



SOFTWARE LABORATORY CENTER

# Practicum Module

## HTML5 AND CANVAS

### Hands on Lab

[MONTH] [YEAR]

*Information in this document, including URL and other Internet Web site references, is subject to change without notice. This document supports a preliminary release of software that may be changed substantially prior to final commercial release, and is the proprietary information of Binus University.*

*This document is for informational purposes only. BINUS UNIVERSITY MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.*

*The entire risk of the use or the results from the use of this document remains with the user. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Binus University.*

*Binus University may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Binus University, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.*

*Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.*

*© 2011 Binus University. All rights reserved.*

*The names of actual companies and products mentioned herein may be the trademarks of their respective owners.*

# DAFTAR ISI

<b>BAB 1 INTRODUCING HTML5 .....</b>	<b>7</b>
1.1 INTRODUCING THE SEMANTIC ELEMENTS (HEADER, FOOTER, ARTICLE, ETC) .....	9
1.2 THE SEMANTIC ELEMENT REVISITED .....	11
1.3 OTHER STANDARDS THAT BOOST SEMANTICS .....	15
1.4 NEW FORM ELEMENTS .....	15
1.5 NEW INPUT TYPES .....	17
1.6 NEW INPUT ATTRIBUTES .....	19
1.7 NEW ATTRIBUTE SYNTAX.....	20
1.8 EXERCISE .....	20
<b>BAB 2 CSS AND VIEWPORT .....</b>	<b>25</b>
2.1 CSS SELECTOR .....	27
2.2 CSS BOX AND POSITIONING .....	29
2.3 WEBKIT CSS EXTENSIONS .....	37
2.4 VIEWPORT.....	38
2.5 EXERCISE .....	41
<b>BAB 3 JAVASCRIPT.....</b>	<b>43</b>
3.1 JAVASCRIPT VARIABLE .....	44
3.2 JAVASCRIPT STRING.....	46
3.3 JAVASCRIPT ARRAY .....	48
3.4 CONVERTING DATA TYPE.....	51
3.5 SORTING .....	53
3.6 JAVASCRIPT DATE .....	54
3.7 GEOLOCATION.....	57
3.8 JAVASCRIPT LIBRARY FOR MOBILE.....	58
3.9 EXERCISE.....	65
<b>BAB 4 JAVASCRIPT – OBJECT ORIENTED PROGRAMMING .....</b>	<b>68</b>
4.1 ENCAPSULATION .....	69
4.2 INHERITANCE .....	72
4.3 POLYMORPHISM .....	73
4.4 EXERCISE .....	75
<b>BAB 5 DRAWING WITH CANVAS.....</b>	<b>80</b>
5.1 INTRODUCTION OF CANVAS.....	81
5.2 DRAWING SHAPES & PATH – FILLS & STROKE .....	84
5.3 DRAWING SHAPES & PATH – RECTANGLE .....	95
5.4 DRAWING SHAPES & PATH - ARC AND CIRCLES.....	100
5.5 DRAWING SHAPES & PATH – GRADIENT.....	103
5.6 ADDING TEXT, IMAGES, AND SHADOW.....	108
5.7 TRANSFORMATIONS (TRANSLATING, SCALING, AND ROTATING) .....	115

5.8	EXERCISE.....	120
<b>BAB 6 ANIMATION .....</b>		<b>125</b>
6.1	MANAGING THE STATE STACK .....	126
6.2	ACCESSING IMAGE DATA .....	128
6.3	RETRIEVING PIXEL VALUES .....	130
6.4	UPDATING PIXEL VALUES.....	132
6.5	EXPORTING IMAGE DATA FILE .....	135
6.6	ANIMATION TIMING .....	137
6.7	RENDERING AND APPLYING ANIMATION .....	143
6.8	MOVING OBJECT, REMOVING OBJECT, ADD OBJECT.....	146
6.9	EXERCISE .....	155
<b>BAB 7 MOUSE AND KEYBOARD DETECTION.....</b>		<b>160</b>
7.1	MOUSE EVENTS.....	161
7.2	KEYBOARD EVENTS.....	165
7.3	CANVAS MOUSE COORDINATES .....	167
7.4	EXERCISE .....	169
<b>BAB 8 STORAGE DATA .....</b>		<b>172</b>
8.1	LOCAL STORAGE .....	173
8.2	WEB SQL .....	174
8.3	EXERCISE .....	180
<b>BAB 9 CREATING AUDIO AND VIDEO.....</b>		<b>183</b>
9.1	INTRODUCTION HTML5 AUDIO AND VIDEO.....	184
9.2	ADDING USER CONTROL .....	188
9.3	PRELOADING AUDIO AND VIDEO .....	189
9.4	SPECIFYING MULTIPLE SOURCE FILES .....	190
9.5	CONTROLLING PLAYBACK (VOLUMES, AUDIO EVENTS, CUSTOMER UI CONTROLS) .....	190
9.6	EXERCISE .....	195

## OVERVIEW

### BAB 1

- Introducing HTML5

Pengenalan singkat HTML5 serta fitur-fitur baru yang ada pada HTML5.

### BAB 2

- CSS and Viewport

Penjelasan mengenai cara penggunaan CSS dan Viewport.

### BAB 3

- JavaScript

Penjelasan mengenai JavaScript dan cara penggunaannya.

### BAB 4

- JavaScript - Object Oriented Programming

Penjelasan mengenai penerapan OOP dalam JavaScript.

### BAB 5

- Drawing with Canvas

Penjelasan mengenai penggunaan canvas di HTML5.

### BAB 6

- Animation

Penjelasan mengenai penerapan animasi pada Canvas.

## **BAB 7**

- Mouse and Keyboard Detection

Penjelasan cara deteksi inputan dari mouse atau keyboard di luar maupun dalam bagian canvas.

## **BAB 8**

- Storage Data

Penjelasan mengenai cara penyimpanan data pada local browser.

## **BAB 9**

- Creating Audio and Video

Penjelasan mengenai cara embed video maupun audio dan settingan yang diperlukan.





Sebelum membahas tentang HTML5, HTML itu sendiri adalah sebuah bahasa *Markup Standard* yang digunakan untuk membuat halaman web. Bahasa *markup* dapat digunakan untuk menentukan *formatting*. Elemen yang terkandung dalam HTML adalah *tag* dimana *tag* tersebut terbagi menjadi dua yaitu:



**Gambar 1.1.1 Logo HTML5**

1. **Single Tag**, *tag* yang tidak mempunyai penutup.

Contoh : `<br/>` `<input/>`

2. **Open and Close Tag**, *tag* yang mempunyai *tag* pembuka dan penutup.

Contoh : `<html>` `</html>`

Apabila suatu dokumen HTML yang berisi banyak *tag* dibaca oleh *browser*, *browser* tidak akan menampilkan *tag-tag* tersebut melainkan mengubahnya menjadi bentuk yang dapat dilihat atau dengar.

HTML5 adalah pengembangan dari HTML 4 yang diresmikan dan dikelola oleh **W3C (World Wide Web Consortium)** pada Oktober 2014. HTML5 ini diawali dengan *tag* `<!DOCTYPE html>`. *Tag* tersebut adalah sebuah instruksi kepada *browser* tentang versi HTML yang akan digunakan untuk membaca halaman tersebut. *Tag* `<!DOCTYPE html>` memberitahukan *browser* bahwa halaman ini merupakan halaman HTML5.



## 1.1 Introducing the Semantic Elements (header, footer, article, etc)

**Semantik** adalah suatu arti atau makna yang terkandung dalam suatu kata, sehingga **elemen semantik** adalah elemen yang mempunyai arti khusus. Elemen semantik mendeskripsikan makna dari elemen tersebut baik kepada *browser* ataupun *developer*. Contoh dari elemen semantik adalah `<form>` dan `<img>` dimana nama *tag* kedua elemen tersebut secara jelas memaparkan kegunaan dari elemen tersebut.

Berikut adalah beberapa elemen semantik beserta dengan kegunaannya:

### a. Elemen `<header>`

Mendefinisikan *header* dari suatu dokumen, biasanya untuk merepresentasikan pembuka untuk suatu konten dan biasanya memuat satu atau lebih *element heading*, logo, atau informasi pembuat.

```
<!DOCTYPE html>
<html>
<head>
  <title>Web Berisi Header</title>
</head>
<body>
  <header>
    <h1>
      Ini adalah judul dari suatu artikel
    </h1>
  </header>
</body>
</html>
```

### b. Elemen `<footer>`

Menjelaskan *footer* dari suatu dokumen atau *section*. biasanya memuat informasi tentang pembuat dokumen, informasi hak cipta, *link*, informasi kontak, atau aturan penggunaan.

```

<!DOCTYPE html>
<html>
<head>
    <title>Web Berisi Footer</title>
</head>
<body>
    <footer>
        <p>
            Copyright Prawowo 2014
        </p>
        <p>
            Contact :wowowo@contoh.com
        </p>
    </footer>
</body>
</html>

```

### c. Elemen <article>

Elemen yang bersifat independen atau dapat berdiri sendiri tanpa dipengaruhi oleh isi dari halaman lainnya. Elemen ini digunakan untuk membuat suatu artikel misalnya untuk forum, blog, berita, dan lainnya.

```

<!DOCTYPE html>
<html>
<head>
    <title>Web Berisi Artikel</title>
</head>
<body>
    <article>
        <h1>
            Pengaruh Makanan cepat saji terhadap kesehatan
        </h1>
        <p>
            Makanan cepat saji memiliki dampak yang buruk pada
            saluran pencernaan manusia apabila dikonsumsi
            secara berlebihan
        </p>
    </article>
</body>
</html>

```

## 1.2 The Semantic Element Revisited

Berikut adalah perbedaan yang membedakan HTML5 dan HTML 4:

Fitur	HTML 4	HTML5
<b>Penulisan sintaks</b>	Penulisan sintaks yang kadang terlalu panjang dan tidak sederhana. Seperti pada penggunaan <i>tag doctype</i> .  <!DOCTYPE HTML PUBLIC "//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">	Penggunaan sintaks yang lebih sederhana.  <!DOCTYPE html>
<b>Multimedia</b>	Apabila ingin memutar audio atau video dalam halaman HTML 4, dibutuhkan perangkat tambahan seperti <b>Adobe Flash Player</b> .	Sudah disediakan spesifikasi yang diperlukan sehingga pemutaran audio ataupun video bisa dilakukan tanpa perangkat tambahan. Hal ini dilakukan menggunakan <i>tag</i> <audio> dan <video>
<b>Error handling</b>	<i>Error handling</i> yang dilakukan tidak konsisten karena tidak ada aturan yang pasti.	Mulai dibuat standarisasi untuk <i>error handling</i> .
<b>API baru</b>	Pada HTML 4 sangat sulit untuk mengambil lokasi dari pengunjung suatu <i>website</i> . Lebih sulit lagi ketika pengunjung mengunjungi lewat perangkat <i>mobile</i> . Pada HTML 4 tidak disediakan <i>local storage</i> .	Disediakan dua buah API baru yaitu <i>geolocation</i> dan <i>local storage</i> dimana <i>geolocation</i> dapat membantu mengakses lokasi dari user dan <i>local storage</i> adalah tempat penyimpanan lokal dari sisi klien yang disimpan dalam <i>browser</i> dengan batasan sekitar 5MB. Selain itu ada beberapa API lain seperti <i>drag and drop</i> , <i>application cache</i> , dan lain sebagainya
<b>Elemen-elemen baru</b>	-	<i>Graphic</i> , <i>semantic</i> , dan <i>control element</i> adalah beberapa elemen baru yang

		<p>dimiliki oleh HTML5.</p> <p>Pada bagian <i>graphic</i> disediakan <code>&lt;canvas&gt;</code>.</p> <p>Pada bagian semantik sudah disediakan <i>tag-tag</i> seperti <code>&lt;header&gt;</code>, <code>&lt;footer&gt;</code>, <code>&lt;article&gt;</code>, dan lainnya.</p> <p>Pada bagian kontrol elemen disediakan <b>number</b>, <b>date</b>, <b>time</b>, <b>calendar</b>, dan <b>range</b> untuk membantu dalam melakukan validasi dan input <i>form</i>.</p>
--	--	---

Berikut adalah *tag-tag* dari HTML 4 yang sudah digantikan atau disarankan sudah tidak digunakan lagi di HTML5:

Tag-tag HTML 4	Kegunaan	Pengganti
<code>&lt;acronym&gt;</code>	Mendefinisikan kepanjangan dari suatu singkatan.	<code>&lt;abbr&gt;</code>
<code>&lt;applet&gt;</code>	Mendefinisikan <i>applet</i> yang ditanam/dipasang.	<code>&lt;embed&gt;</code>
<code>&lt;basefont&gt;</code>	Mendefinisikan warna, ukuran, dan jenis font di semua teks.	css
<code>&lt;big&gt;</code>	Memperbesar ukuran teks.	css
<code>&lt;center&gt;</code>	Membuat konten menjadi di tengah.	css
<code>&lt;dir&gt;</code>	Mendefinisikan direktori.	<code>&lt;ul&gt;</code>
<code>&lt;font&gt;</code>	Mendefinisikan warna, ukuran, dan jenis font.	css
<code>&lt;frame&gt;</code>	Mendefinisikan <i>frame</i> /bingkai	<code>&lt;iframe&gt;</code>

	didalam <i>frameset</i> /kumpulan bingkai.	
<frameset>	Mendefinisikan kumpulan <i>frame</i> /bingkai.	-
<noframes>	Mendefinisikan isi <i>alternative</i> apabila tidak mendukung <i>frame</i> .	-
<strike>	Mendefinisikan teks yang dicoret/digaris tengah	<del>
<tt>	Mendefinisikan teks seperti teks pada mesin ketik.	css

Berikut adalah *tag-tag* baru yang dimunculkan di HTML5 beserta dengan kegunaannya:

Tag	Kegunaan
<article>	Elemen yang bersifat independen atau dapat berdiri sendiri tanpa dipengaruhi oleh isi dari halaman lainnya. Biasanya berisi artikel.
<aside>	Elemen yang menjelaskan beberapa bagian dari konten namun memiliki letak disamping konten utama. Elemen ini harus memiliki hubungan dengan konten utama tersebut.
<audio>	Mendefinisikan konten suara.
<bdi>	Memisahkan bagian dari teks yang akan dibuat dengan format berbeda dari teks yang lain.
<canvas>	Sebagai media untuk menggambar (menggunakan <i>JavaScript</i> ).
<datalist>	Mendefinisikan nilai-nilai <i>autocomplete</i> yang digunakan sebagai pilihan dari elemen <input>.
<details>	Menampilkan informasi tambahan dimana user bisa menentukan akan ditampilkan atau tidak.
<dialog>	Mendefinisikan kotak dialog atau <i>window</i> .
<embed>	Mendefinisikan container untuk aplikasi yang bukan

	HTML.
<figcaption>	Menspesifikasi teks yang merupakan keterangan dari suatu gambar pada elemen <figure>.
<figure>	Menspesifikasi gambar.
<footer>	Menjelaskan <i>footer</i> dari suatu dokumen atau <i>section</i> .
<header>	Mendefinisikan <i>header</i> dari suatu dokumen.
<keygen>	Digunakan untuk menghasilkan kunci pada <i>form</i> , dimana akan dibuatkan dua kunci yaitu <i>private key</i> yang disimpan secara lokal dan <i>public key</i> yang dikirim ke <i>server</i> .
<main>	Menjelaskan konten utama dari suatu dokumen.
<mark>	Menjelaskan teks yang ditandai.
<menuitem>	Mendefinisikan perintah/ <i>menu item</i> yang digunakan <i>user</i> melalui menu <i>popup</i> .
<meter>	Mendefinisikan nilai skalar dalam bentuk <i>bar</i> .
<nav>	Berisi <i>link-link</i> untuk navigasi.
<output>	Mendefinisikan hasil kalkulasi.
<progress>	Representasi <i>progress</i> dari suatu pekerjaan.
<rp> , <rt>, dan <ruby>	Untuk penulisan huruf asia timur seperti huruf kanji dan hiragana dari jepang.
<section>	Mendefinisikan <i>section</i> dari suatu dokumen.
<source>	Mendefinisikan sumber dari media untuk elemen media seperti video atau audio.
<summary>	Rangkuman atau legenda dari konten elemen detail.
<time>	Untuk tanggal/waktu.
<track>	Mendefinisikan tulisan <i>track</i> untuk <i>tag</i> video dan audio.
<video>	Mendefinisikan video atau <i>movie</i> .
<wbr>	Untuk memberikan tanda dimana kalimat boleh diputuskan dengan   atau <i>break</i> .

### 1.3 Other Standards that Boost Semantics

Berikut adalah beberapa *tag* semantik yang meningkatkan kejelasan makna dari suatu bagian pada halaman web.

Tag	Kegunaan
<aside>	Elemen yang menjelaskan beberapa bagian dari konten namun memiliki letak disamping konten utama. Elemen ini harus memiliki hubungan dengan konten tersebut.
<details>	Menampilkan informasi tambahan dimana pengguna bisa menentukan akan ditampilkan atau tidak.
<figcaption>	Menspesifikasi tulisan sebagai keterangan gambar dari elemen <figure>.
<figure>	Menspesifikasi gambar.
<main>	Menjelaskan konten utama dari suatu dokumen.
<mark>	Menjelaskan tulisan yang ditandai.
<nav>	Untuk <i>link</i> navigasi.
<section>	Untuk mendefinisikan <i>section</i> dari suatu dokumen.
<summary>	Untuk rangkuman atau legenda dari konten elemen detail.
<time>	Untuk tanggal/waktu.

### 1.4 New Form Elements

Berikut adalah beberapa elemen dari *form* yang dapat digunakan pada HTML5:

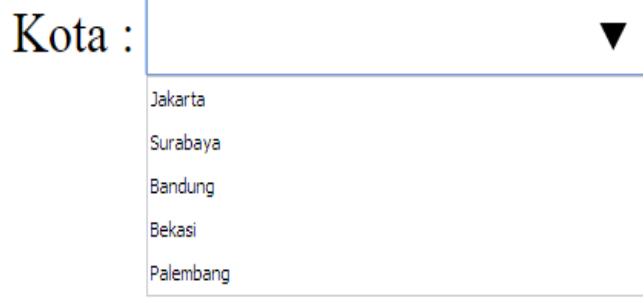
#### a. Elemen <Datalist>

Mendefinisikan nilai-nilai *autocomplete* yang digunakan sebagai pilihan dari elemen <input>. Sehingga *tag* input mempunyai kemampuan *autocomplete* secara *dropdown* berdasarkan nilai-nilai yang sudah ditentukan.

```

<!DOCTYPE html>
<html>
<head>
  <title>Web Berisi Datalist</title>
</head>
<body>
  Kota : <input list="kota"/>
  <datalist id="kota">
    <option value="Jakarta"/>
    <option value="Surabaya"/>
    <option value="Bandung"/>
    <option value="Bekasi"/>
    <option value="Palembang"/>
  </datalist>
</body>
</html>

```



Kota :

- Jakarta
- Surabaya
- Bandung
- Bekasi
- Palembang

Gambar 1.1.2 Contoh Datalist

### b. Elemen <keygen>

Digunakan untuk menghasilkan kunci pada *form*, dimana akan dibuatkan dua kunci yaitu *private key* yang disimpan secara lokal dan *public key* yang dikirim ke *server*.

```

<!DOCTYPE html>
<html>
<head>
  <title>Web Berisi Keygen</title>
</head>
<body>
  <form action="#" method="get">
    Username: <input type="text" name="nama">
    Encryption: <keygen name="security">
    <input type="submit">
  </form>
</body>
</html>

```



### c. Elemen <output>


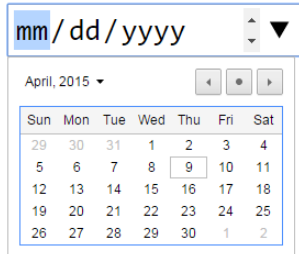
Mendefinisikan hasil kalkulasi yang didapat dari perhitungan yang ditentukan.

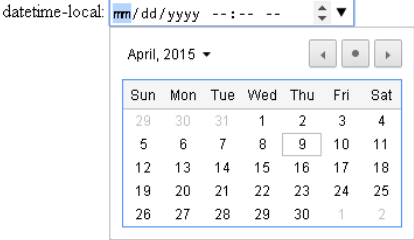
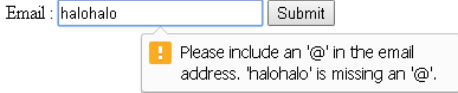
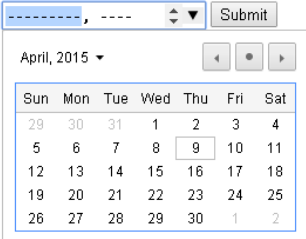
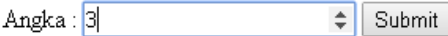

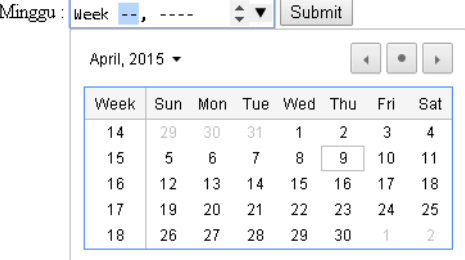

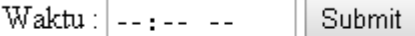
```
<!DOCTYPE html>
<html>
<head>
  <title>Web Berisi Keygen</title>
</head>
<body>
  <form oninput= "nilai.value = parseInt(angka1.value) +
    parseInt(angka2.value)">


    <input type="number" id="angka1" value="50">
    +
    <input type="number" id="angka2" value="50">
    =
    <output name="nilai" for="angka1 angka2"></output>
  </form>
</body>
</html>
```

## 1.5 New Input Types

Berikut adalah beberapa jenis tipe input yang baru di HTML5:

Tipe Input	Kegunaan	Hasil
Color	Digunakan untuk menerima input warna.	Warna : 
Date	Digunakan untuk menerima input tanggal.	Tanggal : 
Datetime	Digunakan untuk menerima input tanggal dan waktu dengan zona waktu.	-

Datetime-local	Digunakan untuk menerima input tanggal dan waktu tanpa zona waktu/dengan waktu local.	
Email	Digunakan untuk menerima input alamat email.	
Month	Digunakan untuk menerima input bulan dan tahun.	
Number	Digunakan untuk menerima input nilai numerik.	
Range	Digunakan untuk menerima input nilai numerik dalam bentuk rentangan bilangan.	
Week	Digunakan untuk menerima input minggu dan tahun.	
Search	Digunakan untuk pencarian (memiliki sifat yang sama dengan tipe input teks biasa).	
Tel	Digunakan untuk menerima input nomor telepon.	-
Time	Digunakan untuk menerima input waktu tanpa zona waktu / dengan waktu local.	

Url	Digunakan untuk menerima input alamat URL.	URL : <input type="text" value="www.google"/> <input type="button" value="Submit"/>  Please enter a URL.
-----	--	--

## 1.6 New Input Attributes

Berikut adalah beberapa atribut baru HTML5 dari suatu *tag* input:

Atribut	Kegunaan
autocomplete	Mendefinisikan apakah suatu input memerlukan <i>autocomplete</i> atau tidak.
autofocus	Atribut <i>boolean</i> dimana suatu elemen secara otomatis akan mendapatkan fokus atau tidak ketika halaman dimuat.
form	Digunakan untuk menspesifikasi satu atau lebih <i>form</i> yang didalamnya terdapat elemen input.
formaction	Menspesifikasi URL dimana <i>file</i> tempat inputan di proses ketika <i>form</i> dikirim/ <i>submit</i> .
formenctype	Menspesifikasikan bagaimana data dari <i>form</i> akan dikodekan ketika di kirim ke <i>server</i> . Hal ini hanya bisa dilakukan dengan metode pengiriman <i>post</i> .
formmethod	Menspesifikasikan metode pengiriman HTTP yang akan digunakan.
formnovalidate	Ketika ada, mendefinisikan bahwa elemen input tidak akan divalidasi ketika dikirim.
formtarget	Atribut yang digunakan untuk menggantikan atribut target di <i>form</i> sehingga apabila ada atribut target di <i>form</i> dan atribut <i>formtarget</i> di input maka atribut target di <i>form</i> akan diabaikan dan menggunakan atribut <i>formtarget</i> di input.
height and width	Menentukan tinggi dan lebar dari suatu elemen input.
list	Menunjuk kepada elemen <i>datalist</i> yang memuat pilihan nilai-nilai dari input elemen.
min and max	Menentukan nilai minimal dan maksimal dari suatu elemen input.
multiple	Atribut <i>boolean</i> dimana ketika bernilai benar, <i>user</i> bisa memasukan lebih dari satu nilai dari input elemen.
pattern (regexp)	Atribut yang akan melakukan validasi suatu input sesuai dengan pola <i>regex</i> ( <i>regular expression</i> ) yang diberikan.

placeholder	Memberikan kalimat bantu dalam mengisi inputan yang diminta.
required	Memberitahukan bahwa input harus diisi sebelum <i>form</i> tersebut dikirim.
step	Menspesifikasi interval angka untuk suatu elemen input.

## 1.7 New Attribute Syntax

Setiap elemen bisa mempunyai atribut. Atribut berfungsi untuk menyediakan informasi tambahan terkait dengan elemen tersebut.

HTML5 mempunyai empat aturan penulisan suatu atribut:

### a. Empty

```
<input type="text" value="John" readonly>
```

Penulisan atribut yang dimaksud adalah atribut 'readonly' dimana atribut tersebut hanya berupa nama atribut, tanpa nilai/*value*.

### b. Unquoted

```
<input type="text" value=John>
```

Penulisan atribut yang dimaksud adalah penulisan nilai dari atribut *value* dimana pada pemberian nilai tidak perlu ditambahkan tanda petik dua.

### c. Double quoted

```
<input type="text" value="John Doe">
```

Penulisan atribut yang dimaksud adalah penulisan nilai dari atribut *value* dimana pada pemberian nilai menggunakan tanda petik dua.

### d. Single quoted

```
<input type="text" value='John Doe'>
```

Penulisan atribut yang dimaksud adalah penulisan nilai dari atribut *value* dimana pada pemberian nilai menggunakan tanda petik satu.

## 1.8 Exercise

Buatlah dua buah halaman web yang saling terhubung dengan menu navigasi yaitu "**beranda.html**" dan "**registrasi.html**". Berikut adalah rinciannya:

"beranda.html" berisi:

- Pada bagian atas terdapat dua buah *link* untuk menuju **beranda.html** dan **registrasi.html**.
- Sebuah artikel tentang HTML5 dengan judul dan isi.
- Sebuah gambar logo HTML5 dengan *caption*.
- Sebuah *footer* berisi hak cipta pemilik.



Gambar 1.1.3 home.html

"registrasi.html" berisi sebuah form yang menerima inputan:

- Pada bagian atas terdapat dua buah *link* untuk menuju **beranda.html** dan **registrasi.html**.
- **Asal Negara** yang menggunakan **datalist** yang terdiri dari **Indonesia**, **Malaysia**, dan **Singapura** dengan **placeholder** 'Negara' serta harus diisi.
- **Angka Kesukaan** dengan nilai minimal 0 dan nilai maksimal 100 serta harus diisi.
- **Warna Kesukaan** yang menggunakan **color** serta harus diisi.
- **Tanggal Lahir** yang menggunakan **date** serta harus diisi.
- **Email** menggunakan **email** serta harus diisi.
- Tombol pengiriman *form* dengan tulisan "**Daftar**".

- Sebuah *footer* berisi hak cipta pemilik.

Gambar 1.1.4 registration.html

Code untuk kedua halaman:

```
<!--beranda.html-->
<!DOCTYPE html>
<html>
<head>
  <title>
    Soal Latihan
  </title>
</head>
<body>
  <nav>
    <a href="beranda.html">
      Beranda
    </a>
    &nbsp;
    <a href="registrasi.html">
      Registrasi
    </a>
  </nav>
  <article>
    <h1>
      Penggunaan HTML5
    </h1>
    <p>
      HTML5 memiliki banyak kelebihan dibandingkan dengan
      versi sebelumnya. Banyak fitur-fitur baru yang sangat
```

menunjang pembuatan suatu halaman web.

```

    </p>
    <figure>
      
      <figcaption>
        Gambar 1. Logo HTML5
      </figcaption>
    </figure>
  </article>
  <footer>
    <p>
      &copy; TM14-0
    </p>
  </footer>
</body>
</html>

```

```

<!--registrasi.html-->
<!DOCTYPE html>
<html>
<head>
  <title>
    Soal Latihan
  </title>
</head>
<body>
  <nav>
    <a href="beranda.html">
      Beranda
    </a>
    &nbsp;
    <a href="registrasi.html">
      Registrasi
    </a>
  </nav>
  <br>
  <h1>
    Formulir Registrasi
  </h1>
  <br>
  <form>
    Asal Negara:
    <input list="negara" required placeholder="Negara"/>
    <datalist id="negara">
      <option value="Indonesia"/>
      <option value="Malaysia"/>
      <option value="Singapura"/>
    </datalist>
  </form>

```

```
</datalist>
<br>
Angka Kesukaan:
<input type="number" required min="0" max="100"/>
<br>
Warna Kesukaan:
<input type="color" required/>
<br>
Tanggal Lahir:
<input type="date" required/>
<br>
Email:
<input type="email" required/>
<br>
<input type="submit" value="Daftar"/>
</form>
<footer>
  <p>
    &copy; TM14-0
  </p>
</footer>
</body>
</html>
```





**CSS** adalah singkatan dari *Cascading Style Sheet* yang ditambahkan ke HTML versi 4.0. Kegunaan CSS adalah untuk mendefinisikan bagaimana elemen HTML ditampilkan. Sebelumnya, HTML tidak pernah memfokuskan pada pengaturan format dokumen melainkan lebih kepada mendefinisikan konten suatu dokumen. Ketika *tag-tag* seperti `<font>` dan atribut warna dimunculkan pada HTML 3.2, hampir setiap halaman menggunakan kedua hal tersebut sehingga proses untuk memuat halaman web menjadi sangat lama. Pada HTML 4.0, semua pengaturan format tersebut harus dihapuskan dan digantikan menggunakan CSS, disarankan adalah CSS yang terpisah dengan file HTML.

