Facades merupakan library milik Laravel yang bekerja untuk mengatur dan menyimpan data atau informasi kedalam bentuk session. Pada contoh kasus saat ini kita akan membangun sebuah halaman website, dimana website tersebut memiliki dua buah authorize yaitu Public dan Private. Halaman public dapat diakses jika pengguna belum melakukan *Sign In*, sedangkan halaman Private dapat diakses jika pengguna sudah melakukan *Sign In* kedalam aplikasi websitenya. Istilah *authorize* tersebut pada *Facades* disebut dengan *auth* dan *guest*. Dimana kata *auth* diperuntukan bagi pengguna yang sudah pernah *sign in*, sedangkan *guest* diartikan sebagai pengguna yang belum melakukan *Sign In* kedalam aplikasi. Library ini dapat diakses dengan memanggilnya seperti berikut:

```
use Illuminate\Support\Facades\Auth;
```

8.2.2 Implement Facades

A. Sign In

- 1) Menambahkan library Facades kedalam Controller Authentication:

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;

```

```

use Illuminate\Support\Facades\Auth;

class Authentication extends Controller
{
    ...
}

```

- 2) Membuat function pada controller dengan tipe request POST untuk mengolah transaksi Sign In, diberi nama authenticate:

```

class Authentication extends Controller {
    public function authenticate(Request $request):RedirectResponse{
        //do logical here for auth
    }
}

```

`Request`, digunakan untuk mengambil nilai parameter yang dikirimkan oleh HTTP. Setiap parameter yang dikirimkan oleh HTTP akan diterima dalam bentuk *object*. Oleh karenanya, jika ingin memanggil key dari parameter yang dikirimkan anda dapat menggunakan:

`$request->name_of_key` atau `request()->name_of_key`

- 3) Memperbarui routes dengan menambahkan tipe request POST untuk alamat `/signin` dengan mengakses function `authenticate`:

```

Route::controller(Authentication::class)->group(function () {
    Route::get('/signin', 'index')->name('signin');
    Route::post('/signin', 'authenticate');
});

```

- 4) Membuat sebuah validasi pada form Sign In dengan mengecek isian email dan password. Dimana kedua isian tersebut wajib diisi, dan mengecek apakah isian email memiliki bentuk valid (name@domain_name_system.com):

- Pada file `signin.blade`, didalam tag `<form>` tambahkan *cross-site request forgery* @csrf. CSRF ini selalu dipasang jika memiliki sebuah form, hal ini diperlukan untuk merequest Token agar terhindar dari request http *anonymous*:

```

<form action="{{ url('/signin') }}" method="post" >
    @csrf
    ...
    <div>
        <label for="email">Email</label>
        <input type="text"
            name="email" required
            value="{{ old('email') }}" />
    </div>
    <div>
        <div>
            <label for="password">Password</label>
            <a href="#">Forgot Password ?</a>
        </div>
        <input type="password" name="password" required />
    </div>

```

```
</div>
...
</form>
```

Penambahan properties `required` pada tag elemen `<input>` diperlukan untuk menjaga validasi dari sisi frontend atau browser.

- Pada file controller

Pada function authentication kita akan memasang validasi bawaan Laravel dengan menggunakan function bernama `validate()`:

```
class Authentication extends Controller {
    public function authenticate(Request $request):RedirectResponse{
        $credentials = $request->validate([
            'email' => ['required', 'email:dns'],
            'password' => ['required'],
        ]);
    }
}
```

Setelah menambahkan code diatas coba anda jalankan dan test berdasarkan kondisi dibawah ini:

- a. Salah satu isian tidak terisi
- b. Kedua isian tidak terisi
- c. Mengisi isian email dengan format asal (bukan bentuk email)

- 5) Mengecek kedalam table users berdasarkan isian email dan password. Pada tahap ini kita akan memanfaatkan *Facades* untuk menyimpan session jika kondisi pengecekan bernilai *true*. Sedangkan jika nilai salah akan mengembalikan *flash message*:

```
class Authentication extends Controller {
    public function authenticate(Request $request):RedirectResponse{
        $credentials = $request->validate([
            'email' => ['required', 'email:dns'],
            'password' => ['required'],
        ]);
        //start: jika email dan password benar
        if (Auth::attempt($credentials)) {
            $request->session()->regenerate();
            return redirect()->intended('dashboard');
        }
        //end of: jika email dan password benar

        return back()->with('loginerror','The provided credentials do not match our records.');
    }
}
```

Pada code diatas code `$request->session()->regenerate();` digunakan untuk melakukan request ulang token agar tidak mengalami *session fixation*.

- 6) Menampilkan pesan error Facades dan flash message. Pada file *signin.blade.php* tambahkan code dibawah ini untuk menampilkan pesan error email dari validate():

```
<div>
  <label for="email">Email</label>
  <input type="text"
    name="email" required
    value="{{ old('email') }}" />