Ada tiga cara untuk menggunakan CSS:

#### a. External

*File* CSS diletakan terpisah dari *file* HTML dengan ekstensi `.css`. Berikut adalah cara menghubungkannya dengan *file* HTML. Tambahkan potongan *code* berikut di dalam *tag* `<head>`:

```
<link rel="stylesheet" type="text/css" href="mystyle.css">
```

#### b. Internal

CSS didefinisikan didalam *tag* `<head>` *file* HTML. Contoh:

```
<head>
  <style>
    div{width:100px;}
  </style>
</head>
```

#### c. Inline

CSS didefinisikan sebagai atribut dari suatu elemen. Contoh:

```
<div style="width:100px;"></div>
```

Contoh CSS *internal* dan *external*:

```
div{background-color:red;}
```

Pada contoh diatas, **div** adalah *selector*, sehingga pada CSS *internal* akan memilih semua **div** yang terdapat pada halaman HTML sedangkan pada CSS *inline* hanya pada elemen **div** itu saja. **Background-color** adalah properti dari elemen **div** yang nilainya akan diubah. Sedangkan **red** adalah nilai atau *value* yang diberikan pada suatu properti.

## 2.1 CSS Selector

CSS *Selector* adalah suatu cara bagaimana memilih elemen-elemen HTML yang ingin diberikan *style*. Berikut adalah *selector* yang dapat digunakan pada CSS:

Selector	Contoh	Penjelasan
.class	.navmenu	Memilih setiap elemen yang memiliki class="navmenu".
#id	#par	Memilih setiap elemen yang memiliki id="par".
*	*	Memilih semua elemen.
elemen	div	Memilih semua elemen <div>.
elemen, elemen	div, span	Memilih semua elemen <div> dan <span>.
elemen elemen	div p	Memilih semua elemen <p> didalam elemen <div>.
elemen>elemen	div>p	Memilih semua elemen <p> yang memiliki <i>parent</i> berupa elemen <div>.
elemen+elemen	div+p	Memilih semua elemen <p> yang ditempatkan setelah elemen <div>.
elemen~elemen	p~div	Memilih semua elemen <div> yang diawali dengan elemen <p>.
[attribute]	[target]	Memilih semua elemen dengan atribut <b>target</b> .
[attribute=value]	[target=_blank]	Memilih semua elemen dengan atribut <b>target</b> yang bernilai "_blank".
[attribute~=value]	[title~=super]	Memilih semua elemen dengan atribut <b>title</b> yang nilainya mengandung kata "super".
[attribute =value]	[title =sup]	Memilih semua elemen dengan atribut <b>title</b> yang nilainya diawali dengan "sup".
[attribute^=value]	a[href^="http"]	Memilih semua elemen <a> dimana atribut <b>href</b> pada elemen tersebut diawali dengan "http".
[attribute\$=value]	a[href\$=".exe"]	Memilih semua elemen <a> dimana atribut <b>href</b> pada elemen tersebut diakhiri dengan ".exe".
[attribute*=value]	a[href*="google"]	Memilih semua elemen <a> dimana atribut <b>href</b> pada elemen tersebut mengandung "google".
:active	a:active	Memilih semua <i>link</i> yang aktif.

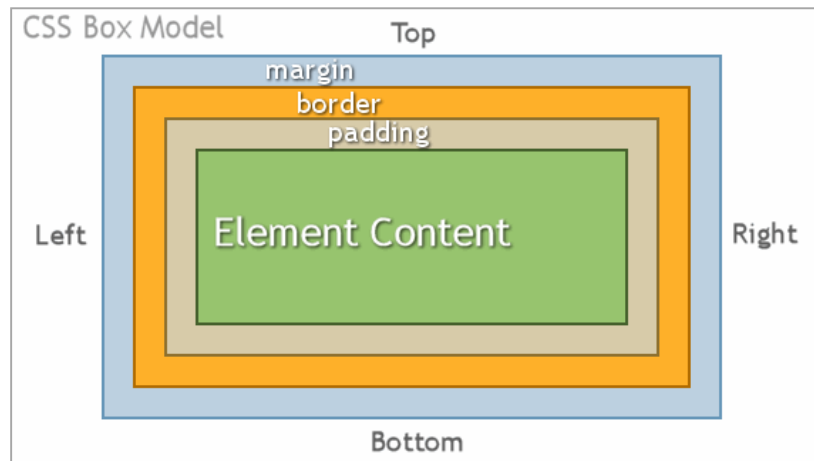
:after	div:after	Memasukan konten setelah setiap elemen <div>.
:before	div:before	Memasukan konten sebelum setiap elemen <div>.
:checked	input:checked	Memilih setiap elemen <input> yang dicek/centang.
:disabled	input:disabled	Memilih setiap elemen <input> yang di <i>disabled</i> .
:empty	div:empty	Memilih setiap elemen <div> yang tidak mempunyai <i>child</i> .
:enabled	input:enabled	Memilih setiap elemen <input> yang di <i>enabled</i> .
:first-child	p:first-child	Memilih setiap elemen <p> yang merupakan <i>child</i> pertama dari <i>parent</i> -nya.
::first-letter	p::first-letter	Memilih huruf pertama dari setiap elemen <p>.
::first-line	p::first-line	Memilih baris pertama dari setiap elemen <p>.
:first-of-type	p:first-of-type	Memilih setiap elemen <p> yang merupakan elemen <p> pertama dari <i>parent</i> -nya.
:focus	input:focus	Memilih elemen <input> yang sedang mendapatkan fokus.
:hover	a:hover	Memilih <i>link</i> yang sedang diarahkan oleh <i>pointer mouse</i> .
:in-range	input:in-range	Memilih setiap elemen <input> yang nilainya dalam rentang yang ditentukan.
:invalid	input:invalid	Memilih setiap elemen <input> yang memiliki nilai yang <i>invalid</i> .
:lang	p:lang(eng)	Memilih setiap elemen <p> yang memiliki atribut <b>lang</b> bernilai “eng” ( <i>English</i> ).
:last-child	p:last-of-type	Memilih setiap elemen <p> yang merupakan <i>child</i> terakhir dari <i>parent</i> -nya.
:last-of-type	p:last-of-type	Memilih setiap elemen <p> yang merupakan elemen <p> terakhir dari <i>parent</i> -nya.
:link	a:link	Memilih setiap <i>link</i> yang belum dikunjungi.
:not(selector)	:not(p)	Memilih semua elemen yang bukan elemen <p>.
:nth-child(n)	p:nth-child(3)	Memilih setiap elemen <p> yang merupakan

		<i>child</i> ketiga dari <i>parent</i> -nya.
:nth-last-child(n)	p:nth-last-child(4)	Memilih setiap elemen <p> yang merupakan <i>child</i> keempat dari <i>parent</i> -nya, dihitung dari belakang.
:nth-last-of-type(n)	p:nth-last-of-type(2)	Memilih setiap elemen <p> yang merupakan elemen <p> kedua, dihitung dari belakang.
:nth-of-type	P:nth-of-type(2)	Memilih setiap elemen <p> yang merupakan elemen <p> kedua dari <i>parent</i> -nya.
:only-of-type	P:only-of-type	Memilih setiap elemen <p> yang merupakan elemen <p> satu-satunya dari <i>parent</i> -nya.
:only-child	P:only-child	Memilih setiap elemen <p> yang merupakan <i>child</i> satu-satunya dari <i>parent</i> -nya.
:optional	Input:optional	Memilih setiap elemen <input> yang tidak memiliki atribut <b>required</b> .
:out-of-range	Input:out-of-range	Memilih elemen input dengan nilai diluar rentang yang ditentukan.
:read-only	Input:read-only	Memilih setiap elemen <input> dengan atribut <b>readonly</b> .
:read-write	Input:read-write	Memilih setiap elemen <input> tanpa atribut <b>readonly</b> .
:required	Input:required	Memilih setiap elemen <input> dengan atribut <b>required</b> .
:root	:root	Memilih elemen <i>root</i> dari dokumen.
::selection	::selection	Memilih bagian dari elemen yang dipilih/diselect oleh <i>user</i> .
:valid	Input:valid	Memilih setiap elemen <input> dengan nilai yang <i>valid</i> .
:visited	A:visited	Memilih setiap <i>link</i> yang sudah dikunjungi.

## 2.2 CSS Box and Positioning

*Box model* adalah sebuah istilah untuk setiap elemen HTML yang dibungkus oleh sebuah lapisan yang terdiri dari *margin*, *border*, *padding* dan konten. *Box model* memiliki

kemampuan untuk memberikan garis tepian disekitar elemen, menentukan jarak antara konten ke garis tepian atau menentukan jarak antar elemen.



**Gambar 2.1** *Box model*

Berikut adalah beberapa properti yang dapat ditentukan terkait *box model*:

#### a. Margin

*Margin* memiliki fungsi untuk mengatur jarak antara elemen satu dengan elemen yang lain. Jarak tersebut dihitung dari *border* menuju elemen lain. Berikut adalah beberapa properti untuk mengatur *margin*:

Properti	Kegunaan	Cara pemberian parameter
Margin-left	Menentukan jarak <i>margin</i> /jarak diluar <i>border</i> bagian kiri dengan elemen lain.	<code>margin-left:20px;</code>
Margin-bottom	Menentukan jarak <i>margin</i> /jarak diluar <i>border</i> bagian bawah dengan elemen lain.	<code>margin-bottom:20px;</code>
Margin-right	Menentukan jarak <i>margin</i> /jarak diluar <i>border</i> bagian kanan dengan elemen lain.	<code>margin-right:20px;</code>
Margin-top	Menentukan jarak <i>margin</i> /jarak diluar <i>border</i> bagian atas dengan elemen lain.	<code>margin-top:20px;</code>
Margin	Menentukan jarak <i>margin</i> /jarak diluar <i>border</i> dengan elemen lain dan merupakan cara pendek dalam menentukan <i>margin</i> ke semua arah ( <i>Shorthand Property</i> ).	<ol style="list-style-type: none"> <li><code>margin: 30px 40px 50px 60px;</code> <ul style="list-style-type: none"> <li>- <i>Margin</i> atas 30px</li> <li>- <i>Margin</i> kiri 40px</li> <li>- <i>Margin</i> bawah 50px</li> <li>- <i>Margin</i> kanan 60px</li> </ul> </li> <li><code>margin: 30px 40px 50px;</code> <ul style="list-style-type: none"> <li>- <i>Margin</i> atas 30px</li> <li>- <i>Margin</i> kiri dan kanan 40px</li> <li>- <i>Margin</i> bawah 50px</li> </ul> </li> <li><code>margin: 30px 40px;</code> <ul style="list-style-type: none"> <li>- <i>Margin</i> atas dan bawah 30px</li> <li>- <i>Margin</i> kiri dan kanan 40px</li> </ul> </li> <li><code>margin: 30px;</code> <ul style="list-style-type: none"> <li>- Semua <i>margin</i> 30px</li> </ul> </li> </ol>

#### b. Padding

*Padding* memiliki fungsi untuk mengatur jarak antara *border* dengan konten dari suatu elemen. Warna *padding* akan mengikuti dengan warna dari suatu elemen. Berikut adalah beberapa properti untuk mengatur *padding*:

Properti	Kegunaan	Cara pemberian parameter
Padding-left	Menentukan jarak antara <i>border</i> bagian kiri dengan konten elemen.	<code>padding-left:20px</code>
Padding-bottom	Menentukan jarak antara <i>border</i> bagian bawah dengan konten elemen.	<code>padding-bottom:20px</code>
Padding-right	Menentukan jarak antara <i>border</i> bagian kanan dengan konten elemen.	<code>padding-right:20px</code>
Padding-top	Menentukan jarak antara <i>border</i> bagian atas dengan konten elemen.	<code>padding-top:20px</code>
Padding	Menentukan jarak antara <i>border</i> dengan konten elemen dan merupakan cara pendek dalam menentukan <i>padding</i> ke semua arah ( <i>Shorthand Property</i> ).	<ol style="list-style-type: none"> <li><code>padding:30px 40px 50px 60px;</code> <ul style="list-style-type: none"> <li>- <i>Padding</i> atas 30px</li> <li>- <i>Padding</i> kiri 40px</li> <li>- <i>Padding</i> bawah 50px</li> <li>- <i>Padding</i> kanan 60px</li> </ul> </li> <li><code>padding: 30px 40px 50px;</code> <ul style="list-style-type: none"> <li>- <i>Padding</i> atas 30px</li> <li>- <i>Padding</i> kiri dan kanan 40px</li> <li>- <i>Padding</i> bawah 50px</li> </ul> </li> <li><code>padding: 30px 40px;</code> <ul style="list-style-type: none"> <li>- <i>Padding</i> atas dan bawah 30px</li> <li>- <i>Padding</i> kiri dan kanan 40px</li> </ul> </li> <li><code>padding: 30px;</code> <ul style="list-style-type: none"> <li>- Semua <i>padding</i> 30px</li> </ul> </li> </ol>

### c. Border

*Border* memiliki fungsi untuk menentukan garis tepi seperti apa yang akan ditampilkan. Berikut adalah beberapa properti untuk mengatur *border*:



Properti	Kegunaan	Cara pemberian parameter
Border-style	Menentukan jenis garis tepian yang akan ditampilkan. Dapat diisi dengan <b>none</b> , <b>dotted</b> , <b>dashed</b> , <b>solid</b> , <b>double</b> , <b>groove</b> , <b>ridge</b> , <b>inset</b> , atau <b>outset</b> .	<code>border-style: solid;</code>
Border-color	Menentukan warna garis tepian yang akan ditampilkan.	<code>border-style: red;</code>
Border-width	Menentukan ketebalan dari garis tepian.	<code>border-width: 15px;</code>
Border	Menentukan jenis, warna, dan ketebalan garis tepian dalam satu deklarasi ( <i>Shorthand property</i> ).	<code>border: 2px solid black;</code>

Selain itu, setiap *border* dapat ditentukan setiap sisinya saja misalnya :

- `border-right: 2px solid black;`
- `border-right-style: solid;`
- `border-top-width: 10px;`

**d. Display**

Properti ini menentukan jenis *box* yang akan digunakan oleh elemen HTML. Contoh:

`display : inline;`

Nilai	Kegunaan
Inline	Menampilkan elemen tersebut sehingga dapat menjadi satu baris dengan elemen <i>inline</i> lainnya.
Block	Menampilkan elemen dimana satu elemen akan ditampilkan sebagai satu buah <i>block</i>
List-item	Menampilkan elemen yang memiliki bentuk elemen <i>list</i> / <code>&lt;li&gt;</code>
Inline-block	Menampilkan elemen dimana kontennya diperlakukan sebagai <i>block</i> dan elemen itu sendiri diperlakukan sebagai <i>inline</i>
Run-in	Menampilkan elemen sebagai <i>block</i> atau <i>inline</i> tergantung dari isinya.
None	Elemen tidak akan ditampilkan.

**e. Float**

Dengan property *float*, suatu elemen dapat didorong ke kiri atau kanan sehingga elemen lain dapat membungkus elemen tersebut. Contoh:

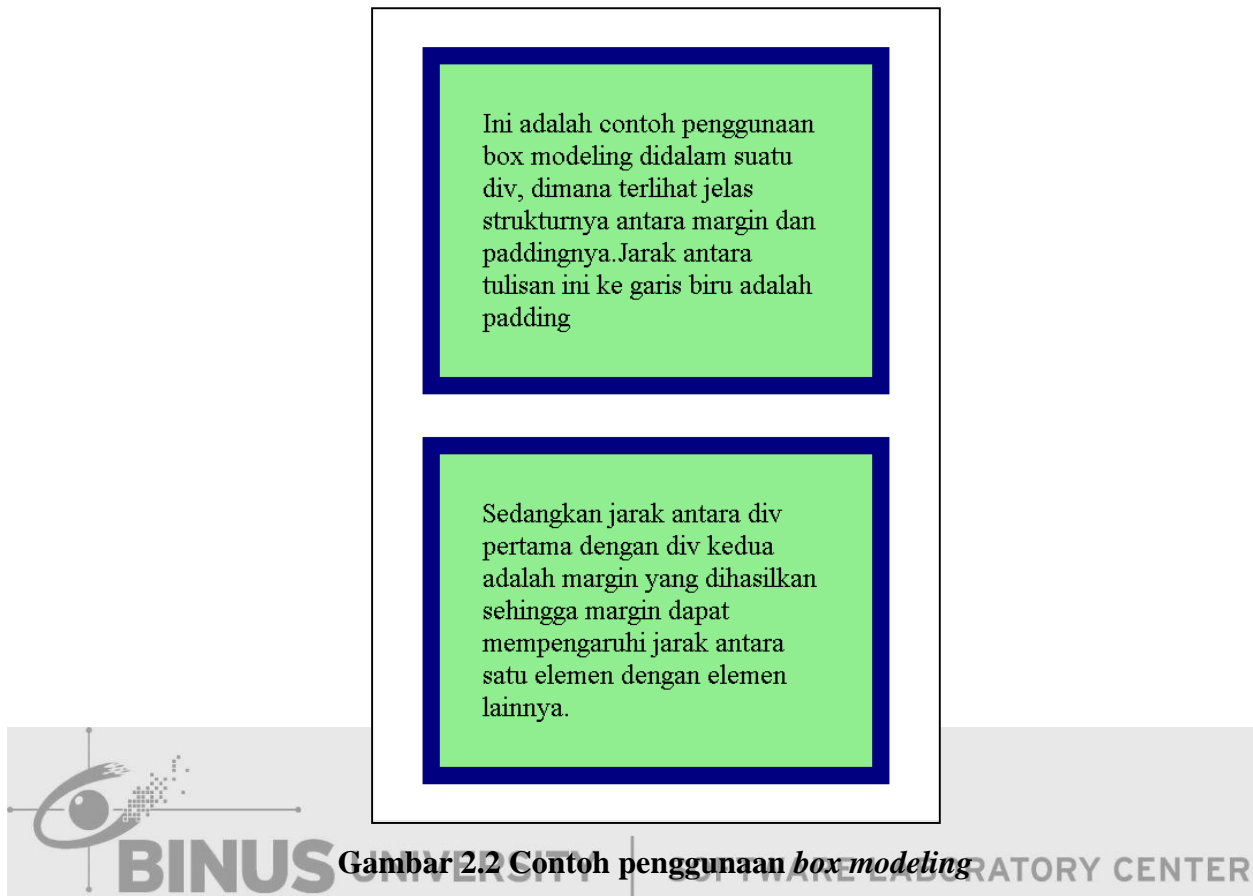
`float: left;`

Nilai	Kegunaan
None	Elemen tidak mengambang dan akan ditampilkan seperti biasa.
Left	Elemen mengambang ke kiri.
Right	Elemen mengambang ke kanan.

Berikut adalah contoh *code* penggunaan *box model* dalam suatu elemen **div**:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div
    {
      background-color: lightgreen;
      width: 200px;
      padding: 25px;
      border: 10px solid navy;
      margin: 25px;
    }
  </style>
</head>
<body>
  <div>
    Ini adalah contoh penggunaan box modeling didalam suatu
    div, dimana terlihat jelas strukturnya antara margin dan
    paddingnya.
    Jarak antara tulisan ini ke garis biru adalah padding
  </div>
  <div>
    Sedangkan jarak antara div pertama dengan div kedua adalah
    margin yang dihasilkan sehingga margin dapat mempengaruhi
    jarak antara satu elemen dengan elemen lainnya.
  </div>
</body>
</html>
```

Hasil dari *code*:



*Positioning properties* mengatur bagaimana suatu elemen diletakan pada suatu halaman web. Dengan hal ini, dapat diletakkan suatu elemen dibelakang elemen lain, suatu elemen yang menempel dengan layar *browser*, ataupun ketika konten dari suatu elemen terlalu besar. Contoh:

```
position:fixed;
```

Ada 4 jenis metode *positioning* :

### 1. Static

Secara *default*, elemen HTML diposisikan secara *static*, elemen yang diposisikan secara *static* akan selalu diposisikan menurut alur normal suatu halaman yaitu dari atas dan kebawah. Elemen yang diposisikan *static* ini tidak akan dipengaruhi oleh properti **top**, **bottom**, **left**, dan **right**.

### 2. Fixed

Elemen yang diposisikan secara *fixed* akan diposisikan relatif dengan *browser window*, sehingga tidak akan bergerak meskipun *window* di-*scroll*. Elemen *fixed* dihapuskan dari alur normal suatu alaman sehingga dianggap tidak ada, hal ini membuat elemen *fixed* bisa tumpang tindih dengan elemen lainnya.

### 3. Relative

Elemen yang diposisikan secara *relative* diposisikan secara relatif terhadap posisi normalnya sehingga elemen tersebut bisa berpindah dan bertumpang tindih dengan elemen lain namun tempat yang dipesan untuk elemen tersebut masih ada di alur normalnya.

### 4. Absolute

Elemen yang diposisikan secara *absolute* diposisikan secara relatif dengan elemen orang tua pertamanya yang memiliki posisi selain *static*. Elemen ini dihapuskan dari alur normal suatu halaman sehingga dianggap tidak ada, hal ini membuat elemen *absolute* bisa tumpang tindih dengan elemen lainnya.

## 2.3 WebKit CSS Extensions

Webkit adalah sebuah prefiks yang diberikan pada fitur *experimental* CSS khusus untuk browser **Google Chrome** dan **Safari**, hal itu dikarenakan **Google Chrome** dibuat berdasarkan mesin *layout* **webkit** sehingga pada beberapa fitur CSS seperti *transform*, *animations*, dan *rotate* memerlukan prefiks **webkit**. Tujuan dari prefiks ini adalah agar *browser* dapat menambahkan bantuan untuk fitur CSS terbaru.

Berikut adalah beberapa prefiks *browser*:

1. -webkit- untuk **Android**, **Google Chrome**, **Safari**, dan **IOS**
2. -moz- untuk **Mozilla Firefox**
3. -o- untuk **Opera**
4. -ms- untuk **Internet Explorer**

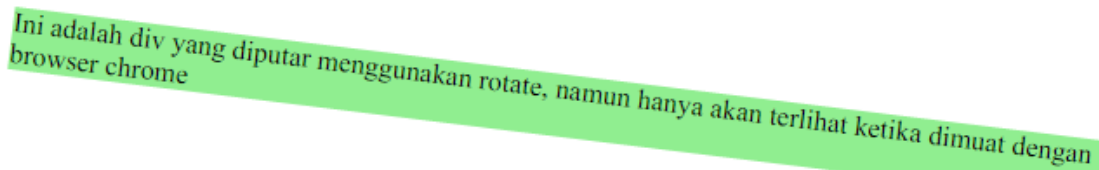
Berikut adalah contoh penggunaan **webkit** untuk beberapa fitur *eksperimental* css:

```

<!DOCTYPE html>
<html>
<head>
  <style>
    div
    {
      -webkit-transform: rotate(7deg);
      background-color: lightgreen;
    }
  </style>
</head>
<body>
  <br><br><br><br>
  <div>
    Ini adalah div yang diputar
    menggunakan rotate, namun hanya
    akan terlihat ketika dimuat dengan
    browser chrome
  </div>
</body>
</html>

```

Berikut adalah hasil dari *code* tersebut:



**Gambar 2.3** Contoh penggunaan webkit

## 2.4 Viewport

**Viewport meta tag** digunakan untuk desain web yang fleksibel sehingga *layout* halaman web bisa dibuka di perangkat lain seperti *mobile phone*, *tab*, ataupun perangkat lainnya. **Viewport** ini digunakan apabila ingin membuat suatu web yang *responsive*. Berikut adalah contoh penggunaannya:

```
<meta name="viewport" content="">
```

Berikut adalah beberapa atribut yang bisa diatur di dalam **viewport**:

### 1. Width

Bagian yang paling penting dalam sebuah *viewport* adalah memberitahukan lebar dari suatu halaman.

**Contoh:** `<meta name="viewport" content="width=device-width">`

Dengan hal ini akan secara tidak langsung memberitahukan suatu halaman untuk mengadaptasi dengan lebar perangkat yang digunakan. Namun apabila digunakan di suatu halaman yang tidak *responsive* maka *website* tersebut akan di *zoom* di bagian pojok atas kiri, membuat *user* harus melakukan *zoom out* untuk kembali ke tampilan webnya.

## 2. Initial-scale

Atribut ini mengatur tingkat *initial zoom* sehingga dapat menyelesaikan beberapa masalah seperti mengubah perangkat menjadi *landscape*.

**Contoh:** `<meta name="viewport" content="width=device-width, initial-scale=1">`

Bagian ini akan memberitahukan kepada halaman bahwa *1 pixel* pada CSS sama dengan *1 pixel* pada *viewport* sehingga tidak akan terjadi masalah ketika melakukan *zoom*.

## 3. Maximum-scale

Atribut ini digunakan untuk mengatur *maximum zoom*. Dengan mengatur nilai *maximum scale* ini menjadi **1**, maka *user* tidak akan diperbolehkan untuk melakukan *zoom*.

**Contoh:** `<meta name="viewport" content="width=device-width, maximum-scale=1">`

## 4. User-scalable

Atribut ini mengatur kemampuan *zoom in* atau *zoom out* yang dimiliki oleh suatu halaman.

**Contoh:** `<meta name="viewport" content="width=device-width, user-scalable=no">`

Dengan hal ini, maka secara tidak langsung akan menghapus kemampuan *zoom in* dan *zoom out* suatu halaman.

Biasanya penggunaan **maximum-scale** dan **user-scalable** jarang sekali digunakan, *tag viewport* yang biasa digunakan hanya seperti ini:

`<meta name="viewport" content="width=device-width, initial-scale=1">`

Selain **viewport**, terdapat **@media rule** yang memungkinkan media yang berbeda memiliki *style* yang berbeda juga. **@media** yang akan dibahas adalah **@media screen**

dimana digunakan untuk kelas *desktop*. Beberapa *browser* pada *smartphone* saat ini juga sudah dianggap kelas *desktop* dalam kategori layarnya. Berikut adalah contoh penggunaanya:

```
<!DOCTYPE html>
<html>
<head>
<style>
  @media screen and (max-width: 1024px) and (min-width: 512px)
  {
    p
    {
      font-family: arial black;
      font-size: 32px;
    }
  }
</style>
</head>
<body>
  <br><br>
  <p>
```

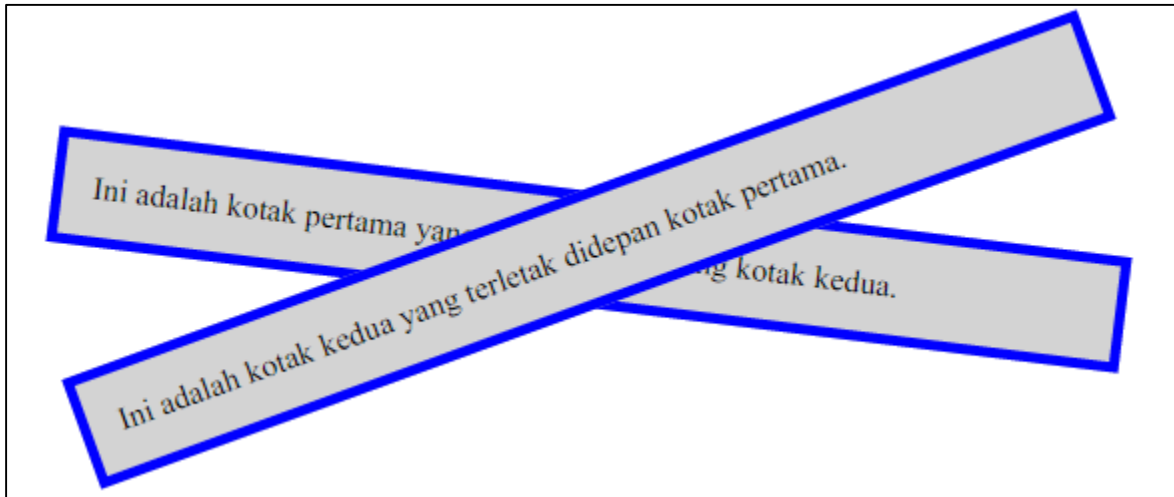
Ini adalah cara penggunaan dari @media screen dimana style diatas akan berlaku ketika lebar minimal browser adalah 512px dan lebar maksimal browser 1024px

```
</p>
</body>
</html>
```



## 2.5 Exercise

Buatlah sebuah halaman web dengan nama “**index.html**” dengan *file* CSS yang terpisah dengan nama “**style.css**”. Gunakan **viewport metatag** yang menyesuaikan layar serta atur nilai *initial zoom* menjadi 1. Berikut adalah tampilannya:



**Gambar 2.4 Hasil dari latihan**

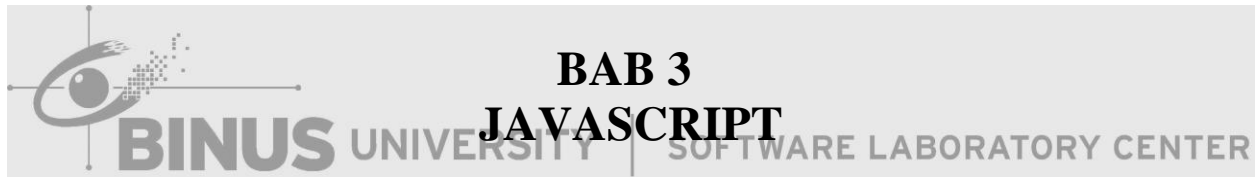
Berikut adalah *code* dari “**index.html**”:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <br><br><br><br>
  <div id="firstbox">
    Ini Adalah kotak pertama yang terletak
    dibelakang kotak kedua.
  </div>
  <div id="secondbox">
    Ini Adalah kotak kedua yang terletak
    didepan kotak pertama.
  </div>
</body>
</html>
```

Berikut adalah isi dari “style.css”:

```
#firstbox
{
    background-color: lightgrey;
    width: 500px;
    padding: 15px;
    border: 5px solid blue;
    margin: 15px;
    -webkit-transform: rotate(7deg);
    position: absolute;
}

#secondbox
{
    background-color: lightgrey;
    width: 500px;
    padding: 15px;
    border: 5px solid blue;
    margin: 15px;
    -webkit-transform: rotate(-20deg);
}
```



**JavaScript** adalah bahasa *scripting* yang digunakan pada *client side/browser*. Dalam implementasinya, *JavaScript* dapat digunakan untuk membuat interaksi antara *client side/browser* dengan *user*, mengontrol *browser*, berkomunikasi secara tidak sinkron (*asynchronously*), dan mengubah isi dari dokumen HTML. Selain itu, *JavaScript* juga dapat digunakan dari *server side*, contohnya adalah *node.js*.

Dalam pembahasan ini, penggunaan *JavaScript* akan berfokus pada HTML. Untuk menggunakan *JavaScript* pada HTML, dapat dilakukan dengan cara berikut:

**a. Internal**

*JavaScript* terletak pada dokumen yang sama dengan dokumen HTML. Dalam hal ini, *JavaScript* akan terletak diantara tag `<script></script>`, jumlah tag `<script>` yang ada tidak memiliki batasan, berapapun jumlah tag `<script>` yang ada dalam satu halaman HTML tidak akan ada masalah. Tag `<script>` tersebut bisa terletak didalam tag `<head>` ataupun `<body>`, namun disarankan menggabungkan semua `<script>` menjadi satu tag `<script>` saja agar memudahkan dalam membaca *code*. Berikut adalah contoh membuat *JavaScript* dengan cara *internal*:

```
<script type="text/javascript">  
  alert("Mencoba JavaScript");  
</script>
```

**b. External**

*JavaScript* terletak pada dokumen yang terpisah dengan dokumen HTML. Berikut adalah contoh membuat *JavaScript* dengan cara *external*:

```
<script src="myScript.js"></script>
```

*myScript.js* adalah nama dokumen *JavaScript* yang akan dihubungkan dengan HTML.

### 3.1 JavaScript Variable

Variabel adalah penampung suatu nilai yang merupakan karakteristik utama suatu bahasa pemrograman dimana setiap penampung tersebut mempunyai nama dan tipe data. Nama unik yang diberikan kepada variabel di *JavaScript* dinamakan *identifier* dimana suatu variabel akan diakses menggunakan *identifier* tersebut.

Beberapa aturan umum dalam penamaan variabel adalah:

1. Nama boleh mengandung huruf, angka, garis bawah, dan tanda dolar
2. Nama harus diawali huruf
3. Nama bisa juga diawali \$ dan \_
4. Nama bersifat *case sensitive*
5. Tidak boleh menggunakan *keyword* dari *JavaScript*

Variabel di *JavaScript* memiliki tiga tipe data primitif:

1. *Numeric*, seperti 1, 123, dan 1.01
2. *String*, seperti "Kata ini"
3. *Boolean*, bernilai *true* atau *false*

Selain tiga tipe data primitif diatas, *JavaScript* juga mempunyai tipe data komposit dimana tipe data komposit tersebut berupa koleksi dari banyak nilai. Contoh dari tipe data komposit adalah *object*, *array*, dan *function*.

Sebelum menggunakan variabel di *JavaScript*, variabel tersebut harus dideklarasikan terlebih dahulu. Pendeklarasian variabel di *JavaScript* menggunakan kata kunci **var**. Sedangkan, pemberian nilai kepada variabel dinamakan inisialisasi. Inisialisasi dapat dilakukan ketika variabel tersebut dibuat atau ketika dibutuhkan. Berikut adalah contohnya:

```
<script type="text/javascript">
    var kata;
    var nama, barang;
    var hari = "Minggu";
    kata = "Meja";
</script>
```

Sistem variabel pada *JavaScript* bersifat **loosely typed language**, artinya tipe data variabel tidak dideklarasikan secara eksplisit, dan terkadang akan melakukan konversi secara otomatis apabila diperlukan. Variabel juga memiliki jangkauan/cakupan/*scope*, artinya

variabel tersebut terdefinisi sampai jangkauan variabel tersebut masih ada. Berdasarkan rentang variabelnya, *JavaScript* memiliki dua jenis variabel, yaitu:

#### a. Variabel global

Memiliki jangkauan global, artinya terdefinisi untuk keseluruhan kode *JavaScript*. Ketika suatu variabel dideklarasikan diluar dari fungsi, maka variabel tersebut secara otomatis menjadi variabel global. Apabila suatu variabel digunakan di suatu fungsi tanpa menggunakan **var**, maka variabel tersebut juga menjadi variabel global.

#### b. Variabel lokal

Jangkauan variabel tersebut hanya sampai fungsi tersebut terdefinisi. Parameter suatu fungsi atau variabel yang dideklarasikan didalam fungsi akan menjadi variabel lokal.

### 3.2 JavaScript String

*String* digunakan untuk menyimpan dan memanipulasi suatu teks dimana teks tersebut terletak diantara dua petik. Petik yang digunakan bisa petik satu atau petik dua.

```
<script type="text/javascript">
```

```
var hari = "Minggu";
```

```
var Kata = 'Meja';
```

```
</script>
```

*String* pada *JavaScript* memiliki *method* dan *property* yang dapat membantu dalam manipulasi isi dari variabel tersebut. Hal ini dikarenakan tipe data primitif *string* ini diperlakukan sebagai *object*.

Berikut beberapa *method* dan *property* yang terdapat pada *string*:

Method/Property	Kegunaan	Contoh	Penjelasan
length	Mengembalikan panjang karakter dari suatu <i>string</i> .	kata.length	Mengembalikan jumlah karakter dari variabel <b>kata</b> .
charAt(index)	Mengembalikan suatu karakter dari indeks yang	kata.charAt(5)	Mengembalikan karakter dari variabel <b>kata</b> pada indeks ke-5.

	ditentukan.		
fromCharCode(num)	Mengubah nilai <i>unicode</i> menjadi karakter.	String.fromCharCode(65)	Mengembalikan <i>string</i> yang merupakan karakter dari nilai ASCII 65 yaitu karakter 'A'.
indexOf(string)	Mengembalikan indeks dari suatu <i>string</i> yang dicari di dalam <i>string</i> yang dicari mulai dari paling kiri ke paling kanan.	kata.indexOf("an")	Mengembalikan indeks dari <i>string</i> "an" yang dicari dari paling kiri ke paling kanan. Jika <i>string</i> yang dicari tidak ditemukan, maka akan mengembalikan nilai -1.
lastIndexOf(string)	Mengembalikan indeks dari suatu <i>string</i> yang dicari di dalam <i>string</i> yang dicari mulai dari paling kanan ke paling kiri.	kata.lastIndexOf("an")	Mengembalikan indeks dari <i>string</i> "an" yang dicari dari paling kanan ke paling kiri. Jika <i>string</i> yang dicari tidak ditemukan, maka akan mengembalikan nilai -1.
replace(old_string, new_string)	Mencari suatu nilai didalam <i>string</i> dan mengganti nilai tersebut serta mengembalikan nilai <i>string</i> yang baru.	kata.replace("bandung", "jakarta")	Mencari "bandung" di <i>string kata</i> dan apabila ditemukan akan digantikan dengan "jakarta".
substr(x, y)	Mengambil sebanyak y karakter yang diambil dari indeks ke-x dari <i>string</i> tertentu.	kata.substr(1, 4)	Mengambil bagian dari <i>string kata</i> yang dimulai dari indeks pertama sebanyak 4 karakter.
substring(x, y)	Mengambil mulai dari indeks ke-x sampai indeks ke-y	kata.substring(1, 4)	Mengambil bagian dari <i>string kata</i> yang dimulai dari indeks pertama sampai

	dari <i>string</i> tertentu.		dengan indeks keempat.
<code>toLowerCase()</code>	Mengubah <i>string</i> menjadi huruf kecil.	<code>kata.toLowerCase()</code>	Mengembalikan <i>string kata</i> yang sudah diubah menjadi huruf kecil.
<code>toUpperCase()</code>	Mengubah <i>string</i> menjadi huruf kapital.	<code>kata.toUpperCase()</code>	Mengembalikan <i>string kata</i> yang sudah diubah menjadi huruf kapital.
<code>trim()</code>	Menghilangkan <i>whitespace</i> dari depan dan belakang suatu <i>string</i> .	<code>kata.trim()</code>	Mengembalikan <i>string kata</i> yang sudah dihilangkan spasi diawal dan diakhir <i>string</i> .

Berikut adalah contoh penggunaan dari beberapa *method* dan *property*:

```

<!DOCTYPE html>
<html>
<head>
  <script type="text/javascript">
    var kata = prompt("Masukan kata yang ingin dicek!");
    if(kata.length < 10)
    {
      alert("Kata terlalu pendek!");
    }else
    {
      alert("Karakter indeks ke 3 adalah : "+kata.charAt(3));
      alert("Mengganti a dengan o : "+kata.replace("a","o"));
      alert("Diubah menjadi kapital : "+kata.toUpperCase());
      alert("Dibubah menjadi huruf kecil : "+kata.toLowerCase());
    }
  </script>
</head>
<body>
  Penggunaan beberapa method dan property di string
</body>
</html>

```

### 3.3 JavaScript Array

*Array* adalah *object* yang digunakan untuk menampung lebih dari satu nilai didalam satu variabel. *Array* bisa menampung nilai sebanyak apapun dan setiap nilai dapat dengan mudah diakses menggunakan nomor indeksinya. *Array* pada *JavaScript* bersifat heterogen,



artinya dalam satu *array* bisa menampung lebih dari satu jenis tipe data. Berikut adalah penggunaan *array*:

```
<!DOCTYPE html>
<html>
<head>
  <script type="text/javascript">
    // inisialisasi array nama
    var nama = ["Joko", "Andri", "Kumeng"];
    // Mengubah nilai array nama indeks pertama
    nama[0]="Amang";
    // inisialisasi array kata
    var kata = new Array("Meja", "Komputer", "Kursi");
    // Menampilkan keseluruhan nilai array nama dari indeks awal sampai indeks
    terakhir
    for(var i=0; i< nama.length;i++)
      alert("Nama : "+nama[i]);

  </script>
</head>
<body>
  Deklarasi, inisialisasi array dan mengakses nilai array
</body>
</html>
```

Array sebagai *object* di *JavaScript* memiliki beberapa *method* dan *property* seperti yang telah digunakan pada *code* diatas. Berikut adalah beberapa *method* dan *property* beserta dengan kegunaannya.

Method/Property	Kegunaan	Contoh	Penjelasan
length	Mengembalikan jumlah elemen <i>array</i> .	nama.length	Mengembalikan jumlah elemen dari <i>array</i> <b>nama</b> .
indexOf(value)	Mencari suatu elemen pada <i>array</i> dari indeks pertama ke indeks terakhir dan mengembalikan indeks posisinya.	nama.indexOf("Joko")	Mengembalikan indeks dimana <i>string</i> <b>"Joko"</b> berada. Pencarian dimulai dari indeks pertama sampai indeks terakhir.
lastIndexOf(value)	Mencari suatu elemen pada <i>array</i> dari indeks	nama.lastIndexOf("Joko")	Mengembalikan indeks dimana

	terakhir ke indeks pertama dan mengembalikan indeks posisinya.		<i>string</i> <b>"Joko"</b> berada. Pencarian dimulai dari indeks terakhir sampai indeks pertama.
pop()	Menghapus elemen terakhir dari <i>array</i> dan mengembalikan elemen tersebut.	<code>nama.pop()</code>	Menghapus elemen terakhir dari <i>array</i> <b>nama</b> dan mengembalikan elemen tersebut.
push()	Menambahkan elemen baru di paling akhir <i>array</i> dan mengembalikan jumlah elemen <i>array</i> yang baru.	<code>nama.push("Ruby")</code>	Menambahkan elemen baru bernilai <b>"Ruby"</b> didalam <i>array</i> <b>nama</b> dan mengembalikan ukuran <i>array</i> tersebut.
reverse()	Membalikan urutan dari elemen <i>array</i> .	<code>nama.reverse()</code>	Membalikan urutan dari <i>array</i> <b>nama</b> .
sort()	Mengurutkan elemen <i>array</i> .	<code>nama.sort()</code>	Mengurutkan <i>array</i> <b>nama</b> secara <i>ascending</i> .

Berikut adalah contoh penggunaannya:

```
<!DOCTYPE html>
<html>
<head>
    <title>Penggunaan Array Method dan property </title>
</head>
<body id="cont">

<script type="text/javascript">
    // inisialisasi array nama
    var nama = ["Joko dodo", "Andri doremi", "Kumeng komeng", "Denny yolo", "Alan
    sudibyo", "Andico"];
```

```

var body = document.getElementById("cont");

// Menambahkan isi dari array nama ke dalam tag body
for(var i=0; i< nama.length;i++)
{
    body.innerHTML += "Nama ke-" + (i+1) + " : " + nama[i] + "<br>";
}

// Menambahkan posisi dari elemen Andico ke dalam tag body
body.innerHTML += "<br><br>Indeks dari Andico : " + nama.indexOf("Andico") + "<br>";

// Menambahkan elemen Andico terakhir array nama ke dalam tag body dan menghapusnya
body.innerHTML += "<br>Elemen terakhir yang dihapus : " + nama.pop();
</script>
</body>

```

### 3.4 Converting Data Type

Pada dasarnya, *JavaScript* secara otomatis akan melakukan konversi tipe data pada waktu yang dibutuhkan namun terkadang pengubahan tipe data secara paksa juga diperlukan pada saat tertentu, terutama ketika mengembalikan suatu nilai dari suatu fungsi. Berikut adalah tipe data baik primitif ataupun komposit yang dapat menampung suatu nilai.

1. *String*
2. *Number*
3. *Boolean*
4. *Object*
5. *Date*
6. *Array*
7. *Function*

*JavaScript* menyediakan operator *typeof* untuk mengembalikan jenis tipe data dari suatu nilai. Berikut adalah penggunaannya:

```

<!DOCTYPE html>
<html>
<head>
    <title>Penggunaan operator typeof</title>
</head>
<body id="cont">
    <script type="text/javascript">

        // Mengembalikan string
        document.getElementById("cont").innerHTML += typeof "Joko" + "<br>";
    </script>
</body>

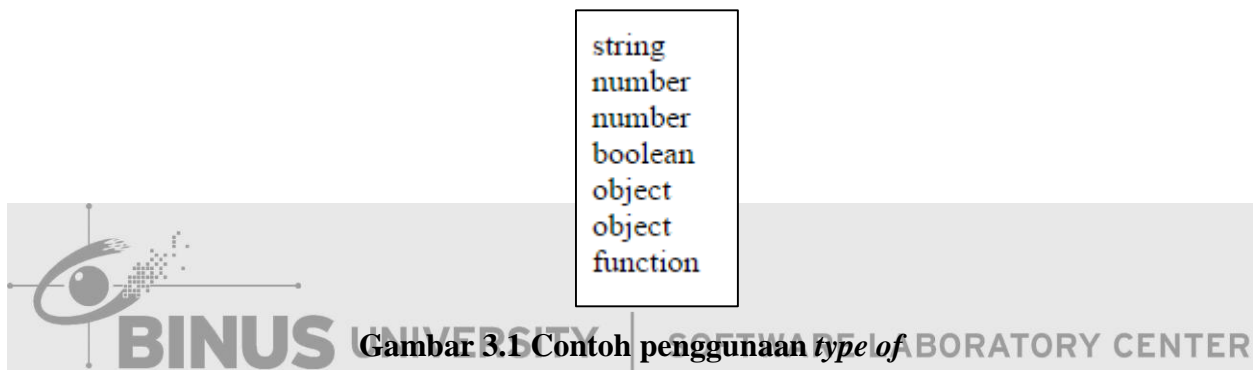
```

```

// Mengembalikan number
document.getElementById("cont").innerHTML += typeof 7.19 + "<br>";
// Mengembalikan number
document.getElementById("cont").innerHTML += typeof 9 + "<br>";
// Mengembalikan boolean
document.getElementById("cont").innerHTML += typeof false + "<br>";
// Mengembalikan object
document.getElementById("cont").innerHTML += typeof [1,2,2,3] + "<br>";
// Mengembalikan object
document.getElementById("cont").innerHTML += typeof new Date() + "<br>";
// Mengembalikan function
document.getElementById("cont").innerHTML += typeof function(){} + "<br>";
</script>
</body>
</html>

```

Hasil dari *code* tersebut:



Gambar 3.1 Contoh penggunaan *type of*

Apabila diperhatikan, tipe data *date* dan *array* mengembalikan berupa objek, untuk hal ini *typeof* tidak bisa membedakan apakah itu suatu *object*, *array* ataupun *date*. Berikut adalah cara mengubah suatu variabel menjadi suatu tipe data tertentu dengan menggunakan fungsi-fungsi yang telah disediakan:

Tipe Data Awal	Tipe Data Tujuan	Method	Contoh
Number	String	<ul style="list-style-type: none"> <li>String()</li> <li>toString()</li> <li>toExponential()</li> <li>toFixed()</li> <li>toPrecision()</li> </ul>	<pre> &lt;script type="text/javascript"&gt;   var a = (400).toString();   var b = String(1.123);   var c = (990.3939).toExponential(2);   var d = (120.9912).toFixed(2);   var e = (380.123).toPrecision(2);   alert(typeof a);   alert(typeof b);   alert(typeof c);   alert(typeof d);   alert(typeof e); &lt;/script&gt; </pre>

Boolean	String	<ul style="list-style-type: none"> <li>String()</li> <li>toString()</li> </ul>	<pre>&lt;script type="text/javascript"&gt;   var a = (false).toString();   var b = String(true);   alert(typeof a);   alert(typeof b); &lt;/script&gt;</pre>
Date	String	<ul style="list-style-type: none"> <li>String()</li> <li>toString()</li> </ul>	<pre>&lt;script type="text/javascript"&gt;   var a = Date().toString();   var b = String(Date());   alert(typeof a);   alert(typeof b); &lt;/script&gt;</pre>
String	Number	<ul style="list-style-type: none"> <li>Number()</li> <li>parseFloat()</li> <li>parseInt()</li> </ul>	<pre>&lt;script type="text/javascript"&gt;   var a = Number("123.33");   var b = Number.parseInt("4444");   var c = Number.parseFloat("123.11");   alert(typeof a);   alert(typeof b);   alert(typeof c); &lt;/script&gt;</pre>
Boolean	Number	<ul style="list-style-type: none"> <li>Number()</li> </ul>	<pre>&lt;script type="text/javascript"&gt;   var a = Number(true);   //var a akan bernilai 1   alert(typeof a); &lt;/script&gt;</pre>
Date	Number	<ul style="list-style-type: none"> <li>Number()</li> </ul>	<pre>&lt;script type="text/javascript"&gt;   var a = Number(new Date());   alert(typeof a); &lt;/script&gt;</pre>

### 3.5 Sorting

*Sorting* adalah proses penyusunan suatu elemen sehingga berurutan sesuai dengan aturan yang ditentukan baik secara *ascending* ataupun *descending*. *JavaScript* memiliki *method array* yang membantu proses ini yaitu `sort()` dan `reverse()`.

*Method* `sort()` berguna untuk mengurutkan elemen pada *array*, sedangkan `reverse()` untuk membalikan urutan elemen dari suatu *array*. *Method* `sort()` dapat mengurutkan alfabet ataupun numerik baik secara *ascending* ataupun *descending*.

*Method* `sort()` apabila parameternya tidak diisi maka akan mengurutkan secara *ascending*. *Parameter method* ini bersifat opsional dan berupa *function* yang harus memiliki *return type* negatif, nol atau nilai positif, misalnya untuk mengurutkan suatu *number* bisa seperti:

```
function(a, b){return a-b}
```

Berikut adalah contoh penggunaan *sort* dan *reverse*:

```

<!DOCTYPE html>
<html>
<head>
    <title>Penggunaan sort dan reverse</title>
</head>
<body id="cont">
    <script type="text/javascript">

        var kata = ["bola", "meja", "kursi", "telinga", "apel", "cacing"];

        for(var i=0; i<kata.length; i++)
            document.getElementById("cont").innerHTML += "Kata ke - " + (i+1) + " : " + kata[i] + "<br>";

        //di reverse
        kata.reverse();

        document.getElementById("cont").innerHTML += "<br><br>" + "Setelah direverse" + "<br><br>";

        for(var i=0; i<kata.length; i++)
            document.getElementById("cont").innerHTML += "Kata ke - " + (i+1) + " : " + kata[i] + "<br>";

        //di sort
        kata.sort();

        document.getElementById("cont").innerHTML += "<br><br>" + "Setelah disort ascending" + "<br><br>";

        for(var i=0; i<kata.length; i++)
            document.getElementById("cont").innerHTML += "Kata ke - " + (i+1) + " : " + kata[i] + "<br>";
    </script>
</body>
</html>

```

### 3.6 JavaScript Date

*Date* adalah sebuah *object* yang digunakan untuk memanipulasi atau mendapatkan tanggal yang meliputi tahun, bulan, hari, jam, menit, detik, dan milisekon.

Ada 4 cara untuk menginisialisasi suatu tanggal :

1. New Date()
2. New Date(milliseconds)
3. New Date(dateString)
4. New Date(year, month, day, hours, minutes, seconds, milliseconds)

Berikut adalah contoh inisialisasi tanggal:

```

<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Date</title>
</head>
<body id="cont">
    <script type="text/javascript">

        //Inisialisasi berdasarkan tanggal sekarang
        var a = new Date();

        //Inisialisasi berdasarkan tanggal 1 januari 1970 ,jam 07:00
        //ditambah dengan milisekon dari parameter
        //dikurang apabila milisekon bernilai minus
        var b = new Date(3600000);

        //inisialisasi dengan datestring
        var c = new Date("2014-12-29");

        //inisialisasi dengan tahun, bulan, hari, jam, dan menit
        var d = new Date(2011,10,9,20,30);

        alert(a);
        alert(b);
        alert(c);
        alert(d);

    </script>
</body>
</html>

```

Perlu diperhatikan bahwa di *JavaScript*, bulan Januari ditandai dengan angka 0, Febuari dengan angka 1 dan seterusnya. *Object date* juga mempunyai *method-method* yang dapat membantu dalam proses manipulasi tanggal, berikut adalah *method – method* nya:

Method	Kegunaan
getDate()	Mengembalikan tanggal hari sebagai Number (1-31)
getDay()	Mengembalikan hari dalam seminggu sebagai <i>number</i> (0-6)
getFullYear()	Mengembalikan tahun
getHours()	Mengembalikan jam (0-23)
getMilliseconds()	Mengembalikan milisekon (0-999)
getMinutes()	Mengembalikan menit (0-59)
getMonth()	Mengembalikan bulan (0-11)
getSeconds()	Mengembalikan detik (0-59)

setDate()	Menetapkan tanggal hari
setFullYear()	Menetapkan tahun
setHours()	Menetapkan Jam
setMilliseconds()	Menetapkan milisekon
setMinutes()	Menetapkan menit
setMonth()	Menetapkan bulan
setSeconds()	Menetapkan detik
setTime()	Menetapkan tanggal yang ditentukan berdasarkan 1 januari 1970 berdasarkan parameter milisekon
toString()	Mengubah tanggal menjadi <i>string</i>

Berikut adalah contoh penggunaannya:

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Date</title>
</head>
<body id="cont">
  <script type="text/javascript">
    var tm = new Date();

    tm.setFullYear(2000);
    tm.setMonth(11);
    tm.setDate(20);
    tm.setHours(5);
    tm.setMinutes(44);
    tm.setSeconds(20);

    alert("Tahun : " + tm.getFullYear());
    alert("Bulan : " + tm.getMonth());
    alert("Tanggal : " + tm.getDate());
    alert("Jam : " + tm.getHours());
    alert("Menit : " + tm.getMinutes());
    alert("Detik : " + tm.getSeconds());

  </script>
</body>
</html>
```



### 3.7 Geolocation

**Geolocation** adalah fitur baru yang dimunculkan untuk HTML5 dimana fitur ini digunakan untuk melacak posisi geografis dari *user*. Dikarenakan posisi *user* bersifat rahasia, maka fitur ini hanya akan tersedia apabila *user* mengijinkannya. Posisi *user* yang ditampilkan berdasarkan *longitude* dan *latitude*, dimana *longitude* adalah garis bujur dan *latitude* adalah garis lintang yang menspesifikasikan posisi dari *user* berdasarkan letaknya di bumi.

Berikut adalah contoh penggunaannya dalam mengambil *longitude* dan *latitude*.

1. Pertama siapkan HTML untuk menampilkan *latitude* dan *longitude*.

```
<!DOCTYPE html>
<html>
<head>
  <title>Geolocation</title>
</head>
<body>
  Berikut adalah posisi anda :
  <p id="pos"></p>
</body>
</html>
```

2. Kemudian buat sebuah fungsi untuk mengecek apakah *browser* didukung oleh fitur *geolocation*.

```
<script type="text/javascript">
function findPos()
{
  if (navigator.geolocation)
  {
    navigator.geolocation.getCurrentPosition(show);
  }
  else
  {
    alert("Geolocation tidak didukung oleh browser");
  }
}
</script>
```

3. Apabila tidak didukung maka akan menampilkan pesan *error*. Namun apabila didukung, maka akan menjalankan fungsi `getCurrentPosition()`, dan apabila fungsi `getCurrentPosition()` sukses dijalankan maka akan mengirimkan posisi *user* ke fungsi yang dispesifikasi di parameter yaitu `show()`. Selanjutnya kita akan membuat fungsi `show()`.

```

<script type="text/javascript">
var par = document.getElementById("pos");
function show(post)
{
    par.innerHTML = "Latitude: " + post.coords.latitude +
                    "<br>Longitude: " + post.coords.longitude;
}
</script>

```

4. Disini posisi akan ditampilkan dengan *property* yang telah disediakan yaitu `.coords.latitude` dan `.coords.longitude`. Berikut adalah keseluruhan kodenya:

```

<!DOCTYPE html>
<html>
<head>
    <title>Geolocation</title>
</head>
<body>
    Berikut adalah posisi anda :
    <p id="pos"></p>
    <script type="text/javascript">
        var par = document.getElementById("pos");
        function show(post)
        {
            par.innerHTML = "Latitude: " + post.coords.latitude +
                            "<br>Longitude: " + post.coords.longitude;
        }
        findPos();
        function findPos()
        {
            if (navigator.geolocation)
            {
                navigator.geolocation.getCurrentPosition(show);
            }
            else
            {
                alert("Geolocation tidak didukung oleh browser");
            }
        }
    </script>
</body>
</html>

```

### 3.8 JavaScript Library for Mobile

**jQuery Mobile** adalah *framework* tampilan pengguna yang berbasis *jQuery* dan bekerja disemua jenis *smartphone*, *tablet*, dan *platform desktop* yang menganut prinsip *responsive web design* (RWD). Namun, banyak orang salah mengartikan bahwa *jQuery*

*mobile* adalah versi *mobile* dari *jQuery*, padahal *jQuery mobile* adalah *framework* tampilan pengguna.

Ada dua cara dalam menghubungkan file HTML kita dengan *jQuery mobile*:

### 1. Menggunakan CDN (*Content Delivery Network*)

CDN digunakan untuk mendistribusi file yang sering digunakan di *web*, sehingga membuat kecepatan *download* jauh lebih cepat untuk pengguna.

```
<head>
    <link rel="stylesheet"
href="http://code.jquery.com/mobile/1.2.0/jquery.mobile-1.2.0.min.css">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
    <script src="http://code.jquery.com/mobile/1.2.0/jquery.mobile-
1.2.0.min.js"></script>
</head>
```

### 2. Menghubungkan *jQuery mobile* secara local

Selain dengan menggunakan CDN, cara lainnya adalah dengan melakukan *download library* tersebut di [jQuerymobile.com](http://jquerymobile.com). Selain itu, kita dapat membuat library *jQuery mobile* yang sesuai dengan tema yang kita buat, hal itu dapat dilakukan di <http://themeroller.jquerymobile.com/>.