  @error('email')
  <div> {{ $message }} </div>
  @enderror
</div>
```

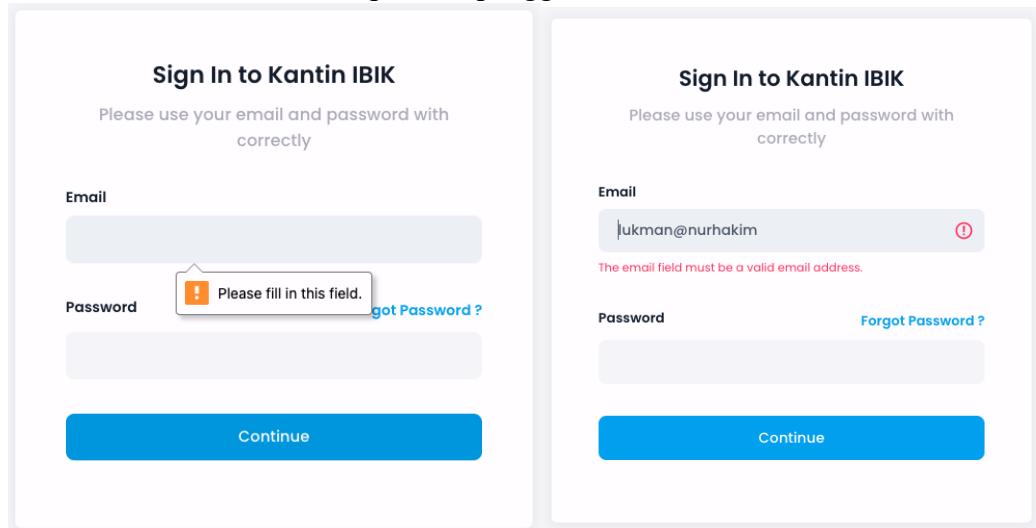
Dan untuk memanggil flash message tambahkan code dibawah ini:

```
<form action="{{ url('/signin') }}" method="post" >
...
@if(session()->has('loginerror'))
<div role="alert">
  {{ session('loginerror') }}
  <button type="button" aria-label="Close"></button>
</div>
@endif
...
</form>
```

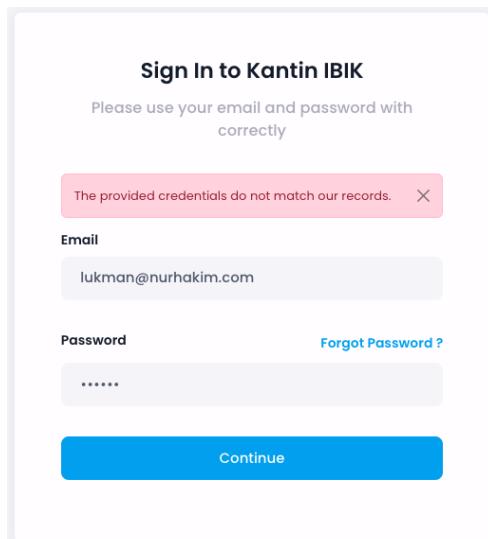
- 7) Pada file RouteServiceProvider.php yang berada di dalam folder *.app/Providers/*, ubahlah code menjadi:

```
//public const HOME = '/home';
public const HOME = '/';
```

Berikut ini adalah contoh output dari penggunaan Facades:



Jika isian form email atau password kosong dan jika isian email memiliki bentuk yang tidak sesuai forman *Domain Name System*.



Jika kondisi pengguna tidak terdaftar pada database maka akan menampilkan flash message yang dikirimkan dari `loginerror`. Sedangkan jika mengisi sesuai dengan yang ada pada database maka aplikasi akan meredirect ke halaman dashboard, sesuai dengan instruksi syntax yang ada pada `authenticate()`.

B. Sign Out

Setelah berhasil membuat sebuah fitur sign in, maka hal selanjutnya ialah membuat fitur sign out. Fitur ini dibuat untuk menghapus atau menghilangkan semua session facades yang ada pada browser berdasarkan alamat dari website tersebut.

1. Membuat *route* baru untuk alamat `/signout` dengan tipe request GET:

```
Route::get('/signout', [Authentication::class, 'signout']);
```

2. Pada class controllers Authentication buatlah function bernama `signout()`:

```
class Authentication extends Controller {
    public function authenticate(Request $request):RedirectResponse{
        ...
        function signout(Request $request):RedirectResponse{
            Auth::logout();
            $request->session()->invalidate();
            $request->session()->regenerateToken();
            return redirect('/signin');
        }
    }
}
```

C. Middleware

Mekanisme ini biasa digunakan untuk melakukan verifikasi terhadap data pengguna untuk mengetahui `authenticate` pada websitenya. Misalnya, terdapat seorang pengguna yang belum terautentikasi pada website, maka system akan meredirect ke halaman tertentu atau halaman yang sudah disiapkan. Sebagai contoh jika belum sign in maka tombol Sign In akan muncul. Namun jika sudah Sign In tombol menjadi bentuk Sign Out.

Pada bagian sebelumnya telah disinggung mengenai auth dan guest, auth merupakan action statement yang menunjukkan status pengguna yang telah sign in atau mengidentifikasi sebagai Private konten. Sedangkan *action statement guest*, digunakan untuk menandai seorang pengunjung yang belum melakukan sign in kedalam system atau diidentifikasi sebagai Public konten.

Berikut ini adalah contoh penggunaan auth dan guest pada file view blade, dimana disini akan membuat sebuah tampilan dengan kondisi, jika sudah pernah sign in maka muncul button Sign Out. Sedangkan yang belum pernah sign in maka akan menampilkan button Sign In.

- Retrive Condition Middleware pada blade

Statement `@guest` dimana tag ini hanya berlaku untuk public area:

```
@guest
<a href="{{ url('signout') }}">Sign Out</a>
@endguest
```

Sedangkan `@auth` diperuntukan bagi private area:

```
@auth
<a href="{{ url(signin) }}">Sign In</a>
@endauth
```

Atau dapat juga dilakukan dengan menggunakan statement `@else` pada statement `@auth`:

```
@auth
<a href="{{ url(signin) }}">Sign In</a>
@else
<a href="{{ url('signout') }}">Sign Out</a>
@endauth
```

- Retrive Value Facades

Jika anda ingin memanggil identitas dari facades untuk menampilkan informasi apa saja yang disimpan dan ditampilkan pada UI website, anda dapat menggunakan format sebagai berikut:

```
{{ auth()->user()->key }}
```

auth() → menunjukkan function private data

user() → menunjukkan nama class model

key → menunjukkan nama key pada database

Contoh:

Jika ingin menampilkan informasi nama dari pengunjung yang telah sign in, seperti pada gambar dibawah ini:

Hai,
M Subhan R A



Berdasarkan gambar 1.3, diketahui nama key untuk menampilkan nama ialah ‘name’. Maka penggunaannya sebagai berikut:

```
Hai, {{ auth()->user()->name }}
```

Selain menggunakan auth dan guest pada blade, kita juga dapat gunakan kedua fitur tersebut untuk membatasi hak akses pada halaman web, dengan cara memberikan menginisialisasi pada route dari masing-masing alamat. Berikut adalah contoh penggunaan middleware pada route:

```
Route::get('/', function () {
    return view('welcome');
})->middleware('auth');

Route::group(['middleware' => 'guest'], function(){
    Route::get('home', function() {} );
    Route::get('about', function() {} );
    Route::get('sign-in', function() {} );
});

Route::group(['prefix' => 'admin', 'middleware' => 'auth'], function(){
    Route::get('dashboard', function() {} );
    Route::get('report', function() {} );
});

Route::get('/dashboard', function () {
    return view('private.dashboard.index');
})->middleware('auth', 'admin');
```

8.3 Latihan Pembelajaran

1. Buatlah alamat baru bernama /sign-up dimana halaman tersebut berisi form pendaftaran yang berisi: name, email dan password. Merujuk pada gambar 1.3.
2. Buatlah logika validasi terhadap form isian soal nomor 1, dengan kondisi:
 - Name, email, dan password tidak boleh kosong.
 - Pengisian email harus mengikuti format DNS.
 - Jumlah maximum name yang terinput sebanyak 100 character, dan minimum 10 character.
 - Jumlah minimum password sebanyak 8 digit, dan maximum 12 digit.
3. Buatlah proses transaksi penginputan data-data tersebut kedalam database dengan menggunakan *eloquent*.
4. Gunakan encrypted data untuk memasukan password.
5. Jika sudah berhasil mengisi kedalam database, alikan ke halaman /signin.

DAFTAR PUSTAKA

- Dorucher, David, 2021. HTML & CSS QuickStart Guide. ClydeBank Media LLC.
- Robbins, Jennifer. 2018. Learning Web Design. O'Reilly Media, Inc.
- Meloni, Julie, 2018. HTML, CSS, and JavaScript All in One, Sams Teach Yourself. Pearson Education.
- Firdaus, Thoriq dan Frain, Ben. 2016. HTML5 and CSS3: Building Responsive Websites. Packt Publishing Ltd.
- Haverbeke, Marjin. 2014. Eloquent JavaScript, 2nd Ed. No Starch Press.

Website :

[https://www.w3schools.com/html/.](https://www.w3schools.com/html/)

[https://www.w3schools.com/css/default.asp.](https://www.w3schools.com/css/default.asp)

[https://www.w3schools.com/js/default.asp.](https://www.w3schools.com/js/default.asp)



INSTITUT BISNIS
& INFORMATIKA
KESATUAN

PEMROGRAMAN

WEB