```
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="jquery.mobile-1.4.5.css">
<script src="jquery.js"></script>
<script src="jquery.mobile-1.4.5.js"></script>
</head>
```

**jQuery mobile** menggunakan atribut data pada HTML5 dimana akan membantu dalam pembentukan tampilan yang lebih menarik untuk perangkat *mobile*. Penggunaan atribut *data-role* dapat mengubah langsung tampilan elemen yang diberikan atribut tersebut sehingga lebih menarik untuk tampilan *mobile*.

Berikut adalah contoh penggunaan *jQuery* dalam membuat tampilan sederhana menggunakan CDN:

```

<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
href="http://code.jquery.com/mobile/1.2.0/jquery.mobile-1.2.0.min.css">
  <script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
  <script src="http://code.jquery.com/mobile/1.2.0/jquery.mobile-
1.2.0.min.js"></script>
</head>
<body>
  <div data-role="page" data-theme="a">
    <div data-role="header">
      <h1>jQuery Mobile</h1>
    </div>
    <div data-role="content">
      <ul data-role="listview" >
        <li data-role="list-divider">Tanggal</li>
        <li><a href="1.html">Januari</a></li>
        <li><a href="2.html">Febuari</a></li>
        <li><a href="3.html">Maret</a></li>
        <li><a href="4.html">April</a></li>
        <li><a href="5.html">Mei</a></li>
        <li><a href="6.html">Juni</a></li>
        <li><a href="7.html">Juli</a></li>
        <li><a href="8.html">Agustus</a></li>
        <li><a href="9.html">September</a></li>
        <li><a href="10.html">Oktober</a></li>
        <li><a href="11.html">November</a></li>
        <li><a href="12.html">Desember</a></li>
      </ul>
    </div>
    <div data-role="footer">
      <h4>&copy; TM14-0</h4>
    </div>
  </div>
</body>
</html>

```

Hasil dari *code*:



**Gambar 3.2 Contoh penggunaan CDN**

*jQuery mobile* juga menawarkan beberapa *event* yang berguna dalam membuat *mobile web* yang interaktif dan menarik baik dari sisi *user* ataupun *developer*. Berikut adalah beberapa *event* yang biasa digunakan:

Event	Kegunaan
Mobileinit	Menandakan <i>jQuery mobile</i> sudah selesai dimuat.
Orientationchange event	Menandakan perubahan orientasi layar ( <i>portrait</i> / <i>landscape</i> ).
Pagebeforecreate	Terpanggil ketika halaman akan diinisialisasi dan sebelum <i>jQuery mobile</i> mulai mengembangkan halaman.
Pagecreate	Terpanggil ketika halaman dibuat tetapi sebelum <i>jQuery mobile</i> selesai mengembangkan halaman.

Scrollstart	Terpanggil ketika <i>scrolling</i> .
Scrollstop	Terpanggil ketika selesai <i>scroll</i> .
Tap	Terpanggil ketika terjadi <i>tap</i> .
Taphold	Terpanggil ketika terjadi <i>tap</i> yang ditahan.
Swipe	Terpanggil ketika <i>user</i> melakukan <i>swipe</i> terhadap suatu elemen secara horizontal.
Swipeleft	Terpanggil ketika <i>user</i> melakukan <i>swipe</i> terhadap suatu elemen ke arah kiri.
Swiperight	Terpanggil ketika <i>user</i> melakukan <i>swipe</i> terhadap suatu elemen ke arah kanan.

Berikut adalah contoh penggunaan *event* pada *jQuery mobile*:

```

<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
    href="http://code.jquery.com/mobile/1.2.0/jquery.mobile-1.2.0.min.css">
  <script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
  <script src="http://code.jquery.com/mobile/1.2.0/jquery.mobile-1.2.0.min.js"></script>
  <script>
    //Fungsi akan terpanggil ketika halaman selesai dibuat
    $(document).on("pagecreate",function(){
      alert("Page has been created!");
    });
    //Fungsi akan terpanggil ketika halaman sedang dibuat
    $(document).on("pagebeforecreate",function(){
      alert("Page before created!");
    });
    //Fungsi akan terpanggil ketika user melakukan scroll
    $(document).on("scrollstart",function(){
      alert("Start Scrolling");
    });
  </script>
</head>
<body>
  <div data-role="page" data-theme="a">
    <div data-role="header">
      <h1>jQuery Mobile</h1>
    </div>
    <div data-role="content">
      <p>
        Berikut adalah contoh penggunaan beberapa event yang
        disediakan oleh jQuery mobile.
      </p>
    </div>
  </div>
</body>
</html>

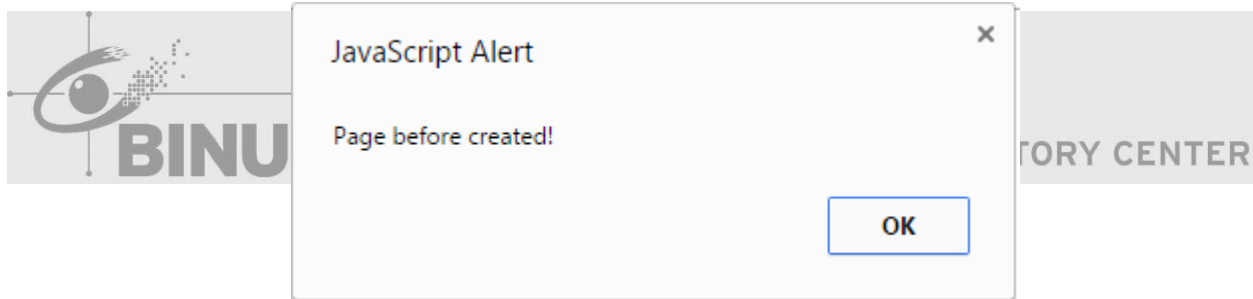
```

```

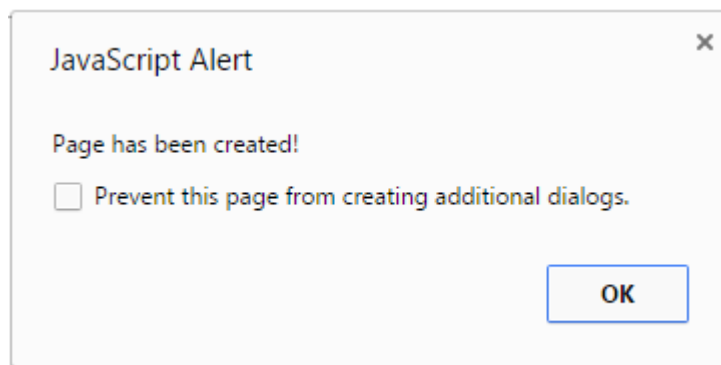
        </p>
        <ul data-role="listview" id="year">
            <li data-role="list-divider">Tahun</li>
        </ul>
        <a href="#" data-role="button" data-icon="refresh">Refresh
    </a>
</div>
<div data-role="footer">
    <h4>&copy; TM14-0</h4>
</div>
</div>
</body>
<script>
    //Menambahkan isi dari data year
    var tahun = document.getElementById("year");
    for(var i=1;i<=100;i++)
    {
        tahun.innerHTML += "<li><a href='#'>" + (1900+i) + "</a></li>";
    }
</script>
</html>

```

Berikut hasil dari *code*:



**Gambar 3.3 Tampilan sebelum halaman dibuat**

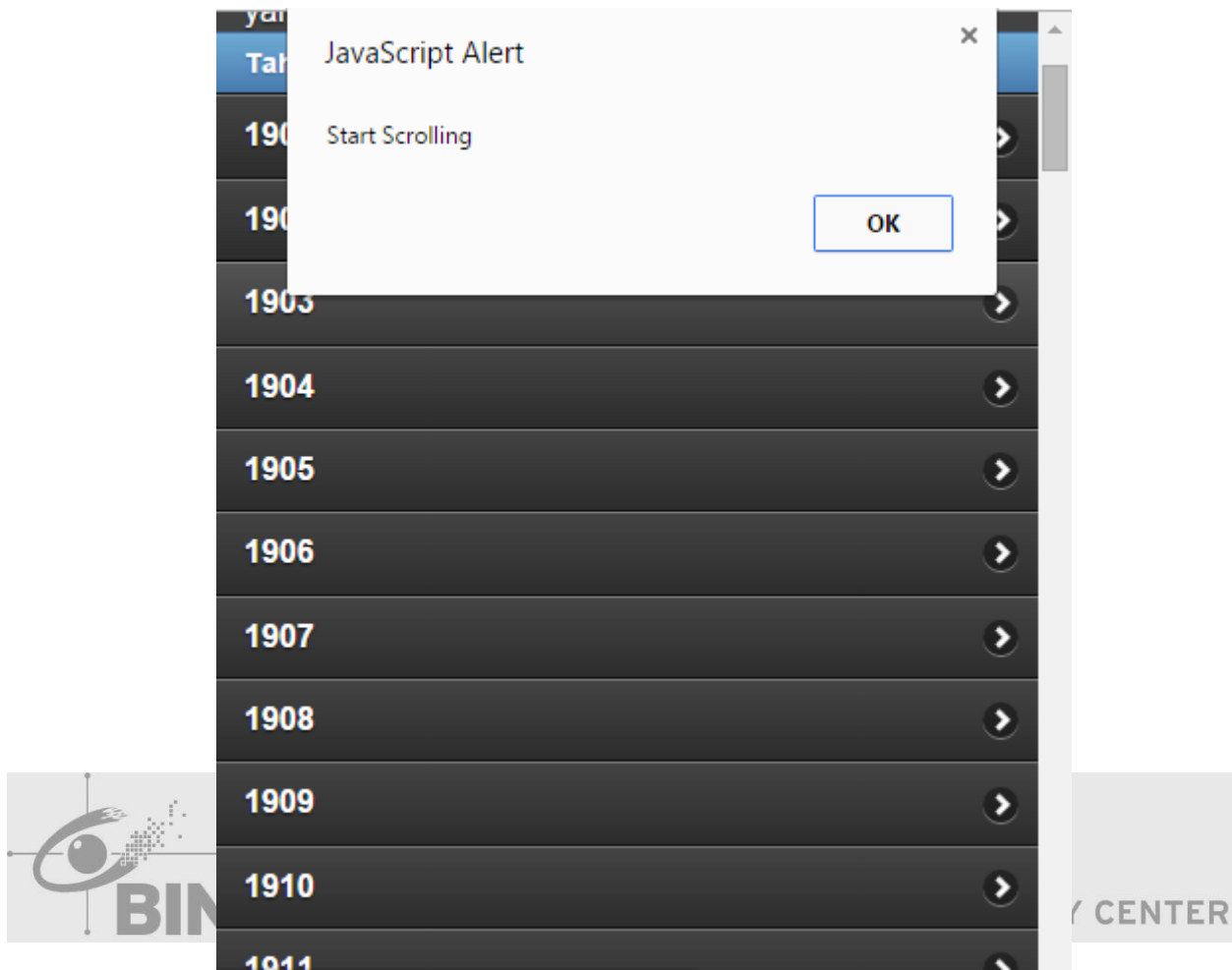


**Gambar 3.4 Tampilan setelah halaman dibuat**



Gambar 3.5 Tampilan halaman





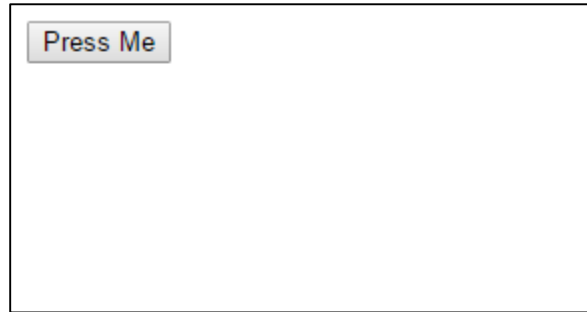
Gambar 3.6 Tampilan saat melakukan *scroll* pada halaman

### 3.9 Exercise

Buatlah sebuah halaman *web* dengan nama “**registration.html**” yang terdapat satu tombol, dan ketika tombol tersebut ditekan akan menampilkan judul “**Registration Form**” dan sebuah *form* yang berisi tiga buah inputan untuk teks yaitu *email*, *password*, *confirmation password* dan satu buah tombol “**Press Me**” dimana ketika tombol ditekan (menggunakan **onclick**), akan melakukan validasi, yaitu :

1. *Email* harus mengandung ‘@’
2. *Email* harus mengandung ‘.’
3. *Password* harus diantara 5 – 15 karakter
4. *Password* dan *confirmation password* harus sama

Apabila sudah sesuai, maka akan menampilkan pesan sukses dan *email* yang diubah menjadi huruf kapital. Berikut adalah hasilnya:



Gambar 3.7 Tampilan registration.html (sebelum ditekan)



Gambar 3.8 Tampilan registration.html (setelah tombol ditekan)

Berikut adalah *code* dari “registration.html”:

```
<!DOCTYPE html>
<html>
<head>
<script>
    function check()
    {
        var email = document.getElementById("txtemail").value;
        var pass = document.getElementById("txtpass").value;
        var con = document.getElementById("txtconpass").value;

        if(email.indexOf('@')== -1)
        {
            //Apabila tidak ditemukan karakter @
            alert("Email harus mengandung 1 karakter '@' ");
        }else if(email.indexOf('.')== -1)
        {
            //Apabila tidak ditemukan karakter .
            alert("Email harus mengandung 1 karakter '.' ");
        }else if(pass.length < 5 || pass.length > 15)
        {
            //Apabila panjang password tidak sesuai
        }
    }
}
```

```

        alert("Password harus terdiri dari 5 sampai 15 karakter! ");
    }else if(pass != con)
    {
        //Apabila password dan confirmation password tidak sama
        alert("Password dan konfirmasi password harus sama!");
    }else
    {
        //Apabila sukses dan menampilkan email dalam huruf kapital
        alert("Registrasi sukses! berikut adalah email anda :
        "+email.toUpperCase());
    }
}
function press()
{
    //Menampilkan form yang disembunyikan
    document.getElementById("form").style.display = "block" ;
    //Menyembunyikan tombol pertama
    document.getElementById("button").style.display = "none";
}
</script>
</head>
<body>
<div id="button">
    <button onclick="press()">Press Me</button>
</div>
<div id="form" style="display:none;">
<h1> Registration Form </h1>
<table>
    <tr>
        <td> Email </td>
        <td> <input type="text" id="txtemail"/></td>
    </tr>
    <tr>
        <td> Password </td>
        <td> <input type="password" id="txtpass"/></td>
    </tr>
    <tr>
        <td> Confirmation Password </td>
        <td> <input type="password" id="txtconpass"/></td>
    </tr>
</table>
    <button onclick="check()">Submit</button>
</div>
</body>
</html>

```

## **BAB 4**

# **JAVASCRIPT – OBJECT ORIENTED PROGRAMMING**



*Object oriented programming* (OOP) adalah paradigma dalam pemrograman yang menggunakan abstraksi dan berorientasi terhadap objek dimana suatu data (atribut dan metode) disimpan dalam suatu struktur data yang dinamakan dengan *class*. Abstraksi adalah suatu cara untuk mengurangi kompleksitas dan meningkatkan efisiensi desain dan implementasi dalam suatu sistem piranti lunak. Pemrograman berorientasi objek ini menggunakan beberapa paradigma yang sudah pernah dikembangkan serta menjadi fitur utama *OOP* itu sendiri, seperti *encapsulation*, *inheritance* dan *polymorphism*. *JavaScript* adalah salah satu bahasa pemrograman berbasis *object oriented* yang memiliki kemampuan untuk membuat aplikasi dari kedua sisi, yaitu sisi klien dan sisi *server*. Dalam paradigma *object oriented*, *class* adalah template dari suatu objek, sehingga *class* hanya berupa rancangan atau kerangka. *Object* adalah hasil yang dihasilkan dari *class* tersebut.

#### 4.1 Encapsulation

Merupakan salah satu fitur penting dalam *OOP* dimana data dan fungsi dibungkus menjadi satu komponen yang dinamakan dengan *class*. Dalam satu kelas terdiri dari *method/fungsi* dan *property/atribut*. *Method* adalah kemampuan dari objek tersebut atau fungsi yang terasosiasi dengan objek tersebut, sedangkan atribut adalah karakteristik yang dimiliki oleh objek tersebut.

*Class* pada *JavaScript* memiliki **constructor**, yaitu *method* yang terpanggil ketika sebuah objek dari *class* tersebut dibuat. Dikarenakan *JavaScript* adalah bahasa yang berbasis *prototype*, maka pembuatan *class* tidak menggunakan *statement class* melainkan sama seperti mendefinisikan suatu fungsi. Bahasa yang berbasis **prototype** adalah cara pemrograman *OOP* dimana penggunaan kembali kode yang sudah dibuat dengan cara membuat *clone* dari objek yang sudah ada sebagai *prototype*. Ada dua tipe penulisan data-data yang terletak didalam *class* pada *JavaScript*:

1. Menggunakan *pointer this*, dilakukan didalam *constructor*:

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript OOP</title>
</head>
<body>
```

```


<script>
  //Membuat kelas manusia
  function Karyawan(nama,umur,gaji)
  {
    //atribut-atribut kelas manusia
    this.nama = nama;
    this.umur = umur;

    //method kelas manusia
    this.halo = function()
    {
      alert("Halo, Teman-teman");
    }
  }

  //Membuat objek dari kelas manusia
  var budi = new Karyawan("Budiman",20);
  //Memanggil method
  budi.halo();
</script>
</body>
</html>

```

## 2. Menggunakan *prototype*, dilakukan diluar *constructor*:



```

<!DOCTYPE html>
<html>
<head>
  <title>JavaScript: OOP</title>
</head>
<body>
  <script>
    //Membuat kelas manusia
    function Karyawan(nama,umur,gaji)
    {
      //atribut-atribut kelas manusia
      this.nama = nama;
      this.umur = umur;
    }

    //method kelas manusia dengan prototype
    Manusia.prototype.halo = function()
    {
      alert("Halo, Teman-teman");
    }

    //Membuat objek dari kelas manusia
    var budi = new Karyawan("Budiman",20);
    //Memanggil method
    budi.halo();
  </script>
</body>
</html>

```

Berikut adalah contoh penggunaan *method* dengan dan tanpa *return type*:

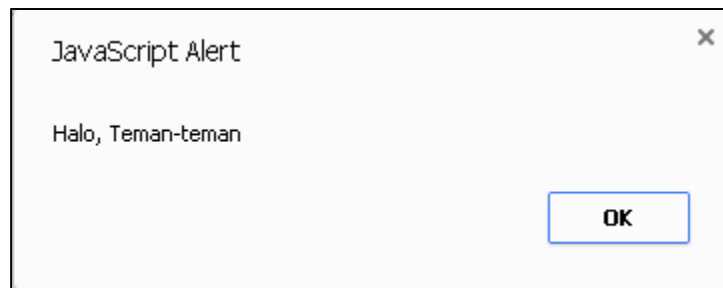
```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Method</title>
</head>
<body>
  <script>
    function Karyawan(nama,umur,gaji)
    {
      this.nama = nama;
      this.umur = umur;
      this.gaji = gaji;

      //Method tanpa return type
      this.halo = function()
      {
        alert("Halo, Teman-teman");
      }

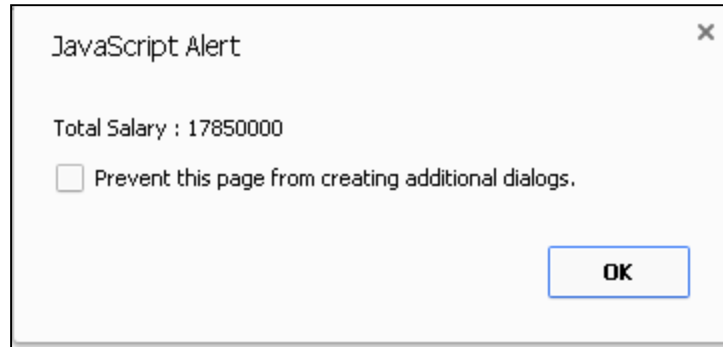
      //Method dengan return type
      Karyawan.prototype.getTotalSalary = function()
      {
        return (this.umur-23)*this.gaji;
      }
    }

    var budi = new Karyawan("Budi",30,2550000);
    //Memanggil method tanpa return type
    budi.halo();

    //Memanggil method dengan return type
    alert("Total Salary : "+budi.getTotalSalary());
  </script>
</body>
</html>
```



**Gambar 4.1** Contoh *method* tanpa *return type*



**Gambar 4.2** Contoh *method* dengan *return type*

## 4.2 Inheritance

Beberapa objek sering sekali mempunyai sesuatu yang sama antara satu dengan yang lain, seperti sepeda, sepeda motor, dan sepeda listrik. Mereka mempunyai karakteristik yang sama antara yang satu dengan yang lain tapi setiap objek tersebut juga mempunyai fitur tambahan yang membedakannya dengan objek lain.

Oleh karena itu, *OOP* memiliki fitur **inheritance** atau pewarisan sifat yang akan menurunkan atribut dan *method* dari kelas orangtuanya/*parent class/base class/super class* sehingga kelas anaknya/*derived class/sub class* memiliki atribut dan *method* yang dimiliki oleh orangtuanya namun juga memiliki atribut dan *method* sendiri yang lebih spesifik. Dengan kata lain, kelas orangtuanya akan bersifat lebih umum dibandingkan kelas anaknya yang lebih spesifik.

Dengan hal ini, fitur *inheritance* meminimalisir jumlah kode yang terduplikasi. Berikut adalah contoh penggunaan *inheritance* dalam *JavaScript*:

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Inheritance</title>
</head>
<body>
  <script type="text/javascript">
    function Manusia(nama,umur)
    {
      this.nama = nama;
      this.umur = umur;
    }

    Manusia.prototype.halo = function()
```



```

{
    alert("Halo, Teman-teman ");
}

//Disini inheritance dilakukan dari kelas Manusia ke Karyawan
Karyawan.prototype = new Manusia();

//Agar kelas karyawan tidak menggunakan Constructor Manusia
Karyawan.prototype.constructor = Karyawan;

//Deklarasi atribut khusus di kelas karyawan
Karyawan.prototype.gaji;
//Deklarasi method khusus di kelas karyawan
Karyawan.prototype.bekerja = function()
{
    alert("Karyawan bekerja dengan keras");
}

//Deklarasi Constructor dari kelas karyawan
function Karyawan(nama,umur,gaji)
{
    this.nama = nama;
    this.umur = umur;
    this.gaji = gaji;
}

andi.halo();
andi.bekerja();
alert("Nama : "+andi.nama);
alert("Umur : "+andi.umur);
alert("Gaji : "+andi.gaji);

</script>
</body>
</html>

```

Pada contoh diatas, kelas **Manusia** memiliki atribut nama dan gaji serta *method* `halo()` dan kelas **Karyawan** adalah kelas turunan dari kelas **Manusia** sehingga kelas **Karyawan** memiliki atribut dan *method* yang diturunkan dari kelas **Manusia** yaitu atribut nama, gaji dan *method* `halo()` serta dispesifikasikan dengan atribut dan *method* yang khusus dimiliki oleh kelas **Karyawan**, yaitu atribut gaji dan *method* `bekerja()`.

### 4.3 Polymorphism

**Polymorphism** merupakan salah satu fitur penting dalam *OOP* selain *encapsulation* dan *inheritance*, apabila diartikan secara umum berarti memiliki banyak bentuk. *Polymorphism* pada *JavaScript* diimplementasikan dalam bentuk *overriding*, yaitu

kemampuan untuk memanggil *method* yang sama di objek yang berbeda namun setiap dari mereka memiliki respon yang berbeda. Sehingga, fitur *polymorphism* ini harus dilengkapi dengan fitur *inheritance* dalam penggunaanya.

Berikut adalah contoh penggunaan *polymorphism* pada *JavaScript*:

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Polymorphism</title>
</head>
<body>
  <script type="text/javascript">

    function Manusia(nama,umur)
    {
      this.nama = nama;
      this.umur = umur;
    }

    Manusia.prototype.halo = function()
    {
      alert("Halo teman-teman, saya Manusia");
    }

    //Disini inheritance dilakukan dari kelas Manusia ke Karyawan
    //Kelas karyawan menggunakan constructor miliknya sendiri
    Karyawan.prototype = new Manusia();
    Karyawan.prototype.constructor = Karyawan;

    function Karyawan(nama,umur,gaji)
    {
      this.nama = nama;
      this.umur = umur;
      this.gaji = gaji;
    }

    //Disini override dilakukan
    //Dimana akan menggantikan method halo yang diturunkan dari kelas Manusia
    Karyawan.prototype.halo = function()
    {
      alert("Halo Kawan, saya karyawan dan sudah bekerja");
    }

    var budi = new Manusia("Budi",20);
    var rio = new Karyawan("Rio",23,1200000);

    budi.halo();

    //rio merupakan objek dari karyawan
    //namun ketika memanggil method halo()
    //method halo() yang terpanggil bukanlah method halo() di kelas manusia
    //namun method halo() yang didefinisikan ulang di kelas Karyawan
```

```
//ini adalah overriding
rio.halo();
</script>
</body>
</html>
```

#### 4.4 Exercise

Buatlah sebuah simulasi pertanian dimana *user* dapat memilih **umur** dan **nama** mereka.

Apabila *user* :

1. **Memilih tua (*old*)**

Maka batas maksimal staminanya adalah 100. Pengalamannya adalah 100.

2. **Memilih muda (*young*)**

Maka batas maksimal staminanya adalah 200. Pengalamannya adalah 50.

Setelah selesai memilih **nama** dan **usia**, maka data *user* akan ditampilkan beserta dengan tiga buah tombol :

1. **Bekerja (*work*)**

Apabila stamina dibawah 50, maka akan menampilkan pesan "**You are too tired!**".

Apabila tidak, maka stamina akan dikurangi dengan 50 dan mendapatkan uang sebagai

berikut:

<p><b>Uang yang didapat = pengalaman * 3</b></p>
--

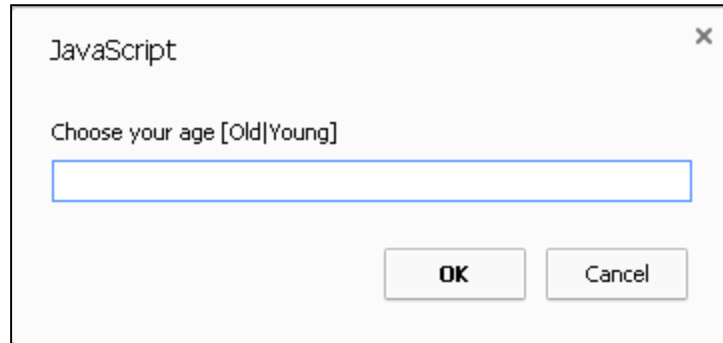
2. **Tidur (*sleep*)**

Akan membuat stamina *user* penuh.

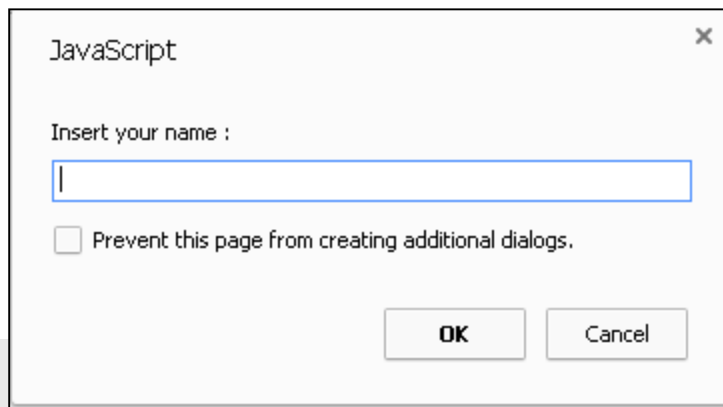
3. **Keluar (*exit*)**

Keluar dari aplikasi.

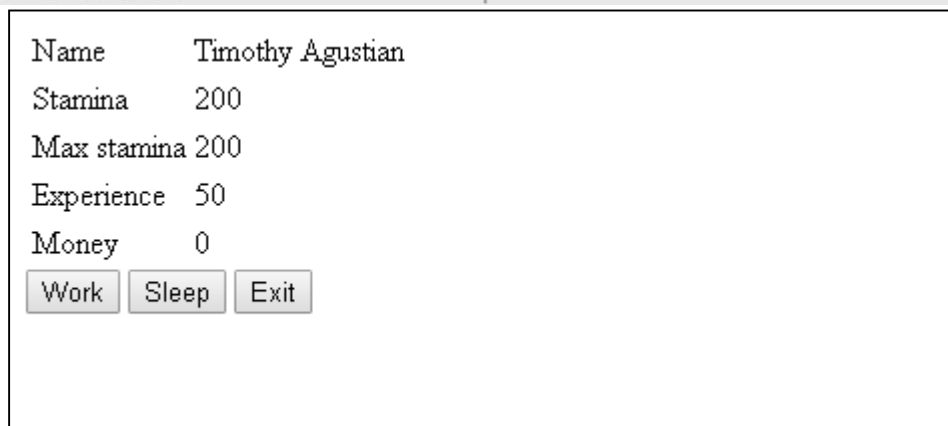
Berikut adalah hasil dari aplikasinya:



**Gambar 4.3 Tampilan saat menginput usia**



**Gambar 4.4 Tampilan saat menginput nama**



**Gambar 4.5 Tampilan setelah melakukan input**

Berikut adalah *code* untuk contoh diatas:

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Exercise</title>
</head>
```

```

<body>
<script>
//Pembuatan kelas farmer
function farmer()
{
    //atribut kelas farmer
    this.stamina;
    this.maxstamina;
    this.exp;
    this.money;
    this.name;

    //method kelas farmer
    this.work = function()
    {
        this.stamina -= 50;
        this.money = this.exp * 3;
    }
    this.sleep = function()
    {
        this.stamina = this.maxstamina
    }
}
//Pembuatan kelas old sebagai turunan dari farmer
old.prototype = new farmer();
old.prototype.constructor = old;
//Constructor untuk kelas old
function old()
{
    this.maxstamina = 100;
    this.stamina = this.maxstamina;
    this.money = 0;
    this.exp = 100;
}
//Pembuatan kelas young sebagai turunan dari farmer
young.prototype = new farmer();
young.prototype.constructor = young;
//Constructor untuk kelas young
function young()
{
    this.maxstamina = 200;
    this.stamina = this.maxstamina;
    this.money = 0;
    this.exp = 50;
}

//Meminta inputan dan validasi usia (old/young)
do
{
    jawaban = prompt("Choose your age [Old|Young]", "");
}
while(jawaban!="Old" && jawaban!="Young");
//Membuat objek global sesuai dengan usianya
if(jawaban == "Old")
{
    hero = new old();
}

```

```

    }
    else
    {
        hero = new young();
    }
    //Meminta inputan nama
    hero.name = prompt("Insert your name : ");
    //fungsi refresh untuk update tampilan html sesuai dengan objek yang ada
    function refresh()
    {
        document.getElementById("name").innerHTML = hero.name;
        document.getElementById("stamina").innerHTML = hero.stamina;
        document.getElementById("maxstamina").innerHTML = hero.maxstamina;
        document.getElementById("exp").innerHTML = hero.exp;
        document.getElementById("money").innerHTML = hero.money;
    }
    //function work untuk menambah uang dan mengurangi stamina disertai
    validasi
    function work()
    {
        if(hero.stamina < 50)
        {
            alert("You are too tired!");
        }
        else
        {
            hero.stamina-=50;
            hero.money = hero.money + (hero.exp * 3);
        }
        refresh();
    }
    //function sleep untuk membuat stamina menjadi penuh
    function sleep()
    {
        hero.stamina = hero.maxstamina;
        alert("Your stamina is full!!");

        refresh();
    }

    //function exit untuk menutup program
    function exit()
    {
        alert("Thank you for playing");
        close();
    }

</script>

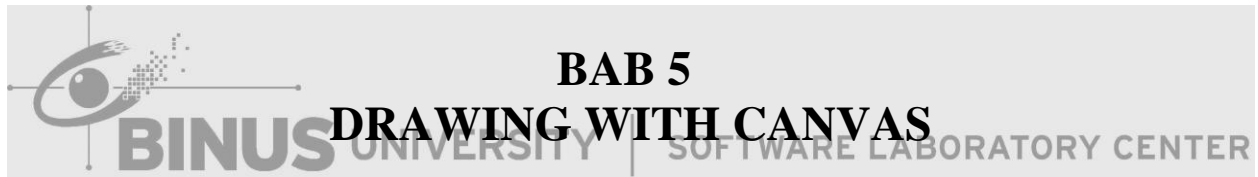
<table>
<tr>
    <td>Name</td>
    <td id="name"></td>
</tr>

```

```
<tr>
  <td>Stamina</td>
  <td id="stamina"></td>
</tr>
<tr>
  <td>Max stamina</td>
  <td id="maxstamina"></td>
</tr>
<tr>
  <td>Experience</td>
  <td id="exp"></td>
</tr>
<tr>
  <td>Money</td>
  <td id="money"></td>
</tr>
</table>
<button onclick="work()"> Work </button>
<button onclick="sleep()"> Sleep </button>
<button onclick="exit()"> Exit </button>

<script>
//function refresh terpanggil ketika semua halaman sudah di load
refresh();
</script>

</body>
</html>
```





## 5.1 Introduction of Canvas

**Canvas** merupakan salah satu *tag* baru yang ada di HTML5. *Canvas* menawarkan sebuah kemudahan untuk menggambar grafis dalam sebuah halaman web, misalnya menggambar kotak, lingkaran, kurva, dan lainnya. Pada dasarnya, bentuk *default* dari tag `<canvas>` hampir sama dengan tag `<div>` yaitu sebuah persegi empat (*rectangle*). Proses menggambar pada *canvas* dilakukan melalui bantuan *JavaScript*, *canvas* hanya sebagai *container* grafisnya saja.

Berikut adalah daftar *browser* minimum yang sudah mendukung penggunaan tag `<canvas>`:

Browser	Versi Minimum
Google Chrome	4.0
Internet Explorer	9.0
Mozilla Firefox	2.0
Apple Safari	3.1
Opera	9.0

Secara *default*, *canvas* tidak memiliki *border* dan konten. Jadi, ketika mendeklarasikan sebuah tag `<canvas>`, perlu untuk mengubah *width* dan *height* dari *canvas* tersebut dengan menggunakan atribut yang ada pada HTML5. Anda juga dapat menambahkan *border* atau warna *background* agar wujud *canvas* dapat terlihat dengan jelas. Berikan sebuah *id* kepada elemen *canvas* yang telah dibuat agar elemen tersebut dapat diakses oleh *script*.

Contoh *script* untuk mendeklarasikan sebuah *canvas* dengan *id* **myCanvas** pada halaman HTML:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: black; /* Memberikan warna
background dasar ke halaman web */
```

```

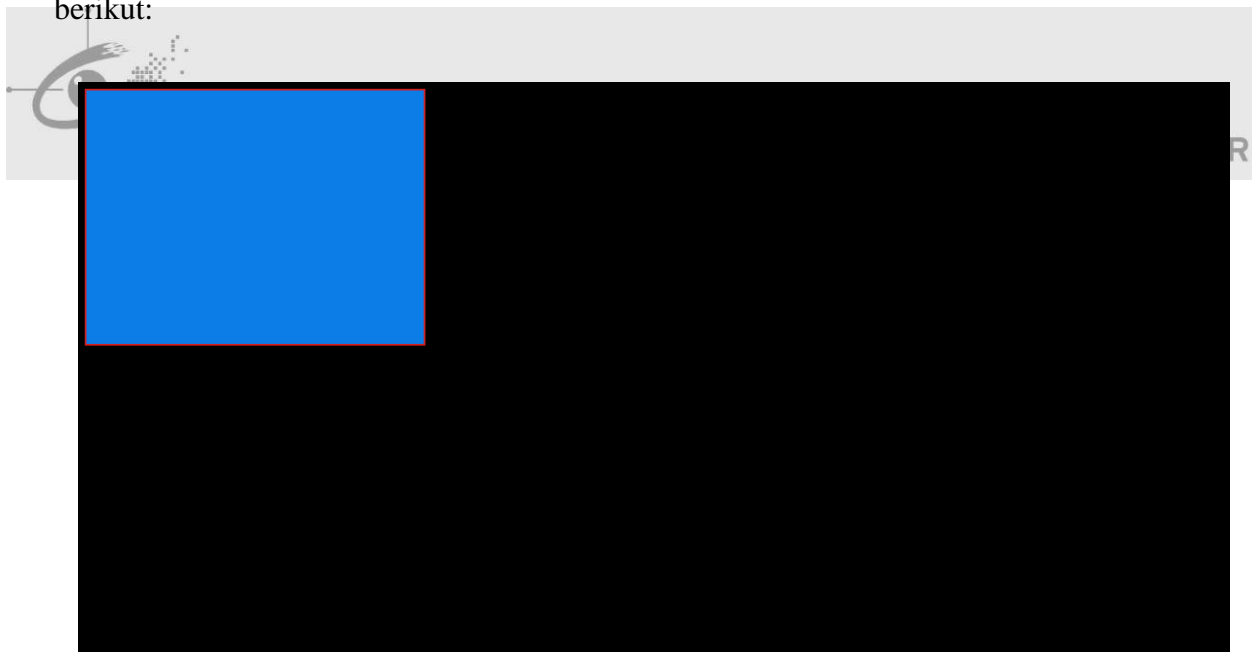
    }
  </style>
</head>
<body>

  <!-- Tag pembuka canvas -->
  <canvas id="myCanvas" width="400" height="300" style="background-
color:#0c7de6; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>
  <!-- Tag penutup canvas -->

</body>
</html>

```

Di dalam tag `<canvas></canvas>` terdapat *exception handling* apabila *browser* yang digunakan belum mendukung penggunaan *canvas*. Pada contoh diatas, akan muncul kalimat **“Browser anda belum mendukung HTML5 canvas.”** apabila *browser* yang digunakan ternyata belum mendukung *canvas*. Dari contoh di atas, hasilnya akan seperti gambar berikut:



**Gambar 5.1 Membuat *canvas* sederhana**

Untuk menggambar di dalam sebuah *canvas*, perlu untuk menggunakan *script*. *Script* yang akan digunakan pada pemahasan ini adalah *JavaScript*. *Method* pada *JavaScript* yang digunakan untuk mulai menggambar yaitu `getContext()`. *Method* tersebut akan mengembalikan sebuah *object JavaScript* yang memiliki banyak sekali *method* dan *property*

yang dapat digunakan untuk menggambar pada sebuah *canvas*. Pada *sub-bab* ini akan dibahas mengenai bentuk-bentuk dasar dua dimensi yang dapat digambar pada *canvas* seperti *rectangle* (kotak), *circle* (lingkaran), dan lainnya.

Untuk mengakses elemen *canvas* yang telah dibuat dengan menggunakan *JavaScript*:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: black;
    }
  </style>

  <!-- JavaScript -->
  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas"); //
      var context = canvas.getContext("2d"); // Membuat
      // Membuat objek canvas dari elemen canvas yang telah dibuat sebelumnya
      // objek context 2D dari canvas
    }
  </script>
</head>
<body>
  <canvas id="myCanvas" width="400" height="300" style="background-
  color:#0c7de6; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>
</body>
</html>
```

Tambahan:

- 1) *Method* `getElementById()` digunakan untuk mengakses elemen pada dokumen HTML dengan menggunakan *id* yang telah dispesifikasi di dalam parameter *method* tersebut. Parameter *id* tersebut tidak boleh kosong (wajib diisi) yang merupakan *id* milik elemen HTML yang ingin diakses.
- 2) *Method* pada *event* `window.onload` akan dijalankan ketika sebuah *object* sudah selesai dimuat (*load*), pada bagian ini *object* yang dimasud adalah *window*. Oleh karena itu, perlu untuk memanggil elemen *canvas* di dalam *method* `onload` agar tidak terjadi *null object*

*exception* yang terjadi karena elemen *canvas* yang dideklarasikan berada di bawah *script* yang memanggilnya.

## 5.2 Drawing Shapes & Path – Fills & Stroke

*Method fill()* merupakan salah satu *method* dari *object context* yang telah dibuat pada contoh *script* sebelumnya, *method* ini digunakan untuk memberikan warna pada bentuk gambar yang telah dibuat. Secara *default* warnanya adalah hitam. *Method* ini tidak memiliki parameter sama sekali. Untuk mengatur *style* yang ingin diterapkan, dapat menggunakan *property fillStyle*. Setelah itu kemudian memanggil *method fill()* untuk menerapkan *style* tersebut.

Misalnya, untuk membuat sebuah kotak berwarna, sebelum memanggil *method fill()*, harus dibuat terlebih dahulu *object* kotak dengan menggunakan fungsi *rect()*, setelah itu lakukan *fill* dengan sebuah warna.

Untuk *property fillStyle*, dapat menerima 3 jenis *value* yaitu:

Value	Deskripsi
<b>color</b>	Diisi dengan <i>value</i> warna yang digunakan oleh CSS. Secara <i>default</i> bernilai #000000 (hitam).
<b>gradient</b>	Diisi dengan <i>gradient object</i> apabila ingin menggunakan tekstur gradasi untuk metode <i>fill</i> -nya.
<b>pattern</b>	Diisi dengan <i>pattern object</i> apabila ingin menggunakan tekstur berbentuk <i>pattern</i> untuk metode <i>fill</i> -nya.

Untuk penjelasan *method rect()* akan dibahas lebih lanjut di *sub-bab* berikutnya.

Contoh *code* membuat sebuah *rectangle* berwarna:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: black;
    }
  </style>
```

```

<script type="text/javascript">
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");

        context.fillStyle = "purple"; // Memberikan sebuah
        style warna "ungu" kepada method fill()
        context.rect(10, 10, 200, 100); // Membuat bentuk
        rectangle terlebih dahulu yang akan digambar nanti
        context.fill(); // Menerapkan style tersebut kepada
        rectangle
    }
</script>
</head>
<body>

    <canvas id="myCanvas" width="400" height="300" style="background-
    color:#0c7de6; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Contoh di atas akan menghasilkan bentuk *rectangle* berwarna ungu di dalam *canvas* seperti berikut ini:



**Gambar 5.2 Membuat persegi berwarna ungu**

Untuk memberikan *stroke* (garis tepi) pada bentuk yang telah digambar, dapat menggunakan *method* `stroke()`. Sama seperti `fill()`, `stroke()` juga harus didefinisikan

terlebih dahulu *style*-nya. Jika pada `fill()` menggunakan *property* `fillStyle`, pada `stroke()` juga tidak jauh berbeda yaitu menggunakan *property* yang bernama `strokeStyle`. *Value* yang dapat diterima oleh `strokeStyle` juga sama dengan *value* pada `fillStyle` yaitu ada *color*, *gradient*, dan *pattern*.

Berikut contoh *code* membuat sebuah *rectangle* yang memiliki *stroke*.

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: black;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      context.fillStyle = "purple";
      context.rect(10, 10, 200, 100);
      context.fill();

      context.strokeStyle = "white"; // Memberikan sebuah
      style "warna putih" kepada method stroke()
      context.lineWidth = 5; // (Optional) mengatur
      ketebalan dari stroke
      context.stroke(); // Membuat stroke dengan style warna
      "putih" untuk rectangle
    }
  </script>
</head>
<body>

  <canvas id="myCanvas" width="400" height="300" style="background-
  color:#0c7de6; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>

</body>
</html>
```

*Code* tersebut akan menambahkan sebuah *stroke* berwarna **putih** dengan ketebalan **5 piksel** pada *rectangle* ungu yang telah di buat sebelumnya seperti berikut ini:



**Gambar 5.3 Menggunakan *stroke* pada *canvas***

Untuk membuat sebuah garis pada *canvas* dapat menggunakan *method* `lineTo()`. *Method* ini tidak menggambar garis tersebut, hanya menentukan sebuah titik baru dan membuat sebuah garis imajiner (*visual path*) dari titik tersebut ke titik terakhir yang telah ditentukan di dalam *canvas*. Untuk menggambar *path* yang telah dibuat, gunakan *method* `stroke()`. Untuk memindahkan posisi ke suatu koordinat tertentu, dapat menggunakan *method* `moveTo()`.

Bentuk umum dari *method* `lineTo()` adalah seperti ini:

```
context.lineTo(x, y);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>x</b>	Koordinat x tujuan <i>line</i>
<b>y</b>	Koordinat y tujuan <i>line</i>

Berikut contoh *code* penggunaan *method* `lineTo()`:

```
<!DOCTYPE html>
<html lang="id">
```

```

<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      context.beginPath(); // Memulai sebuah path drawing
      context.moveTo(20, 60); // Memindahkan posisi current
      context.lineTo(40, 80); // Membuat line path ke posisi
      context.lineTo(170, 10); // Melanjutkan line path ke
      context.stroke(); // Memberikan stroke pada path line

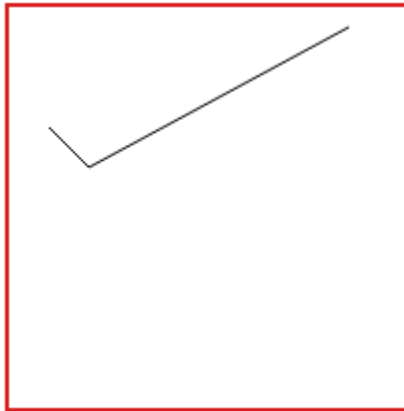
    }
  </script>
</head>
<body>
  <canvas id="myCanvas" width="200" height="200" style="background-
  color:#ffffff; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>

</body>
</html>

```



Hasil dari *code* tersebut yaitu:



**Gambar 5.4 Membuat *line* sederhana**

*Line* pada *canvas* memiliki beberapa *extended styles* yang dapat digunakan untuk memodifikasi *line* yang telah dibuat agar terlihat lebih bagus dan menarik, diantaranya:

**a. *lineWidth***

Properti ini digunakan untuk mengatur ketebalan (*width*) dari *line* yang akan buat (dalam *pixels*). Properti ini hanya menerima satu buah *value* yaitu *width* (dalam *pixels*). Secara *default*, nilainya adalah 1 piksel.

Contoh penggunaan properti *lineWidth*:

SOFTWARE LABORATORY CENTER

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenalan HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      context.lineWidth = 30; // Mengatur ketebalan garis
      context.beginPath();
      context.moveTo(20, 60);
      context.lineTo(180, 60);
      context.stroke();
    }
  </script>
</head>
</html>
```

menjadi 30

```

    }
  </script>
</head>
<body>

  <canvas id="myCanvas" width="200" height="200"
style="background-color:#ffffff; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>

</body>
</html>

```

Hasil dari *code*:



Gambar 5.5 Penggunaan *line width*

#### b. *lineCap*

Properti ini digunakan untuk mengubah bentuk akhir *cap* dari suatu *line*. Properti ini menerima tiga opsi *value* yaitu *butt*, *round*, dan *square*. Secara *default* tipenya adalah *butt*. Khusus untuk *round* dan *square* membuat *line* sedikit lebih panjang dibanding *butt*.

Contoh penggunaan properti *lineCap*:

```

<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

```

```

</style>

<script type="text/javascript">
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");

        context.lineWidth = 8; // Mengatur ketebalan line
        // round
        context.beginPath();
        context.lineCap = "round";
        context.moveTo(20, 60);
        context.lineTo(40, 80);
        context.lineTo(170, 10);
        context.stroke();

        // square
        context.beginPath();
        context.lineCap = "square";
        context.moveTo(20, 100);
        context.lineTo(170, 100);
        context.stroke();

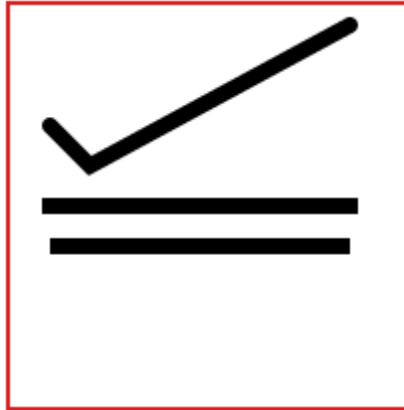
        // butt
        context.beginPath();
        context.lineCap = "butt"; // Dengan menggunakan
        // butt, line akan terlihat sedikit lebih pendek dibanding round dan
        // square
        context.moveTo(20, 120);
        context.lineTo(170, 120);
        context.stroke();
    }
</script>
</head>
<body>

    <canvas id="myCanvas" width="200" height="200"
    style="background-color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Hasil dari *code* tersebut:



Gambar 5.6 Penggunaan *line cap*

### c. `lineJoin`

Properti ini digunakan untuk mengubah *style* sudut yang terbentuk bila dua buah garis bertemu. Properti ini menerima tiga opsi *value*, yaitu *bevel*, *round*, dan *miter*. Secara *default* adalah *miter*. *Value* dari *miter* ditentukan dengan properti *miterLimit*.

Contoh penggunaan properti `lineJoin`:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      context.lineWidth = 25;

      // round
      context.beginPath();
      context.lineJoin = "round";
      context.moveTo(20, 60);
      context.lineTo(40, 80);
      context.lineTo(170, 10);
      context.stroke();
    }
  </script>
</head>
</html>
```

```

        // bevel
        context.beginPath();
        context.lineJoin = "bevel";
        context.moveTo(20, 110);
        context.lineTo(40, 130);
        context.lineTo(170, 60);
        context.stroke();

        // miter
        context.beginPath();
        context.lineCap = "miter";
        context.moveTo(20, 160);
        context.lineTo(40, 180);
        context.lineTo(170, 110);
        context.stroke();
    }
</script>
</head>
<body>

    <canvas id="myCanvas" width="200" height="200"
    style="background-color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Hasil dari *code*:



Gambar 5.7 Penggunaan *line join*

#### d. *miterLimit*

Properti ini digunakan untuk mengatur panjang maksimal dari *miter*. Panjang *miter* adalah jarak antara simpangan dalam dan simpangan luar yang terbentuk bila dua buah

garis bertemu. Properti ini hanya dapat digunakan bila properti `lineJoin` berisi *value* “`miter`”. Panjang *miter* semakin besar apabila sudut simpangan semakin kecil. Untuk mencegah panjang *miter* yang terlalu panjang dapat menggunakan properti ini. Jika panjang *miter* sudah melebihi batas maksimum, maka simpangan yang terbentuk akan ditampilkan sebagai `lineJoin` yang bertipe “`bevel`”. Properti ini hanya menerima satu buah *value* yaitu nilai dari `miterLimit` yang diinginkan. Secara *default* nilainya adalah 10.

Contoh penggunaan properti `miterLimit`:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");
      context.lineWidth = 12;

      // miter
      context.beginPath();
      context.lineJoin = "miter";
      context.miterLimit = 4; // Mengatur batas panjang
      miter dengan value = 4
      context.moveTo(20, 20);
      context.lineTo(50, 30);
      context.lineTo(20, 35);
      context.stroke();

      context.beginPath();
      context.lineJoin = "miter";
      context.miterLimit = 7; // Mengatur batas panjang
      miter dengan value = 7
      context.moveTo(20, 50);
      context.lineTo(50, 60);
      context.lineTo(20, 65);
      context.stroke();
    }
  </script>
</head>
</html>
```

```

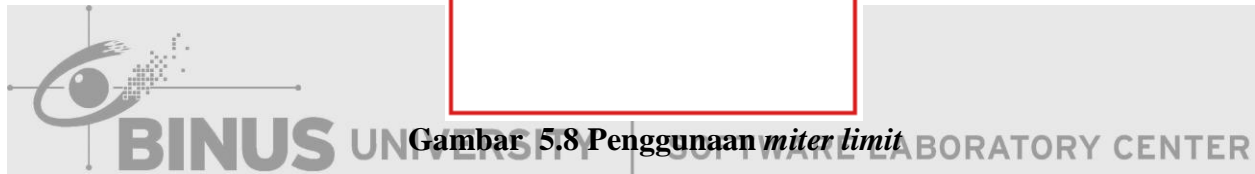
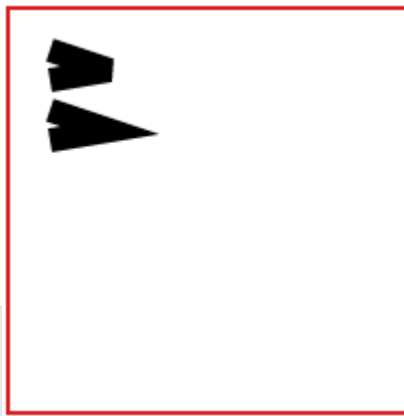
    </script>
</head>
<body>

    <canvas id="myCanvas" width="200" height="200"
style="background-color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Hasil dari *code*:



Gambar 5.8 Penggunaan *miter limit*

### 5.3 Drawing Shapes & Path – Rectangle

Untuk menggambar sebuah *rectangle* pada *canvas*, dapat menggunakan *method* `rect()`. *Method* ini jelas lebih mudah daripada membuat *rectangle* dengan menghubungkan empat buah garis lurus. Terdapat empat macam *method* umum untuk membuat dan memodifikasi *rectangle* pada *canvas*, yaitu sebagai berikut:

Method	Deskripsi
<code>rect(x, y, width, height)</code>	Membuat sebuah <i>rectangle</i> (belum termasuk <i>fill</i> ).
<code>fillRect(x, y, width, height)</code>	Membuat sebuah <i>rectangle</i> yang <i>auto-fill</i> .
<code>strokeRect(x, y, width, height)</code>	Membuat sebuah <i>rectangle</i> yang tidak ter- <i>fill</i> , hanya memiliki garis tepi ( <i>stroke</i> ).
<code>clearRect(x, y, width, height)</code>	Menghapus beberapa <i>pixels</i> yang terdapat di dalam sebuah <i>rectangle</i> .
Keempat method tersebut memiliki parameter yang sama persis yaitu <b>posisi x</b> , <b>posisi y</b> , <b>width</b> , dan	

*height* dari *rectangle*. Posisi x dan y tersebut mengacu pada posisi sudut kiri atas *rectangle*.

Berikut contoh *code* penggunaannya:

**a. Menggunakan *method* `rect()`**

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      // rect()
      context.rect(10, 10, 400, 200); // Membuat sebuah
      // rectangle di posisi (10,10) dengan width 400 dan height 200 (hanya
      // membuat saja, tetapi belum ter-fill, perlu method fill())
      context.fillStyle = "#3b5998";
      context.fill();
    }
  </script>
</head>
<body>

  <canvas id="myCanvas" width="800" height="600"
  style="background-color:#ffffff; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>

</body>
</html>
```

Hasil dari *code* tersebut:





Gambar 5.9 Membuat *rectangle* menggunakan *method* `rect()`

b. Menggunakan *method* `fillRect()`

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      // fillRect()
      context.fillStyle = "#3b5998";
      context.fillRect(10, 10, 400, 200); // Membuat
      // filled-rectangle di posisi (10, 10) dengan width 400 dan height 200
    }
  </script>
</head>
<body>

  <canvas id="myCanvas" width="800" height="600"
  style="background-color: #ffffff; border: 2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>

</body>
```

```
</html>
```

Hasil dari *code* tersebut:



**Gambar 5.10** Membuat *rectangle* menggunakan *method* `fillRect()`

**c. Menggunakan *method* `strokeRect()`**

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      // strokeRect()
      context.strokeStyle = "#3b5998";
      context.strokeRect(10, 10, 400, 200); // Membuat
      stroked-rectangle di posisi (10, 10) dengan width 400 dan height 200
      (tidak ter-fill)
    }
  </script>
</head>
<body>

  <canvas id="myCanvas" width="800" height="600"
  style="background-color:#ffffff; border:2px solid #d81e1e;">
```

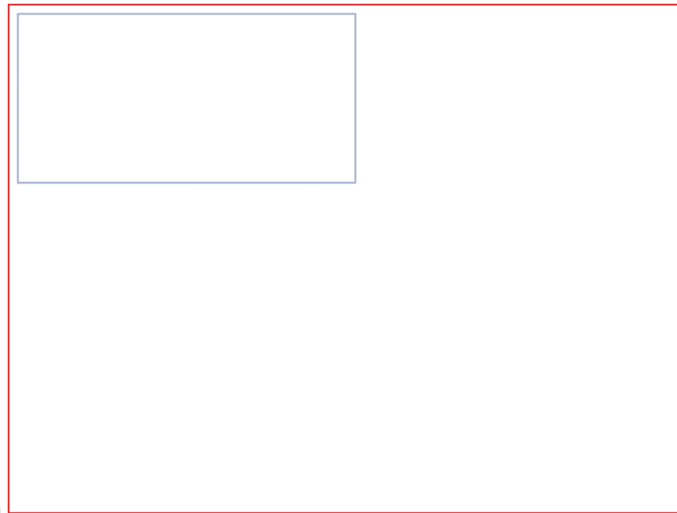
```

        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Hasil dari *code*:



Gambar 5.11 Membuat *rectangle* menggunakan *method* *strokeRect()*

#### d. Menggunakan *method* *clearRect()*

```

<!DOCTYPE html>
<html lang="id">
<head>
    <title>Pengenal HTML5 Canvas</title>
    <style>
        body {
            background-color: white;
        }
    </style>

    <script type="text/javascript">
        window.onload = function() {
            var canvas = document.getElementById("myCanvas");
            var context = canvas.getContext("2d");

            // clearRect()
            context.fillStyle = "#3b5998";
            context.fillRect(10, 10, 400, 200);
            context.clearRect(20, 20, 80, 80); // Menghapus
            sebagian pixel di posisi (20,20) dengan width 80 dan height 80 dari
            rectangle
        }
    </script>

```

```

</head>
<body>

    <canvas id="myCanvas" width="800" height="600"
style="background-color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Hasil dari *code*:



Gambar 5.12 Membuat *rectangle* menggunakan *method* `clearRect()`

## 5.4 Drawing Shapes & Path - Arc and Circles

Untuk menggambar sebuah *arc* (*curve/kurva*), dapat menggunakan *method* `arc()`. *Method* ini juga digunakan untuk menggambar *circle* (lingkaran).

Bentuk umum dari *method* `arc()` adalah seperti ini:

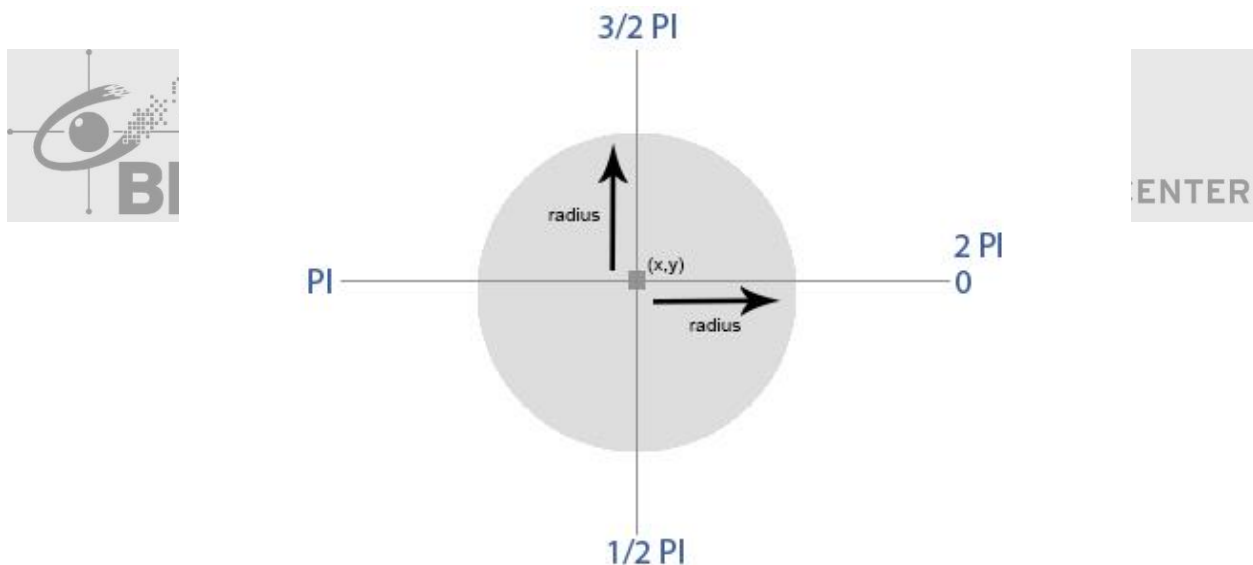
```
context.arc(x, y, r, sAngle, eAngle, counterclockwise);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>x</b>	Posisi x pada titik pusat lingkaran yang akan digambar.
<b>y</b>	Posisi y pada titik pusat lingkaran yang akan digambar.

<b>r</b>	Radius (jari-jari) lingkaran.
<b>sAngle</b>	Sudut mulai digambarnya lingkaran (dalam <i>radian</i> ) dengan sudut 0 mewakili arah jarum jam 3 atau sumbu x positif dalam koordinat kartesian.
<b>eAngle</b>	Sudut akhir digambarnya lingkaran (dalam <i>radian</i> )
<b>counterclockwise</b>	Parameter ini bersifat <i>optional</i> . Menunjukkan arah proses menggambar apakah searah jarum jam atau sebaliknya. Memiliki 2 opsi <i>value</i> , yaitu <i>true</i> atau <i>false</i> . Secara <i>default</i> bernilai <i>false</i> yang berarti <i>clockwise</i> , apabila <i>true</i> artinya <i>counter-clockwise</i> .

Jika digambarkan dalam koordinat *Cartesius*, sudut 0 dan 2 PI (360°) berada pada arah jarum jam 3, sudut 1/2 PI (90°) berada pada arah jarum jam 6, sudut PI (180°) berada pada arah jarum jam 9, dan sudut 3/2 PI (270°) berada pada arah jarum jam 12.



**Gambar 5.13 Parameter *arc* dalam gambar**

Berikut contoh *code* penggunaan *arc()*:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>
</head>
<body>
  <canvas id="myCanvas" width="300px" height="150px">
    </canvas>
  </body>
</html>
```

```

</style>

<!-- JavaScript Here -->
<script type="text/javascript">
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");

        context.beginPath();
        context.fillStyle = "#3b5998";
        context.arc(100, 100, 50, 0, 0.5 * Math.PI, true); //
        3/4 Lingkaran dari sudut 0 hingga 1/2 PI counter-clockwise
        context.fill();
        context.closePath();

        context.beginPath();
        context.fillStyle = "#3b5998";
        context.arc(300, 100, 50, 0, 2 * Math.PI, true); // 1
        Lingkaran penuh
        context.fill();
        context.closePath();

        context.beginPath();
        context.fillStyle = "#3b5998";
        context.arc(500, 100, 50, 0, Math.PI, true); //
        Setengah Lingkaran
        context.fill();
        context.closePath();
    }
</script>
</head>
<body>

    <canvas id="myCanvas" width="800" height="600" style="background-
    color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Tambahan:

1. *Method* `beginPath()` digunakan untuk memulai sebuah *path* baru atau dengan kata lain me-reset path sekarang agar tidak terhubung dengan *path* sebelumnya. Jika tidak menggunakan `beginPath()`, maka gambar lingkaran kedua dan ketiga akan terhubung dengan gambar lingkaran pertama.

2. *Method* `closePath()` digunakan untuk membuat sebuah *path* dari titik sekarang ke titik awal.

Hasil dari *code*:



Gambar 5.14 Membuat *arc*

## 5.5 Drawing Shapes & Path – Gradient

Dalam *canvas*, ada dua jenis metode *fill* secara *gradient* (gradasi) yaitu ***Linear Gradient*** dan ***Radial Gradient***.

### Linear Gradient

Untuk membuat *linear gradient* dapat menggunakan method `createLinearGradient()`. Arah gradasi ditentukan oleh sebuah garis imajiner dan untuk menentukan warna yang digunakan dengan memanggil method `addColorStop()`. Arah gradasi dari *linear gradient* terhitung mulai dari titik awal (*starting point*) hingga titik akhir (*ending point*) garis imajiner yang terdapat pada parameter method `createLinearGradient()`.

Secara umum, bentuk method `createLinearGradient()` yaitu seperti ini:

```
context.createLinearGradient(x0, y0, x1, y1);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>x0</b>	Koordinat x titik dimulainya ( <i>starting point</i> ) <i>gradient</i>
<b>y0</b>	Koordinat y titik dimulainya ( <i>starting point</i> ) <i>gradient</i>
<b>x1</b>	Koordinat x titik diakhirinya ( <i>ending point</i> ) <i>gradient</i>
<b>y1</b>	Koordinat x titik diakhirinya ( <i>ending point</i> ) <i>gradient</i>

Untuk method `addColorStop()`, bentuk umumnya seperti berikut:

```
addColorStop(stop, color);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>stop</b>	Sebuah nilai antara 0.0 hingga 1.0 yang menandakan posisi di antara <i>start</i> (0) dan <i>end</i> (1) dalam sebuah <i>gradient</i> .
<b>color</b>	Kode warna CSS untuk warna pada posisi <i>stop</i> .

Contoh *code* membuat *linear gradient* pada *canvas*:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      var gradient = context.createLinearGradient(0, 0,
canvas.width, 0.25 * canvas.height); // Membuat sebuah objek dari Linear
Gradient
      gradient.addColorStop(0, "#f0a3f4"); // Color stop
pada posisi 0 (light purple)
      gradient.addColorStop(0.25, "#ef7af5"); // Color stop
```



```

pada posisi 0.25 (light purple)
        gradient.addColorStop(0.5, "#ed56f5"); // Color stop
pada posisi 0.5 (dark purple)
        gradient.addColorStop(1, "#ec3df5"); // Color stop
pada posisi 1 (dark purple)

        context.rect(0, 0, canvas.width, 0.25 *
canvas.height); // Membuat sebuah object rectangle
        context.fillStyle = gradient; // Menerapkan style
gradient ke fill-style
        context.fill(); // Mengisi (fill) warna gradient pada
rectangle

    }
</script>
</head>
<body>

    <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Untuk *method* `addColorStop()` dapat dipanggil lebih dari satu kali dengan syarat *value* untuk *end color* masih berada pada rentang 0 hingga 1.

Hasil dari *code*:



**Gambar 5.15 Membuat *Linear Gradient***

### **Radial Gradient**

Untuk membuat *radial gradient* (*circular gradient*) dapat menggunakan *method* `createRadialGradient()`. Cara penggunaannya hampir sama dengan *method* `createLinearGradient()`, hanya berbeda di *parameter method*-nya saja.

Secara umum, bentuk *method* `createRadialGradient()` ialah seperti ini:

```
context.createRadialGradient(x0, y0, r0, x1, y1, r1);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>x0</b>	Koordinat x titik mulai <i>circle</i> (lingkaran) dalam <i>gradient</i>
<b>y0</b>	Koordinat y titik dimulainya <i>circle</i> (lingkaran) dalam <i>gradient</i>
<b>r0</b>	<i>Radius</i> (jari-jari) lingkaran awal
<b>x1</b>	Koordinat x titik akhir <i>circle</i> (lingkaran) dalam <i>gradient</i>
<b>y1</b>	Koordinat y titik akhir <i>circle</i> (lingkaran) dalam <i>gradient</i>
<b>r1</b>	<i>Radius</i> (jari-jari) lingkaran akhir

Contoh *code* membuat *radial gradient* di *canvas*:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenalan HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <!-- JavaScript Here -->
  <script type="text/javascript">
    window.onload = function() {
```

```

        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");

        var gradient = context.createRadialGradient(380, 100,
100, 420, 155, 250); // Membuat sebuah objek dari Radial Gradient
        gradient.addColorStop(0, "#f0a3f4");
        gradient.addColorStop(0.25, "#ef7af5");
        gradient.addColorStop(0.5, "#ed56f5");
        gradient.addColorStop(1, "#ec3df5");

        context.rect(0, 0, canvas.width, 0.25 *
canvas.height);
        context.fillStyle = gradient;
        context.fill();

    }
</script>
</head>
<body>

    <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>
</body>
</html>

```

Hasil dari *code*:



**Gambar 5.16** Membuat *radial gradient*

## 5.6 Adding Text, Images, and Shadow

### 5.6.1 Text

Untuk menambahkan sebuah teks ke dalam *canvas*, dapat menggunakan method `fillText()`. Warna *default* dari teks yang digambar adalah hitam.

Bentuk umum dari *method* `fillText()` adalah seperti ini:

```
context.fillText(text, x, y, maxWidth);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>text</b>	Teks yang akan ditulis di dalam <i>canvas</i>
<b>x</b>	Posisi x dimana teks akan mulai digambar (relatif terhadap <i>canvas</i> )
<b>y</b>	Posisi y dimana teks akan mulai digambar (relatif terhadap <i>canvas</i> )
<b>maxWidth</b>	Bersifat opsional. Besar teks maksimum yang diperbolehkan (dalam <i>pixel</i> )

Tambahan:

Untuk menspesifikasi jenis dan ukuran *font*, dapat menggunakan properti `font` dan gunakan properti `fillStyle` untuk memodifikasi *text* dengan warna lain atau *gradient*.

Contoh *code* membuat teks pada *canvas*:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>
  <script type="text/javascript">
```

```

        window.onload = function() {
            var canvas = document.getElementById("myCanvas");
            var context = canvas.getContext("2d");

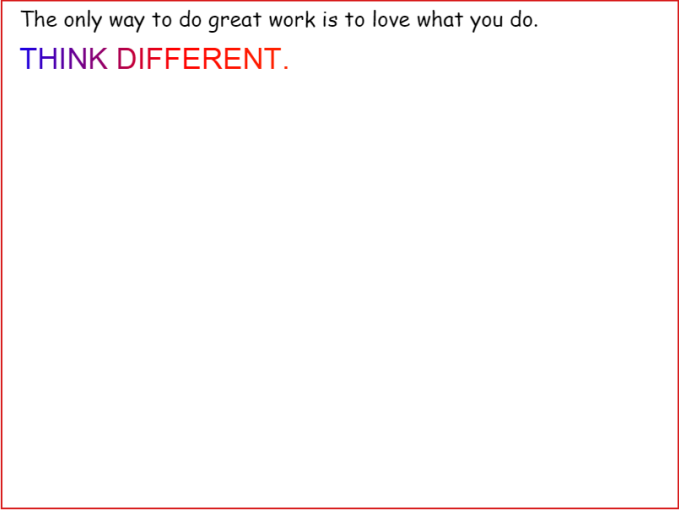
            context.font = "25px Comic Sans MS"; /* Customize
ukuran dan jenis font */
            context.fillText("The only way to do great work is to
love what you do.", 20, 30); /* Menggambar text di posisi (20,30) */

            context.font = "35px Helvetica";
            var gradient = context.createLinearGradient(0, 0,
canvas.width, 0); /* Memberikan efek linear gradient ke text */
            gradient.addColorStop(0, "blue");
            gradient.addColorStop(0.25, "red");
            gradient.addColorStop(1, "orange");
            context.fillStyle = gradient;
            context.fillText("THINK DIFFERENT.", 20, 80);
        }
    </script>
</head>
<body>

    <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>
</body>
</html>

```

Hasil dari *code* tersebut:



The only way to do great work is to love what you do.  
**THINK DIFFERENT.**

**Gambar 5.17 Menambahkan teks**

### 5.6.2 Image

Untuk menambahkan sebuah *image* ke dalam *canvas*, dapat menggunakan method `drawImage()`. Method ini juga dapat menggambar sebagian dari *image* dan mengatur *width* dan *height* dari sebagian *image* tersebut (biasanya digunakan untuk menggambar *sprite image* pada *game programming*).

Untuk mengatur posisi *image* pada *canvas*:

```
context.drawImage(image, x, y);
```

Untuk mengatur posisi *image* di *canvas* berserta menspesifikasi *width* dan *height* dari *image*:

```
context.drawImage(image, x, y, width, height);
```

Untuk mengambil sepotong *image* dari suatu *image* dan menggambarinya di *canvas*:

```
context.drawImage(image, sx, sy, swidth, sheight, x, y, width, height);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>image</b>	<i>Object</i> dari gambar yang akan digunakan.
<b>sx</b>	Koordinat x dari potongan <i>image</i> yang ingin diambil (opsional).
<b>sy</b>	Koordinat y dari potongan <i>image</i> yang ingin diambil (opsional).
<b>swidth</b>	Lebar dari potongan <i>image</i> yang ingin diambil (opsional).
<b>sheight</b>	Tinggi dari potongan <i>image</i> yang ingin diambil (opsional).
<b>x</b>	Posisi x dimana <i>image</i> akan digambar di <i>canvas</i> .
<b>y</b>	Posisi y dimana <i>image</i> akan digambar di <i>canvas</i> .
<b>width</b>	Lebar dari <i>image</i> yang akan digambar (opsional).
<b>height</b>	Tinggi dari <i>image</i> yang akan digambar (opsional).

Contoh *code* penggunaan *method* `drawImage()` di *canvas*:

```

<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      var image = document.getElementById("ironman"); // Me-
      load gambar asli dari tag <img> di HTML
      context.drawImage(image, 20, 20); // Menggambar gambar
      asli di posisi (20,20)

      context.drawImage(image, 20, 220, 150, 100); //
      Menggambar gambar asli di posisi (20,220) dengan ukuran 150 x 100

      context.drawImage(image, 0, 0, image.width / 3,
      image.height, 20, 360, image.width / 3, image.height); // Menggambar
      sprite pertama di posisi (20,360)
    }
  </script>
</head>
<body>

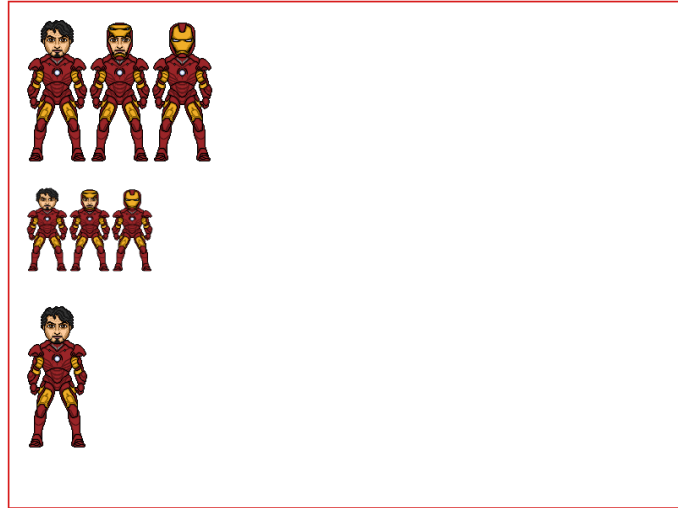
  <!-- Me-load aset gambar ironman_sprites.png dengan display awal
  hidden untuk digambar menggunakan Javascript -->
  

  <canvas id="myCanvas" width="800" height="600" style="background-
  color:#ffffff; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>

</body>
</html>

```

Hasil dari *code*:

Gambar 5.18 Menambahkan *image*

### 5.6.3 Shadow

Untuk menambahkan efek *shadow* ketika menggambar sebuah *object* di *canvas*, dapat menggunakan properti `shadowBlur` untuk *blur level* dan properti `shadowColor` untuk menentukan warna *shadow* yang digunakan.

Bentuk umum dari properti `shadowBlur` yaitu:

```
context.shadowBlur = number;
```

Dengan properti *value*:

Value	Deskripsi
<b>number</b>	<i>Blur level</i> untuk <i>shadow</i> (default: 0)

Bentuk umum dari properti `shadowColor` yaitu:

```
context.shadowColor = color;
```

Dengan properti *value*:

Value	Deskripsi
<b>color</b>	Kode warna CSS untuk warna <i>shadow</i> . Secara <i>default</i> warnanya adalah hitam (#000000).



Posisi *shadow* juga dapat diatur dengan properti `shadowOffsetX` dimana properti ini digunakan untuk mengatur posisi X (horizontal) dan properti `shadowOffsetY` digunakan untuk mengatur posisi Y (vertikal).

Berikut contoh penggunaan `shadowOffsetX`:

Code	Deskripsi
<code>shadowOffsetX = 30</code>	Posisi <i>shadow</i> dimulai 30 <i>pixels</i> ke kanan (dari posisi <i>left</i> shape).
<code>shadowOffsetX = -30</code>	Posisi <i>shadow</i> dimulai 30 <i>pixels</i> ke kiri (dari posisi <i>left</i> shape).

Berikut contoh penggunaan `shadowOffsetY`:

Code	Deskripsi
<code>shadowOffsetY = 30</code>	Posisi <i>shadow</i> dimulai 30 <i>pixels</i> di bawah posisi <i>top</i> shape.
<code>shadowOffsetY = -30</code>	Posisi <i>shadow</i> dimulai 30 <i>pixels</i> di atas posisi <i>top</i> shape.

Contoh *code* menambahkan efek *shadow* di *canvas*:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      var image = document.getElementById("ironman");
      context.shadowBlur = 10; // Mengatur blur-level dengan
      value 10
      context.shadowColor = "blue"; // Mengatur warna shadow
      dengan warna biru
    }
  </script>
</html>
```

```

        context.drawImage(image, 20, 20);

        context.shadowOffsetX = 15; // Mengatur posisi X
        shadow sebesar 15 ke kanan dari posisi left shape
        context.drawImage(image, 20, 220, 150, 100);

        context.shadowOffsetY = -15; // Mengatur posisi Y
        shadow sebesar 15 ke atas dari posisi top shape
        context.drawImage(image, 0, 0, image.width / 3,
        image.height, 20, 360, image.width / 3, image.height);
    }
</script>
</head>
<body>

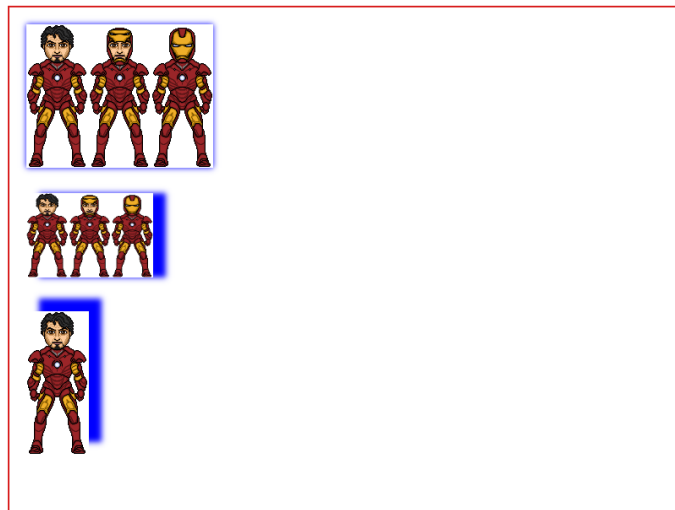
    

    <canvas id="myCanvas" width="800" height="600" style="background-
    color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Hasil dari *code* tersebut:



**Gambar 5.19 Menambahkan *shadow***

## 5.7 Transformations (Translating, Scaling, and Rotating)

### 5.7.1 Translating

Untuk melakukan translasi atau pergeseran terhadap suatu *object* pada *canvas*, dapat menggunakan method `translate()`.

Bentuk umum dari *method* `translate()` adalah seperti ini:

```
context.translate(x, y);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>x</b>	Value pergeseran untuk posisi X (horizontal).
<b>y</b>	Value pergeseran untuk posisi Y (vertikal).

Contoh *code* penggunaan method `translate()`:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenalan HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      var image = document.getElementById("ironman");
      context.drawImage(image, 20, 20);
      context.translate(200, 200); // Melakukan translasi
      // sejauh (20,20)
      context.drawImage(image, 20, 20);
    }
  </script>
</head>
<body>

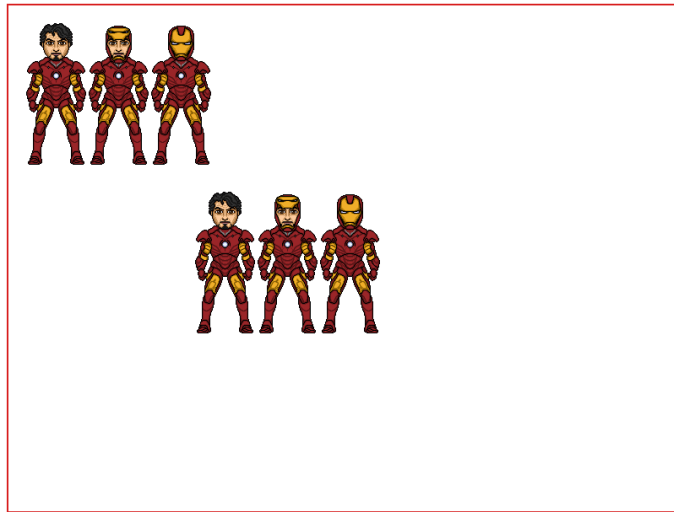
  

<canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
</canvas>

</body>
</html>

```

Hasil dari *code* tersebut:



**Gambar 5.10 Transformasi (translasi)**

### 5.7.2 Scaling

Untuk melakukan perubahan ukuran pada suatu *object* pada *canvas*, dapat menggunakan method `scale()`.

Bentuk umum dari *method* `scale()` adalah seperti ini:

```
context.scale(scalewidth, scaleheight);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>scalewidth</b>	Skala <i>width</i> yang ingin diterapkan pada <i>scaling</i> . Contoh: <ul style="list-style-type: none"> <li>Jika <i>scalewidth</i> = 0.5, maka lebar akan diperkecil 50% dari ukuran semula.</li> </ul>

	<ul style="list-style-type: none"> <li>• Jika <i>scalewidth</i> = 1.5, maka lebar akan diperbesar 150% dari ukuran semula.</li> </ul>
<b>scaleheight</b>	<p>Skala <i>height</i> yang ingin diterapkan pada <i>scaling</i>. Contoh:</p> <ul style="list-style-type: none"> <li>• Jika <i>scaleheight</i> = 0.5, maka tinggi akan diperkecil 50% dari ukuran semula.</li> <li>• Jika <i>scaleheight</i> = 1.5, maka tinggi akan diperbesar 150% dari ukuran semula.</li> </ul>

Contoh *code* penggunaan method *scale()*:

```

<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>
  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      var image = document.getElementById("ironman");
      context.drawImage(image, 20, 20);
      context.scale(2, 2); // Melakukan scaling terhadap
      gambar asli sebesar (2,2)
      context.drawImage(image, 80, 80);
    }
  </script>
</head>
<body>

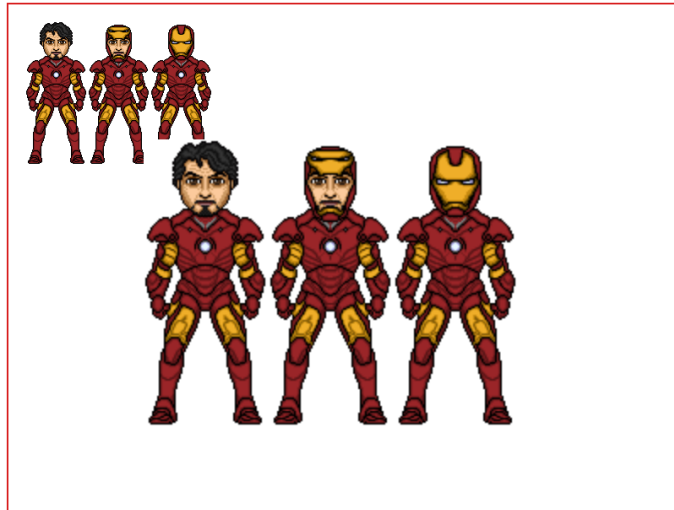
  

  <canvas id="myCanvas" width="800" height="600" style="background-
  color:#ffffff; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>

</body>
</html>

```

Hasil dari *code* tersebut:



Gambar 5.21 Transformasi (perbesaran)

### 5.7.3 Rotating

Untuk melakukan rotasi terhadap suatu *object* pada *canvas*, dapat menggunakan method `rotate()`.

Bentuk umum dari *method* `rotate()` adalah seperti ini:

```
context.rotate(angle);
```

Dengan parameter-parameter sebagai berikut.

Parameter	Deskripsi
<b>angle</b>	Sudut rotasi (dalam radian).

Untuk mengkonversi dari derajat ke radian menggunakan rumus:

```
radian = degree * Math.PI / 180
```

Contoh *code* penggunaan *method* `rotate()`:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
```

```

        background-color: white;
    }
</style>

<script type="text/javascript">
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");

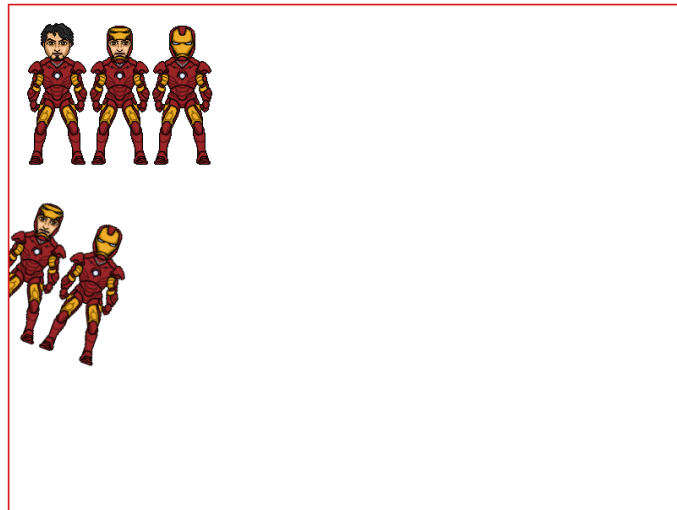
        var image = document.getElementById("ironman");
        context.drawImage(image, 20, 20);
        context.rotate(20 * Math.PI / 180); // Melakukan
        rotasi sebesar 20 derajat
        context.drawImage(image, 20, 200);
    }
</script>
</head>
<body>

    <canvas id="myCanvas" width="800" height="600" style="background-
    color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>
</body>
</html>

```

Hasil dari *code* tersebut:

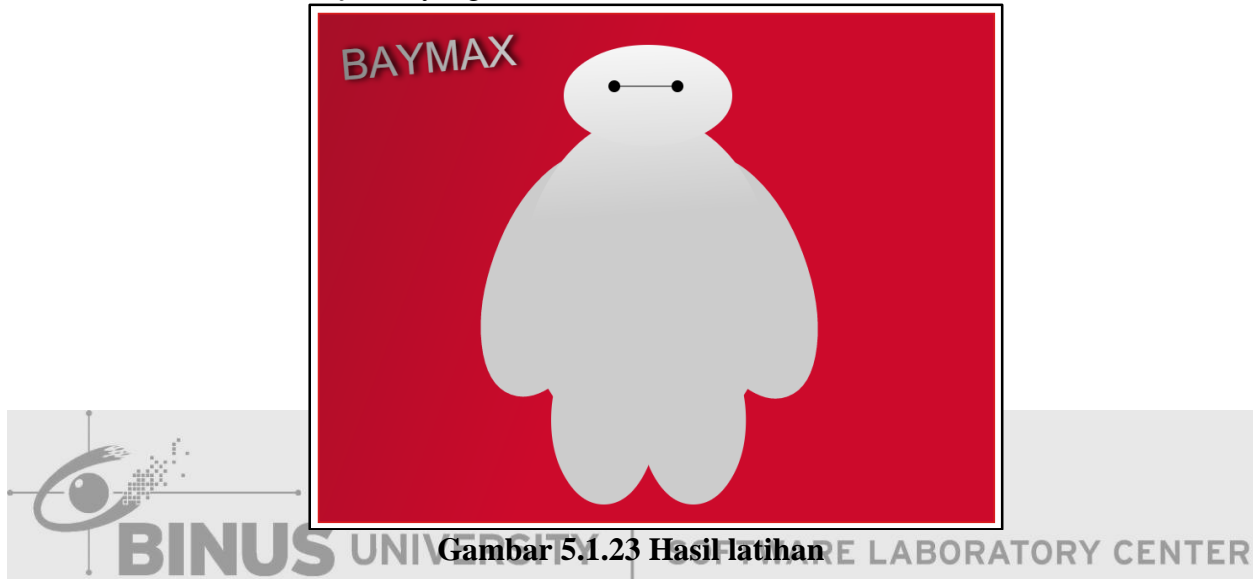


**Gambar 5.22 Transformasi (rotasi)**

## 5.8 Exercise

Gambarlah sebuah karakter **Baymax** dengan menggunakan semua *drawing tool canvas* yang telah dipelajari. Tambahkan efek *gradient* pada *background* dan **Baymax**. Tambahkan juga teks yaitu “**Baymax**” dengan ditambahkan efek *gradient*, *shadow*, dan rotasi pada teks.

Berikut hasil akhir **Baymax** yang dibuat:



Gambar 5.1.23 Hasil latihan

Berikut adalah *code* dari latihan diatas:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {

      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      // Background
      var gradient = context.createLinearGradient(0, 0,
canvas.width, 0.5 * canvas.height);
      gradient.addColorStop(0.5, "#cc0a2b");
```



```

        gradient.addColorStop(0, "#a60f28");

        context.rect(0, 0, canvas.width, canvas.height);
        context.fillStyle = gradient;
        context.fill();

        // Baymax Hand
        // First Hand
        context.save();
        context.rotate(20 * Math.PI / 180);
        var firsthandX = 245;
        var firsthandY = 65;
        var handRadius = 50;
        context.scale(1.5, 3);

        var gradientHand = context.createLinearGradient(245,
0, 270, 0.5 * canvas.height);
        gradientHand.addColorStop(0.5, "#cccccc");
        gradientHand.addColorStop(0, "#ffffff");

        context.beginPath();
        context.arc(firsthandX, firsthandY, handRadius, 0, 2 *
Math.PI, false);

        context.restore();

        context.fillStyle = gradientHand;
        context.fill();

        // Second Hand
        context.save();
        context.rotate(-20 * Math.PI / 180);
        var secondhandX = 245;
        var secondhandY = 155;
        context.scale(1.5, 3);

        context.beginPath();
        context.arc(secondhandX, secondhandY, handRadius, 0, 2
* Math.PI, false);

        context.restore();

        context.fillStyle = gradientHand;
        context.fill();

        // Baymax Foot
        // First Foot
        context.save();
        var firstfootX = 270;
        var firstfootY = 240;

```

```

        var footRadius = 50;
        context.scale(1.25, 2);

        var gradientFoot = context.createLinearGradient(245,
0, 270, 0.5 * canvas.height);
        gradientFoot.addColorStop(0.5, "#cccccc");
        gradientFoot.addColorStop(0, "#ffffff");

        context.beginPath();
        context.arc(firstfootX, firstfootY, footRadius, 0, 2 *
Math.PI, false);

        context.restore();

        context.fillStyle = gradientFoot;
        context.fill();

        // Second Foot
        context.save();
        var secondfootX = 355;
        var secondfootY = 240;
        context.scale(1.25, 2);

        context.beginPath();
        context.arc(secondfootX, secondfootY, footRadius, 0, 2
* Math.PI, false);
        context.restore();

        context.fillStyle = gradientFoot;
        context.fill();

        // Baymax Body
        context.save();
        var bodyX = 130;
        var bodyY = 80;
        var bodyRadius = 50;
        context.scale(3, 4);

        var gradientBody = context.createLinearGradient(130,
0, 150, 0.5 * canvas.height);
        gradientBody.addColorStop(0.85, "#cccccc");
        gradientBody.addColorStop(0, "#ffffff");

        context.beginPath();
        context.arc(bodyX, bodyY, bodyRadius, 0, 2 * Math.PI,
false);

        context.restore();

```

```

context.fillStyle = gradientBody;
context.fill();
// End of Body

// Baymax Head
context.save();
var headX = 195;
var headY = 80;
var headRadius = 50;
context.scale(2, 1.2);

var gradientHead = context.createLinearGradient(195,
0, 205, 0.5 * canvas.height);
gradientHead.addColorStop(0.85, "#cccccc");
gradientHead.addColorStop(0, "#ffffff");

context.beginPath();
context.arc(headX, headY, headRadius, 0, 2 * Math.PI,
false);

context.restore();

context.fillStyle = gradientHead;
context.fill();
// End of Head

// First Eye
context.save();
var firsteyeX = 350;
var firsteyeY = 85;
var eyeRadius = 7;

context.beginPath();
context.arc(firsteyeX, firsteyeY, eyeRadius, 0, 2 *
Math.PI, false);

context.restore();

context.fillStyle = '#000000';
context.fill();

context.lineWidth = 1;
context.beginPath();
context.moveTo(350, 85);
context.lineTo(425, 85);
context.stroke();

// Second Eye
context.save();

```

```

        var secondeyeX = 425;
        var secondeyeY = 85;

        context.beginPath();
        context.arc(secondeyeX, secondeyeY, eyeRadius, 0, 2 *
Math.PI, false);

        context.restore();

        context.fillStyle = '#000000';
        context.fill();

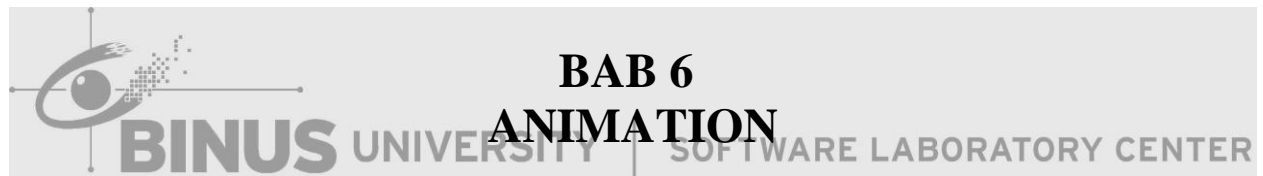
        // Title
        context.shadowBlur = 10;
        context.shadowColor = "black";
        context.shadowOffsetX = 5;
        context.font = "50px Helvetica";
        var gradient = context.createLinearGradient(0, 0,
canvas.width, 0);
        gradient.addColorStop(0, "gray");
        gradient.addColorStop(0.5, "white");
        context.fillStyle = gradient;
        context.rotate(-5 * Math.PI / 180);
        context.fillText("BAYMAX", 20, 80);
    }
</script>
</head>
<body>

    <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```







```

</style>

<script type="text/javascript">
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");

        /* State 1 */
        context.fillStyle = "#C20EBF";
        context.strokeStyle = "#1789C2";
        context.lineWidth = 3;
        context.fillRect(5, 5, 50, 50);
        context.strokeRect(5, 5, 50, 50);
        context.save(); // State 1 dipush ke dalam stack
        /* Akhir State 1 */

        /* State 2 */
        context.fillStyle = "#EB2640";
        context.fillRect(65, 5, 50, 50);
        context.strokeRect(65, 5, 50, 50);
        context.save(); // State 2 dipush ke dalam stack
        /* Akhir State 2 */

        /* State 3 */
        context.strokeStyle = "#31BA22";
        context.fillRect(125, 5, 50, 50);
        context.strokeRect(125, 5, 50, 50);
        /* Akhir State 3 */

        /* Back to State 2 */
        context.restore(); // State 2 dipop dari stack dan
diterapkan ke context
        context.fillRect(185, 5, 50, 50);
        context.strokeRect(185, 5, 50, 50);
        /* Akhir State 2 */

        /* Back to State 1 */
        context.restore(); // State 1 dipop dari stack dan
diterapkan ke context
        context.fillRect(245, 5, 50, 50);
        context.strokeRect(245, 5, 50, 50);
        /* Akhir State 1 */
    }
</script>
</head>
<body>

    <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.

```

```
</canvas>
</body>
</html>
```

Hasil dari *code* tersebut:



**Gambar 6.1** Tampilan hasil penggunaan *save()* dan *restore()*

## 6.2 Accessing Image Data

*Object* *ImageData* pada *canvas* bukanlah sebuah *image*, melainkan hanyalah sebuah area yang berbentuk *rectangle* pada *canvas*. *Object* ini menyimpan informasi setiap *pixel* yang ada di dalam *rectangle* tersebut.

Untuk mendapatkan *image data* (*pixel data*) dari sebuah *rectangle* pada *canvas*, dapat menggunakan *method* *getImageData()*. *Method* ini mengembalikan sebuah *object* dari *class* *ImageData* dimana *object* tersebut berisi duplikasi dari *pixel data* pada *rectangle* tersebut.

Bentuk umum dari *method* *getImageData()* adalah seperti ini:

```
context.getImageData(x, y, width, height);
```

Dengan parameter-parameter sebagai berikut.

Parameter	Deskripsi
X	Posisi x (dalam <i>pixel</i> ) sudut kiri atas area <i>rectangle</i> yang akan



	di-copy.
<b>Y</b>	Posisi y (dalam <i>pixel</i> ) sudut kiri atas area <i>rectangle</i> yang akan di-copy.
<b>Width</b>	Lebar dari area <i>rectangle</i> yang akan di-copy.
<b>Height</b>	Tinggi dari area <i>rectangle</i> yang akan di-copy.

Contoh code penggunaan *method* `getImageData()` di *canvas*:

```

<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      context.fillStyle = "#14BCF5";
      context.fillRect(10, 10, 150, 150); // Menggambar
      sebuah rectangle

      var imageData = context.getImageData(50, 50, 50, 50);
      // Meng-copy sebuah area berbentuk rectangle dari rectangle yang telah
      digambar

      context.putImageData(imageData, 180, 180); //
      Menggambar area rectangle yang telah di-copy di suatu titik
    }
  </script>
</head>
<body>

  <canvas id="myCanvas" width="800" height="600" style="background-
  color:#ffffff; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>

</body>
</html>

```

Hasil dari *code*:



Gambar 6.2 Hasil penggunaan *getImageData()*

### 6.3 Retrieving Pixel Values

Untuk setiap *pixel* di dalam sebuah *object* *ImageData* terdapat empat buah bagian informasi, yaitu *RGBA values*.

Values	Deskripsi
<b>R</b>	Red Color (0 – 255)
<b>G</b>	Green Color (0 – 255)
<b>B</b>	Blue Color (0 – 255)
<b>A</b>	Alpha Channel (0 – 255, 0 untuk transparan dan 255 untuk berwarna penuh)

Informasi *color/alpha* pada setiap *pixel image data* tersimpan di dalam suatu *array* pada properti yang bernama *data* milik *object* *ImageData*. Untuk mendapatkan *value* dari masing-masing *pixel* yang ada di dalam *object* tersebut dapat dilakukan dengan mengakses *array* itu sendiri.

Contoh *code* mendapatkan *pixel values* dari *image data* di *canvas*:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
```

```

        background-color: white;
    }
</style>

<script type="text/javascript">
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");

        context.fillStyle = "#14BCF5";
        context.fillRect(10, 10, 150, 150);

        var imageData = context.getImageData(50, 50, 50, 50);
        var red = imageData.data[0]; // Mendapatkan value red
        var green = imageData.data[1]; // Mendapatkan value
green
        var blue = imageData.data[2]; // Mendapatkan value
blue
        var alpha = imageData.data[3]; // Mendapatkan value
alpha
        alert("Red : " + red + ", " + "Green : " + green + ",
" + "Blue : " + blue + ", " + "Alpha : " + alpha); // Menampilkan semua
pixel data dari area tersebut
    }
</script>
</head>
<body>
    <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Hasil dari *code*:



**Gambar 6.3 Mendapatkan *image values***

## 6.4 Updating Pixel Values

Setelah mendapatkan *pixel value* dari area tersebut, *value* tersebut dapat dimanipulasi, misalnya dengan menambahkan *value*, mengurangi *value*, dan sebagainya. Setelah memanipulasinya, hasil manipulasi tersebut dapat didapatkan kembali untuk digambar kembali pada *canvas* dengan menggunakan *method* `putImageData()`.

Bentuk umum dari *method* `putImageData()` adalah seperti ini:

```
context.putImageData(imgData, x, y, dirtyX, dirtyY, dirtyWidth,
                    dirtyHeight);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>imgData</b>	<i>Object</i> <code>ImageData</code> yang ingin digambar kembali ke <i>canvas</i> .
<b>x</b>	Posisi x (dalam <i>pixel</i> ) sudut kiri atas dari <i>object</i> <code>ImageData</code> .
<b>y</b>	Posisi y (dalam <i>pixel</i> ) sudut kiri atas dari <i>object</i> <code>ImageData</code> .
<b>dirtyX</b>	Opsional. <i>Value</i> horizontal (x) (dalam <i>pixel</i> ) dimana <i>image</i> akan digambar pada <i>canvas</i> .
<b>dirtyY</b>	Opsional. <i>Value</i> vertikal (y) (dalam <i>pixel</i> ) dimana <i>image</i> akan digambar pada <i>canvas</i> .

<b>dirtyWidth</b>	Opsional. <i>Value</i> lebar yang akan digunakan untuk menggambar <i>image</i> pada <i>canvas</i> .
<b>dirtyHeight</b>	Opsional. <i>Value</i> tinggi yang akan digunakan untuk menggambar <i>image</i> pada <i>canvas</i> .

Untuk mencoba *code* selanjutnya, gunakan contoh gambar berikut:



Gambar 6.4 Gambar yang digunakan (sumber gambar: [http://cdn1-www.superherohype.com/assets/uploads/gallery/avengers-age-of-ultron-promo-art-2/10968381\\_725021770952233\\_6523847713573272717\\_n.jpg](http://cdn1-www.superherohype.com/assets/uploads/gallery/avengers-age-of-ultron-promo-art-2/10968381_725021770952233_6523847713573272717_n.jpg))

Contoh *code* penggunaan *method* `putImageData()` di *canvas*:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      var ultron = document.getElementById('ultron'); //
      Mengambil elemen html img yang ingin digambar

      context.drawImage(ultron, 0, 0);
      var imageData = context.getImageData(0, 0,
      canvas.width, canvas.height); // Mengcopy area rectangle yang ingin
      diambil image datanya

      // Manipulasi pixel (invers warna)
```

```

        for (var i = 0; i < imageData.data.length; i += 4) {
// Melakukan looping ke seluruh pixel dari image data
            imageData.data[i] = 255 - imageData.data[i]; //
Melakukan manipulasi berupa invers dari value red
            imageData.data[i + 1] = 255 - imageData.data[i +
1]; // Melakukan manipulasi berupa invers dari value green
            imageData.data[i + 2] = 255 - imageData.data[i +
2]; // Melakukan manipulasi berupa invers dari value blue
            imageData.data[i + 3] = 255; // Melakukan
manipulasi berupa invers dari value alpha
        }

        context.putImageData(imageData, 0, 0); // Menggambar
hasil manipulasi ke dalam canvas
    }
</script>
</head>
<body>

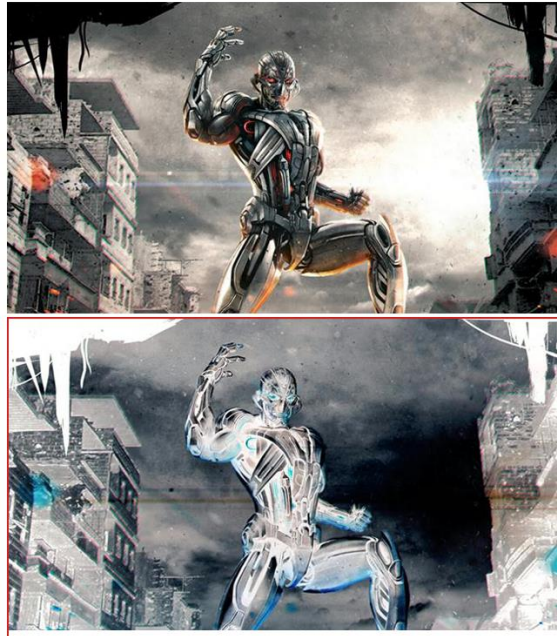
     <!--
Memasang gambar asli 'Ultron' sebelum dimanipulasi -->

    <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>
</body>
</html>

```

*Method* `getImageData()` tidak dapat menarik gambar langsung dari *server* lain. Ini disebabkan masalah *security* pada *browser* dimana *canvas* akan dikenakan status ‘*tainted*’ oleh *browser*. Untuk menangani masalah ini, gunakan *web server* untuk menjalankan dokumen HTML dan buka halaman tersebut melalui *server localhost*.

Hasil dari *code*:



Gambar 6.5 Hasil gambar

## 6.5 Exporting Image Data File

Untuk menyimpan hasil seluruh *drawing* pada *canvas* menjadi sebuah file *image*, dapat dilakukan dengan membuat sebuah object *image* dan men-set *source image* tersebut dengan *image data URL*. Dari sana, user dapat mengklik kanan pada *image* tersebut dan menyimpannya di komputer mereka. Untuk mendapatkan *image data URL* tersebut, dapat menggunakan *method* `toDataURL()`.

Contoh *code* penggunaan *method* `toDataURL()` di *canvas*:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    window.onload = function() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");

      var ultron = document.getElementById('ultron');
```

```

        context.drawImage(ultron, 0, 0);
        var imageData = context.getImageData(0, 0,
canvas.width, canvas.height);

        for (var i = 0; i < imageData.data.length; i += 4) {
            imageData.data[i] = 255 - imageData.data[i];
            imageData.data[i + 1] = 255 - imageData.data[i +
1];
            imageData.data[i + 2] = 255 - imageData.data[i +
2];
            imageData.data[i + 3] = 255;
        }

        context.putImageData(imageData, 0, 0);

        var dataURL = canvas.toDataURL(); // Menggenerate URL
dari hasil drawing pada canvas

        ultron.src = dataURL; // Mengeset src pada elemen img
html dengan URL yang baru saja digenerate oleh toDataURL()
    }
</script>
</head>
<body>
    

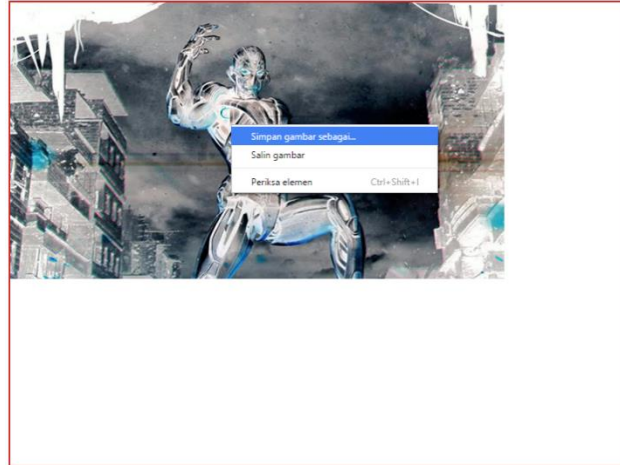
    <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

</body>
</html>

```

Hasil dari *code*:





Gambar 6.6 Hasil penggunaan *toDataURL()*

## 6.6 Animation Timing

### 6.6.1 `setTimeout()`

Dalam HTML5 *canvas animation*, terdapat *method* `setTimeout()` dari *object window* yang berfungsi untuk memanggil sebuah fungsi atau menjalankan sebuah *statement* setelah beberapa *milliseconds* (**1 second = 1000 ms**) yang telah dispesifikasikan di dalam parameter dari *method* ini.

Fungsi-fungsi yang dijalankan dalam `setTimeout()` hanya dijalankan sebanyak satu kali saja. Jika ingin dijalankan berulang-ulang dapat menggunakan *method* `setInterval()`. Untuk membatalkan jalannya *method* `setTimeout()`, dapat menggunakan *method* `clearTimeout()`.

Berikut adalah daftar *browser* minimum yang mendukung penggunaan *method* `setTimeout()`:

Browser	Versi Minimum
Google Chrome	1.0
Internet Explorer	4.0
Mozilla Firefox	2.0
Apple Safari	1.0
Opera	4.0

Bentuk umum dari *method* `setTimeout()` adalah seperti ini:

```
setTimeout(function, milliseconds, param1, param2, ...);
```

Dengan parameter-parameter sebagai berikut:

Parameter	Deskripsi
<b>function</b>	( <i>Required</i> ). Fungsi yang akan dijalankan oleh <code>setTimeout()</code> .
<b>milliseconds</b>	( <i>Required</i> ). <i>Delay</i> (dalam <i>milliseconds</i> ) sebelum fungsi dijalankan.
<b>param1, param2, ...</b>	( <i>Optional</i> ). Parameter-parameter tambahan yang ingin di- <i>passing</i> ke dalam fungsi. (Tidak di- <i>support</i> oleh IE9 dan versi di bawahnya)

Tambahan:

*Return value* dari *method* ini adalah sebuah *ID* dari *timer* yang telah di-*set*. *ID* ini akan menjadi *parameter* dari *method* `clearTimeout()` untuk membatalkan *timer*.

Contoh *code* penggunaan *method* `setTimeout()`:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Animasi pada HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    function showAlert() { // Mendeklarasikan fungsi showAlert()
      setTimeout(function() { // Menjadwalkan alert untuk
        dijalankan setelah 3 detik dengan setTimeout()
          alert('Hello, World!'); // Fungsi yang akan
        dijalankan oleh method setTimeout()
      }, 3000);
    }
  </script>
</head>
<body>
```

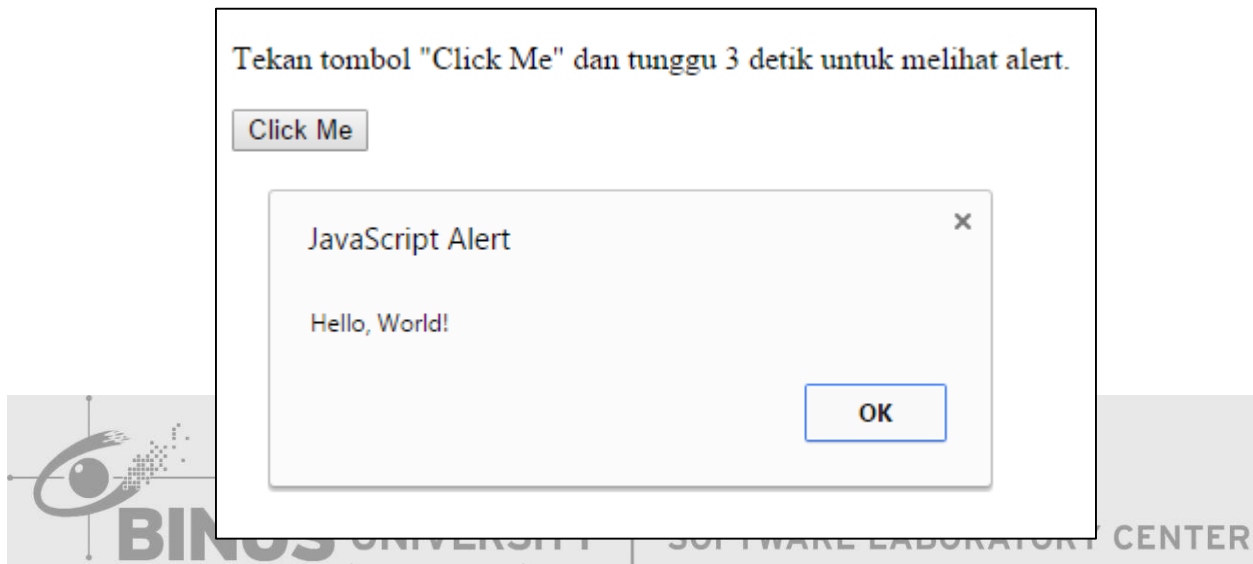
```

    <p>Tekan tombol "Click Me" dan tunggu 3 detik untuk melihat
    alert.</p>
    <button onclick="showAlert()">Click Me</button> <!-- Membuat
    sebuah button untuk diklik -->

</body>
</html>

```

Hasil dari *code* tersebut:



Gambar 6.7 Contoh penggunaan setTimeout()

Contoh *code* penggunaan method clearTimeout():

```

<!DOCTYPE html>
<html lang="id">
<head>
  <title>Animasi pada HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>

  <script type="text/javascript">
    var id;

    function showAlert() {
      id = setTimeout(function() { // Mendapatkan id dari
        timer untuk digunakan pada clearTimeout()
        alert('Hello, World!');
      }, 3000);
    }
  </script>

```

```

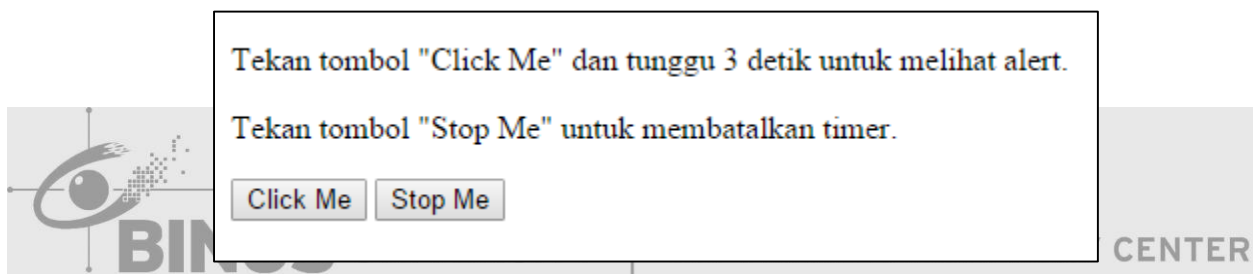
        function cancelAlert() {
            clearTimeout(id); // Membatalkan timer
        }
    </script>
</head>
<body>

    <p>Tekan tombol "Click Me" dan tunggu 3 detik untuk melihat
    alert.</p>
    <p>Tekan tombol "Stop Me" untuk membatalkan timer.</p>
    <button onclick="showAlert()">Click Me</button>
    <button onclick="cancelAlert()">Stop Me</button> <!-- Button untuk
    membatalkan scheduled alert -->

</body>
</html>

```

Hasil dari *code* tersebut:



Gambar 6.8 Contoh penggunaan `clearTimeout()`

### 6.6.2 `requestAnimationFrame()`

*Method* `window.requestAnimationFrame()` menyediakan cara yang lebih *smooth* dan efisien untuk membuat animasi pada halaman *web* dengan memanggil *frame-frame* animasi ketika sistem sudah siap untuk menggambar *frame-frame* tersebut. Sebelum API ini ada, animasi digambar dengan menggunakan *method* `setTimeout()` dan `setInterval()` yang tidak memberikan *web developer* cara yang efisien untuk menjadwalkan *timer* grafis untuk animasi yang dibuat. Hal ini mengakibatkan animasi digambar secara berlebihan, siklus CPU terbuang, dan menggunakan daya tambahan CPU untuk menjalankan animasi. Kemudian, seringkali animasi masih dijalankan bahkan ketika *web* itu sedang tidak terlihat oleh *user*, contohnya ketika halaman *web* sedang berada pada tab yang sedang tidak aktif atau ketika *browser* sedang di-*minimize*, yang mengakibatkan pemborosan daya.

Untuk menggunakan API ini, dapat memanggil `requestAnimationFrame()` dengan menggunakan sebuah *callback function*. Untuk masalah *animation timing*, sudah ditangani

oleh sistem secara otomatis. Seperti yang sudah dijelaskan di *sub-bab* sebelumnya, pada *method* `setTimeout()`, *animation timing* diatur seperti ini:

```
var handle = setTimeout(renderLoop, PERIOD, param1, param2, ...);
```

Note:

`renderLoop`: *Function* yang ingin dijalankan oleh `setTimeout()`.

`PERIOD`: *Delay* waktu yang diberikan sebelum *function* tersebut dijalankan.

Sekarang `setTimeout()` dapat diganti dengan menggunakan `requestAnimationFrame()` tanpa mendefinisikan *animation timing*, seperti berikut ini:

```
var handle = requestAnimationFrame(renderLoop);
```

Contoh di atas adalah penggambaran pertama kali animasi pada *canvas*. Untuk melanjutkan animasi, dapat dengan memanggil kembali `requestAnimationFrame()` dari dalam *callback function* (fungsi *render*) yang dibuat.

Contoh *code* penggunaan *method* `requestAnimationFrame()`:

```
<!DOCTYPE html>
<html>
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>
  <script type="text/javascript">
    var canvas, context; /* Inisialisasi variable canvas dan context */

    /* Menginisialisasi canvas dan context */
    function init() {
      canvas = document.getElementById("myCanvas");
      context = canvas.getContext("2d");
    }

    /* Merender animasi */
```

```

function render() {
    console.log('Animation Frame Loop'); // Mengetest animasi
    dengan mencetak di console
    window.requestAnimationFrame(render);
}

/* Method untuk memulai animasi */
function start() {
    window.requestAnimationFrame(render);
}

// Menghandle semua jenis browser untuk penggunaan API
requestAnimationFrame
/*
    ** Webkit (Chrome, Safari)
    ** Moz (Mozilla Firefox)
    ** O (Opera)
*/
window.requestAnimationFrame = (function () {
    return window.requestAnimationFrame ||
        window.webkitRequestAnimationFrame ||
        window.mozRequestAnimationFrame ||
        window.oRequestAnimationFrame ||
        // Jika semuanya fail, gunakan setTimeout()
        function (callback) {
            return window.setTimeout(callback, 1000 / 60);
        };
})();

// Menghandle semua jenis browser untuk penggunaan API
cancelAnimationFrame
/*
    ** Webkit (Chrome, Safari)
    ** Moz (Mozilla Firefox)
    ** O (Opera)
*/
window.cancelAnimationFrame = (function () {
    return window.cancelAnimationFrame ||
        window.webkitCancelAnimationFrame ||
        window.mozCancelAnimationFrame ||
        window.oCancelAnimationFrame ||
        // Jika semuanya fail, gunakan clearTimeout()
        function (id) {
            window.clearTimeout(id);
        };
})();
</script>
</head>
<body onload="init()"> <!-- Memanggil fungsi init() ketika document

```

```

sudah di-load -->

<canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
  Browser anda belum mendukung HTML5 canvas.
</canvas>

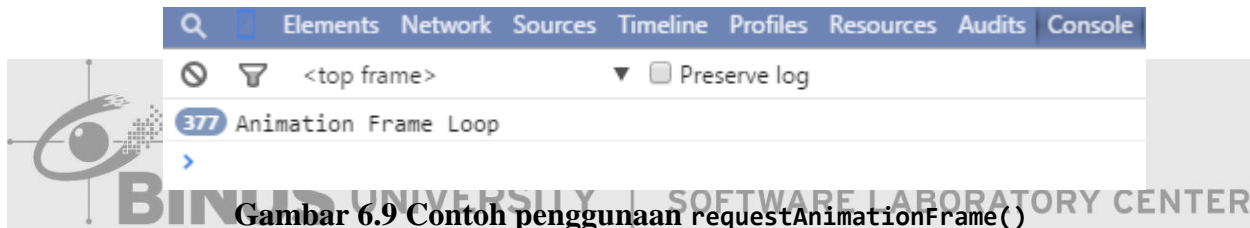
<script type="text/javascript">
  start(); // Memulai animasi
</script>

</body>
</html>

```

Untuk melihat *output*, klik kanan pada halaman *web*, lalu klik *Inspect Element* dan buka pada *tab Console*. Akan terlihat *log* “Animation Frame Loop” yang terus menerus di-loop oleh *animation frame*.

Hasil dari *code* tersebut:



Gambar 6.9 Contoh penggunaan `requestAnimationFrame()`

### 6.6.3 `cancelAnimationFrame()`

*Method* `window.cancelAnimationFrame()` digunakan untuk menghentikan *animation frame* yang sebelumnya telah dipanggil oleh *method* `window.requestAnimationFrame()`. *Method* ini meminta sebuah parameter yaitu `requestID` yang di-return oleh *method* `window.requestAnimationFrame()`.

## 6.7 Rendering and Applying Animation

Untuk merender sebuah animasi, dua fungsi utama yang diperlukan adalah fungsi *update* dan fungsi *render*. Fungsi *update* berisi segala *statement* perubahan *state object-object* dalam *canvas* seperti perubahan posisi atau perubahan jumlah *object*. Fungsi *render* berisi fungsi untuk me-reset ulang *canvas* kosong dan menggambar ulang isi *canvas* dengan *state* yang baru. Apabila dua fungsi utama tersebut di-loop dengan `requestAnimationFrame()`, maka akan terbentuk sebuah animasi.

Contoh code rendering and applying animation:

```

<!DOCTYPE html>
<html>
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white; /* Memberikan warna background
dasar ke halaman web */
    }
  </style>
  <script type="text/javascript">
    var canvas, context;
    var requestId = 0; // Init variable request id
    var x = 0, y = 0; // Init posisi awal object rectangle

    function init() {
      canvas = document.getElementById("myCanvas");
      context = canvas.getContext("2d");
    }

    /* Method untuk clear/reset canvas */
    function clear() {
      context.clearRect(0, 0, canvas.width, canvas.height);
    }

    /* Update posisi x dan y object */
    function update() {
      x++;
      y++;
    }

    function render() {
      clear(); // Clear canvas sebelum re-paint

      /* Statement untuk re-paint */
      context.beginPath();
      context.rect(x, y, 200, 100);
      context.fillStyle = 'yellow';
      context.fill();
      context.lineWidth = 7;
      context.strokeStyle = 'black';
      context.stroke();
      /* ===== */

      update(); // Memanggil fungsi update untuk dijalankan setiap
kali render
  </script>

```



```

        requestId = window.requestAnimationFrame(render); // Memanggil
        fungsi render sebagai callback dari requestAnimationFrame
    }

    function start() {
        requestId = window.requestAnimationFrame(render); // Menampung
        requestId untuk digunakan di cancelAnimationFrame
    }

    /* Method untuk menghentikan animasi dengan cancelAnimationFrame
    */

    function stop() {
        if (requestId) {
            window.cancelAnimationFrame(requestId);
        }
    }

    window.requestAnimationFrame = (function () {
        return window.requestAnimationFrame ||
            window.webkitRequestAnimationFrame ||
            window.mozRequestAnimationFrame ||
            window.oRequestAnimationFrame ||
            // Jika semuanya fail, gunakan setTimeout()
            function (callback) {
                return window.setTimeout(callback, 1000 / 60);
            };
    })();

    window.cancelAnimationFrame = (function () {
        return window.cancelAnimationFrame ||
            window.webkitCancelAnimationFrame ||
            window.mozCancelAnimationFrame ||
            window.oCancelAnimationFrame ||
            function (id) {
                window.clearTimeout(id);
            };
    })();
</script>
</head>
<body onload="init()">

    <canvas id="myCanvas" width="800" height="600" style="background-
    color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

    <button onclick="start()">Start</button> <!-- Button untuk memulai
    animasi -->
    <button onclick="stop()">Stop</button> <!-- Button untuk

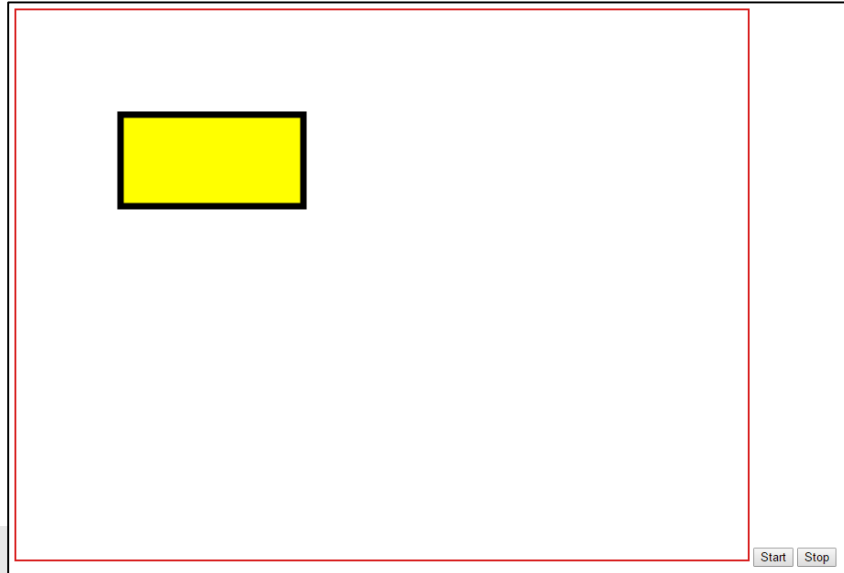
```

```

menghentikan animasi -->
</body>
</html>

```

Hasil dari *code* tersebut:



**Gambar 6.10 Contoh animasi pada *canvas***

Ketika tombol “*Start*” ditekan, *rectangle* akan bergerak ke bawah dan ketika *button* “*Stop*” ditekan, animasi akan berhenti (*paused*). Animasi yang terbentuk lebih *smooth* dibanding dengan menggunakan *method* *setTimeout()*.

## 6.8 Moving Object, Removing Object, Add Object

### 6.8.1 Moving Object

Untuk memindahkan posisi sebuah *object* dalam animasi, dapat dilakukan dengan mengubah posisi *x* dan *y* dari *object* tersebut pada fungsi *update()*. Misalkan untuk menggeser objek ke kanan, dapat mengubah posisi *x* dengan menambahkannya sebesar jarak perpindahannya dari posisi semula dan panggil fungsi *render()* lagi sehingga terbentuk sebuah animasi pergeseran.

Contoh *code moving object*:

```

<!DOCTYPE html>
<html>
<head>
  <title>Pengenal HTML5 Canvas</title>

```

```

<style>
  body {
    background-color: white;
  }
</style>
<script type="text/javascript">
  var canvas, context;
  var requestId = 0;

  var x = 45, y = 45; // Mengatur posisi x dan y awal bola
  var vx = 5, vy = 5; // Mengatur kecepatan x dan y bola
  var radius = 35; // Radius atau jari-jari bola yang akan
  digambar

  function init() {
    canvas = document.getElementById("myCanvas");
    context = canvas.getContext("2d");
  }

  function clear() {
    context.clearRect(0, 0, canvas.width, canvas.height);
  }

  // Memindahkan posisi x dan y bola sehingga bola bergerak dengan
  // memantul ke dinding
  function update() {
    if (x <= 0 + radius || x >= canvas.width - radius) {
      vx *= -1;
    }
    if (y <= 0 + radius || y >= canvas.height - radius) {
      vy *= -1;
    }
    x += vx;
    y += vy;
  }

  /* Render Animasi */
  function render() {
    if (context != null) {
      clear();

      /* Statement untuk re-paint */
      context.beginPath();
      context.arc(x, y, radius, 0, 2 * Math.PI, false);
      context.fillStyle = 'blue';
      context.fill();
      /* ===== */

      update(); /* Memanggil fungsi update() setiap kali
  render */
    }
  }
</script>

```

```

    }

    window.requestAnimationFrame(render);
  }

  function start() {
    window.requestAnimationFrame(render);
  }

  window.requestAnimationFrame = (function () {
    return window.requestAnimationFrame ||
      window.webkitRequestAnimationFrame ||
      window.mozRequestAnimationFrame ||
      window.oRequestAnimationFrame ||
      function (callback) {
        return window.setTimeout(callback, 1000 / 60);
      };
  })();

  window.cancelAnimationFrame = (function () {
    return window.cancelAnimationFrame ||
      window.webkitCancelAnimationFrame ||
      window.mozCancelAnimationFrame ||
      window.oCancelAnimationFrame ||
      function (id) {
        window.clearTimeout(id);
      };
  })();
</script>
</head>
<body onload="init()">

  <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
    Browser anda belum mendukung HTML5 canvas.
  </canvas>

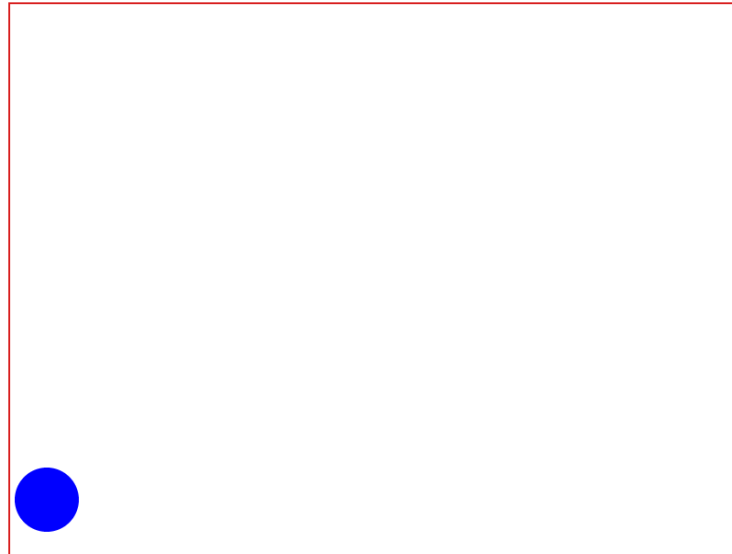
  <script type="text/javascript">
    start();
  </script>

</body>
</html>

```

Hasil dari *code* tersebut:

Bola akan bergerak dengan memantul setiap mengenai dinding.

Gambar 6.11 Contoh *moving object*

### 6.8.2 Adding Object

Untuk menambahkan sebuah *object* dalam animasi, dapat dilakukan dengan menggambar lebih dari satu *object* dengan posisi yang berbeda pada fungsi render. Untuk mempermudah penambahan atau penghapusan *object*, dapat menggunakan array pada *Javascript*. Untuk menambahkan *object* baru, dapat menggunakan *method* `array.push()`.

Contoh code adding *object*:

```
<!DOCTYPE html>
<html>
<head>
  <title>Pengenaln HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>
  <script type="text/javascript">
    var canvas, context;

    var x = [], y = []; // Init array posisi x dan y untuk array
bola
    var vx = [], vy = []; // Init array kecepatan x dan y untuk
array bola
    var radius = 35; // Jari-jari atau radius bola sebesar 35

    function init() {
      canvas = document.getElementById("myCanvas");
```

```

        context = canvas.getContext("2d");

        /* Menambahkan sebuah bola baru ke dalam array di awal */
        x.push(45);
        y.push(45);
        vx.push(5);
        vy.push(5);
    }

    function clear() {
        context.clearRect(0, 0, canvas.width, canvas.height);
    }

    function update() {
        for (var i = 0; i < x.length; i++) { // Loop untuk semua bola di dalam array
            if (x[i] <= 0 + radius || x[i] >= canvas.width - radius)
            {
                vx[i] *= -1;
            }
            if (y[i] <= 0 + radius || y[i] >= canvas.height - radius) {
                vy[i] *= -1;
            }
            x[i] += vx[i];
            y[i] += vy[i];
        }
    }

    /* Render Animasi */
    function render() {
        if (context != null) {
            clear();

            for (var i = 0; i < x.length; i++) { // Loop untuk semua bola di dalam array
                /* Statement untuk re-paint */
                context.beginPath();
                context.arc(x[i], y[i], radius, 0, 2 * Math.PI,
false);

                context.fillStyle = 'blue';
                context.fill();
                /* ===== */
            }

            update();
        }

        window.requestAnimationFrame(render);
    }

```

```

function start() {
    window.requestAnimationFrame(render);
}

/* Fungsi untuk menambahkan bola baru */
function addBall() {
    x.push(45); /* Push posisi x untuk bola baru */
    y.push(45); /* Push posisi y untuk bola baru */
    vx.push(5); /* Push kecepatan x untuk bola baru */
    vy.push(5); /* Push kecepatan y untuk bola baru */
}

window.requestAnimationFrame = (function () {
    return window.requestAnimationFrame ||
        window.webkitRequestAnimationFrame ||
        window.mozRequestAnimationFrame ||
        window.oRequestAnimationFrame ||
        function (callback) {
            return window.setTimeout(callback, 1000 / 60);
        };
})();

window.cancelAnimationFrame = (function () {
    return window.cancelAnimationFrame ||
        window.webkitCancelAnimationFrame ||
        window.mozCancelAnimationFrame ||
        window.oCancelAnimationFrame ||
        function (id) {
            window.clearTimeout(id);
        };
})();
</script>
</head>
<body onload="init()">

    <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

    <button onclick="addBall()">Add Ball</button> <!-- Button untuk
menambahkan bola baru -->

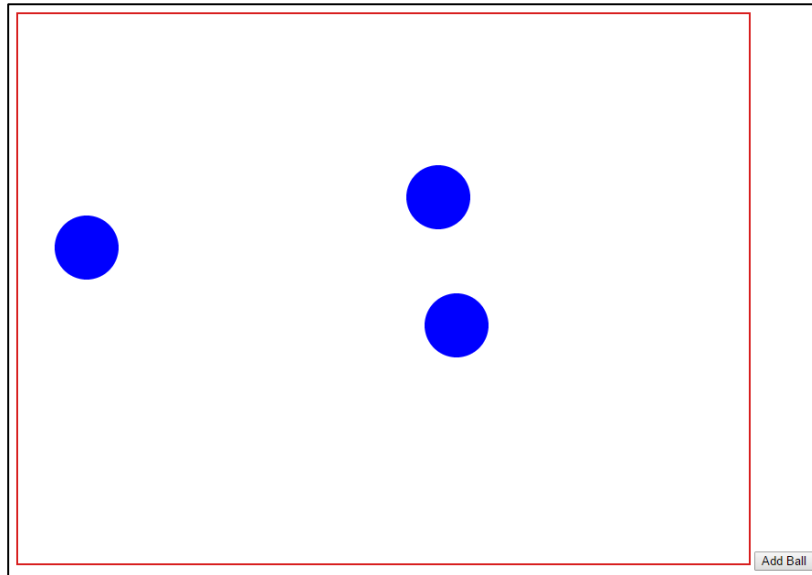
    <script type="text/javascript">
        start();
    </script>

</body>
</html>

```

Hasil dari *code* tersebut:

Ketika *button* “Add Ball” ditekan sekali, bola akan bertambah satu.



**Gambar 6.12** Contoh *adding object*

### 6.8.3 Removing Object

Untuk melakukan penghapusan objek yang sudah ada pada *canvas*, konsep yang digunakan sama seperti konsep menambahkan objek baru, dapat menggunakan array. Hanya saja untuk menghapus *object* di dalam array gunakan *method* `array.pop()`.

Contoh *code removing object*:

```
<!DOCTYPE html>
<html>
<head>
  <title>Pengenal HTML5 Canvas</title>
  <style>
    body {
      background-color: white;
    }
  </style>
  <script type="text/javascript">
    var canvas, context;

    var x = [], y = [];
    var vx = [], vy = [];
    var radius = 35;

    function init() {
      canvas = document.getElementById("myCanvas");
```



```

        context = canvas.getContext("2d");

        x.push(45);
        y.push(45);
        vx.push(5);
        vy.push(5);
    }

    function clear() {
        context.clearRect(0, 0, canvas.width, canvas.height);
    }

    function update() {
        for (var i = 0; i < x.length; i++) {
            if (x[i] <= 0 + radius || x[i] >= canvas.width - radius)
            {
                vx[i] *= -1;
            }
            if (y[i] <= 0 + radius || y[i] >= canvas.height -
radius) {
                vy[i] *= -1;
            }
            x[i] += vx[i];
            y[i] += vy[i];
        }
    }

    function render() {
        if (context != null) {
            clear();

            for (var i = 0; i < x.length; i++) {
                /* Statement untuk re-paint */
                context.beginPath();
                context.arc(x[i], y[i], radius, 0, 2 * Math.PI,
false);

                context.fillStyle = 'blue';
                context.fill();
                /* ===== */
            }

            update();
        }

        window.requestAnimationFrame(render);
    }

    function start() {
        window.requestAnimationFrame(render);
    }

```

```

function addBall() {
    x.push(45);
    y.push(45);
    vx.push(5);
    vy.push(5);
}

/* Fungsi untuk menghapus sebuah bola dengan menggunakan method
pop() */
function removeBall() {
    x.pop();
    y.pop();
    vx.pop();
    vy.pop();
}

window.requestAnimationFrame = (function () {
    return window.requestAnimationFrame ||
        window.webkitRequestAnimationFrame ||
        window.mozRequestAnimationFrame ||
        window.oRequestAnimationFrame ||
        function (callback) {
            return window.setTimeout(callback, 1000 / 60);
        };
})();

window.cancelAnimationFrame = (function () {
    return window.cancelAnimationFrame ||
        window.webkitCancelAnimationFrame ||
        window.mozCancelAnimationFrame ||
        window.oCancelAnimationFrame ||
        function (id) {
            window.clearTimeout(id);
        };
})();
</script>
</head>
<body onload="init()">

    <canvas id="myCanvas" width="800" height="600" style="background-
color:#ffffff; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

    <button onclick="addBall()">Add Ball</button>
    <button onclick="removeBall()">Remove a Ball</button> <!-- Button
untuk menghapus sebuah bola -->

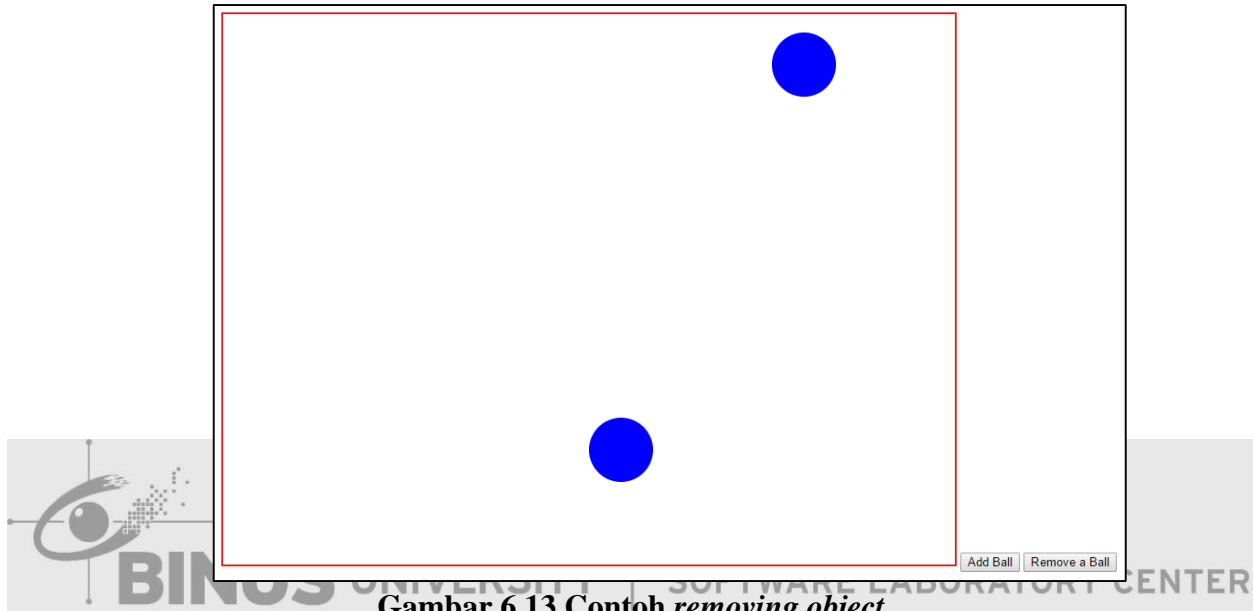
    <script type="text/javascript">

```

```
start();  
</script>  
  
</body>  
</html>
```

Hasil dari *code* tersebut:

Ketika *button* “*Remove Ball*” ditekan, bola terakhir yang ditambahkan akan hilang.

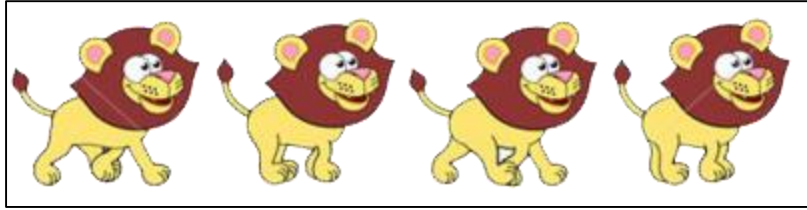


Gambar 6.13 Contoh *removing object*

## 6.9 Exercise

Buatlah sebuah animasi pada *canvas* dengan menggunakan *API* *requestAnimationFrame* yang telah dipelajari sebelumnya. Animasi terdiri dari sebuah *sprite animation* dari karakter **lion** yang berjalan ke arah kanan menuju **mushroom**. **Lion** bertambah satu ekor setiap *interval* 5 detik. Tambahkan *collision detection* dimana setiap kali **Lion** menabrak **mushroom**, **Lion** tersebut akan hilang dari *scene*.

*Sprite images* yang digunakan:

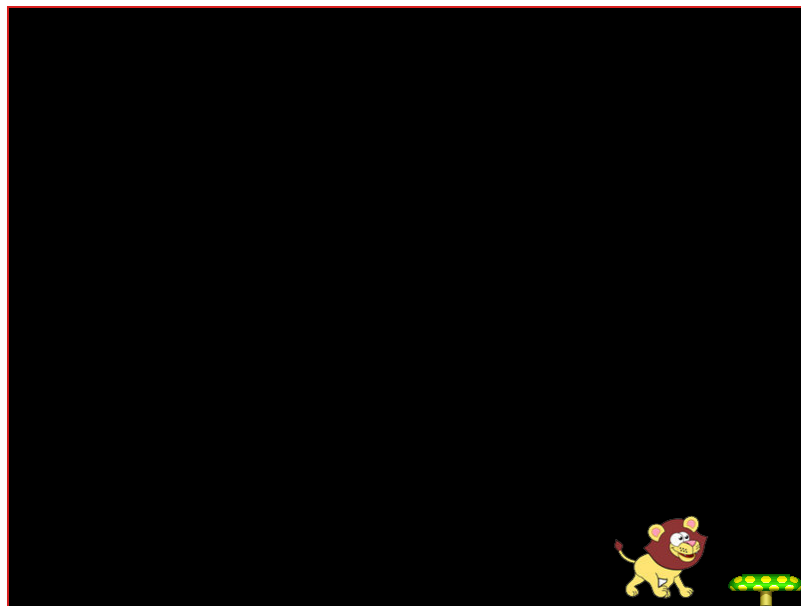


Gambar 6.14 *Sprite images* untuk Lion (sumber gambar: [https://0.s3.envato.com/files/70017753/01\\_Animal\\_Walking4.jpg](https://0.s3.envato.com/files/70017753/01_Animal_Walking4.jpg))



Gambar 6.15 *Mushroom* (sumber gambar: <http://www.smackjееves.com/images/uploaded/comics/5/5/55e8b3232f1jw.png>)

Berikut hasil akhir animasi yang dibuat:



Gambar 6.16 Hasil akhir latihan

Gunakan gambar yang sudah di-*crop*:

**Lion** : <http://postimg.org/image/d1014evvv/>

**Mushroom** : <http://postimg.org/image/75uo91k1t/>

Berikut adalah *code* dari latihan di atas (simpan gambar **lion.png** dan **mushroom.png** di atas yang sudah di-*crop* ke dalam folder **images**):

```
<!DOCTYPE html>
<html lang="id">
<head>
  <title>Latihan</title>
  <style>
```

```

        body {
            background-color: white;
        }
    </style>
</head>
<body>

    <canvas id="canvas" width="800" height="600" style="background-
color:#000; border:2px solid #d81e1e;">
        Browser anda belum mendukung HTML5 canvas.
    </canvas>

    <script type="text/javascript">
        window.onload = function() {

            /*
             * Merequest Animation Frame API
             * Untuk semua jenis browsers
            */

            window.requestAnimFrame = (function() {
                return window.requestAnimationFrame ||
                    window.webkitRequestAnimationFrame ||
                    window.mozRequestAnimationFrame ||
                    window.oRequestAnimationFrame ||
                    window.msRequestAnimationFrame ||
                    function (callback) {
                        window.setTimeout(callback,

1000 / 60);
                    };
            })();

            var canvas = document.getElementById("canvas");
            var context = canvas.getContext("2d");
            var spriteImg = null; // Membuat variabel untuk sprite
            // Lion
            var mushroomImg = null; // Membuat variabel untuk
            // gambar mushroom
            var frame = 0; // Mendeklarasikan variable untuk
            // menampung posisi current frame ketika update

            /* Time Based */
            var lastUpdate = 0; // Variabel untuk menampung waktu
            // terakhir fungsi update dijalankan
            var deltaTime = 0; // Variabel untuk menampung selisih
            // last update dengan waktu sekarang
            var animInterval = 100; // Interval untuk saatnya
            // pindah ke frame sprite berikutnya

```

```

var xLion = []; // Array untuk menampung posisi x dari
array of lions

/* Fungsi untuk me-reset canvas */
var clear = function() {

    context.clearRect(0, 0, canvas.width,
canvas.height);

}

/* Fungsi untuk me-load semua gambar sprite yang
digunakan */
var loadSprite = function() {

    spriteImg = new Image();
    spriteImg.src = 'images/sprite.png';

    mushroomImg = new Image();
    mushroomImg.src = 'images/mushroom.png';

}

/* Fungsi untuk merender animasi */
var render = function() {

    clear(); // Clear canvas

    // Menggambar Lion
    for (var i = 0; i < xLion.length; i++) {
        /* Menggambar sprite lion sesuai frame
        sekarang (parameter 2) */
        context.drawImage(spriteImg, frame * 100,
0, 100, 102, xLion[i], canvas.height - 102, 100, 102);
    }

    // Menggambar Mushroom di posisi paling kanan
    canvas
    context.drawImage(mushroomImg, canvas.width -
mushroomImg.width, canvas.height - mushroomImg.height,
mushroomImg.width, mushroomImg.height);

}

/* Fungsi untuk mengupdate state */
var update = function() {

    requestAnimationFrame(update);

    /* Mengatur kapan saatnya pindah ke frame

```

```

        berikutnya */
        var tempDelta = Date.now() - lastUpdate;
        if (deltaTime > animInterval) {
            deltaTime = 0;
            render();
            frame++; /* Pindah ke frame berikutnya
apabila delay sudah selesai */
            if (frame > 3) { /* Apabila frame sudah
lebih dari 3, balik ke state 1 lion */
                frame = 0;
            }
        } else {
            deltaTime += tempDelta;
        }

        lastUpdate = Date.now();

        /* Menggerakkan Lion ke arah kanan */
        for (var i = 0; i < xLion.length; i++) {
            xLion[i] += 5;
        }

        /* Collision Detection */
        if (xLion[0] + 100 >= (canvas.width -
mushroomImg.width)) {
            xLion.shift();
        }

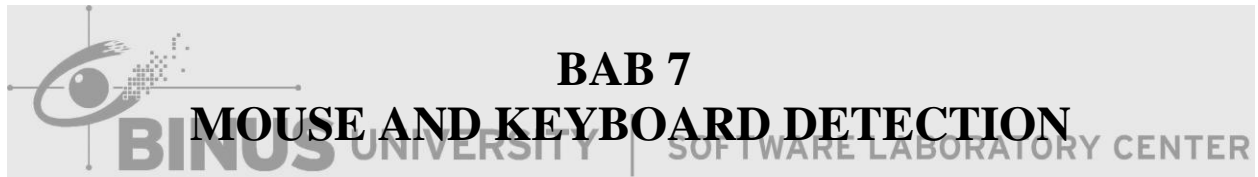
        /* Menambahkan lion baru setiap 5 detik */
        setInterval(function() {
            xLion.push(-20);
        }, 5000);

        loadSprite(); // Memanggil fungsi load sprite sebelum
animasi dijalankan
        update(); // Menjalankan animasi
    }
</script>

</body>

</html>

```





## 7.1 Mouse Events

**Mouse event** adalah *event* yang akan ter-*trigger* saat suatu aksi dilakukan pada tombol *mouse* pada suatu elemen. *Mouse event* terdiri dari dua jenis, yaitu:

### a. Simple Event

**Simple event** adalah *event* yang terdiri atas satu aksi saja dan hanya memakai tombol kiri *mouse*. *Event* yang termasuk *simple event* adalah:

Event	Penjelasan
<b>mousedown</b>	<i>Event</i> yang terjadi ketika tombol <i>mouse</i> ditekan pada suatu elemen.
<b>mouseup</b>	<i>Event</i> yang terjadi ketika tombol <i>mouse</i> terlepas atau diangkat dari suatu elemen.
<b>mouseover</b>	<i>Event</i> yang terjadi ketika penunjuk <i>mouse</i> masuk ke dalam bagian dari suatu elemen.
<b>mouseout</b>	<i>Event</i> yang terjadi ketika penunjuk <i>mouse</i> keluar dari bagian dari suatu elemen.
<b>mousemove</b>	<i>Event</i> yang terjadi ketika penunjuk <i>mouse</i> bergerak di dalam bagian dari suatu elemen.

### Contoh *mousedown* dan *mouseup*:

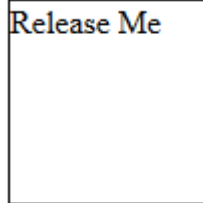
```

...
<body>
  <div onmousedown="mDown(this)" onmouseup="mUp(this)"
  style="border: 1px solid; width:100px; height:100px"> Press Me </div>
  <script>
    function mDown(obj) {
      obj.innerHTML = "Release Me"; //teks berubah ketika mouse
      di tekan
    }

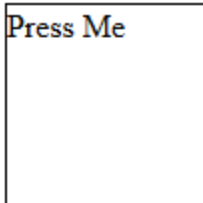
    function mUp(obj) {
      obj.innerHTML="Press Me"; //teks berubah ketika mouse di
      lepas
    }
  </script>
</body>
...

```

Hasil dari *code*:



Gambar 7.1 Tampilan ketika tombol *mouse* ditekan



Gambar 7.2 Tampilan ketika tombol *mouse* dilepas

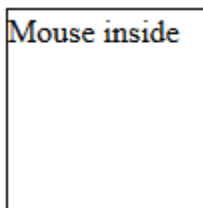
Contoh *mouseover* dan *mouseout*:

```
...
<body>
  <div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="border: 1px solid; width:100px; height:100px;"> Mouse outside
</div>

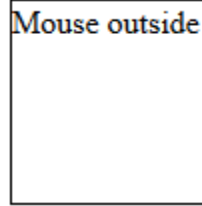
  <script>
    function mOver(obj) {
      obj.innerHTML = "Mouse inside"// text berubah ketika
      pointer mouse berada di dalam div
    }

    function mOut(obj) {
      obj.innerHTML = "Mouse outside"// text berubah ketika
      pointer mouse berada di luar div
    }
  </script>
</body>
...
```

Hasil dari *code*:



Gambar 7.3 Tampilan ketika *pointer mouse* berada di dalam div



Gambar 7.4 Tampilan ketika *pointer mouse* berada diluar div

## b. Complex Event

**Complex event** adalah *event* yang terdiri dari gabungan dua aksi dan juga memakai tombol *mouse* selain tombol kiri. *Event* yang termasuk *complex event* adalah:

Event	Penjelasan
click	<i>Event</i> yang terjadi ketika sebuah tombol <i>mouse</i> ditekan dan dilepas dari sebuah elemen (gabungan dari <i>mousedown</i> dan <i>mouseup</i> ).
dblclick	<i>Event</i> yang terjadi ketika terdapat dua buah <i>click</i> pada sebuah elemen dalam suatu jangka waktu yang pendek.
contextmenu	<i>Event</i> yang terjadi ketika tombol kanan <i>mouse</i> ditekan di atas suatu elemen.

Contoh *click*, *dblclick*, dan *contextmenu*:

```

...
<body>
  <input type="button" onclick = "Click()" value="click me"
  id="btn1">

  <input type="button" oncontextmenu = "ctxMenu()" value="right
  click me" id="btn2">

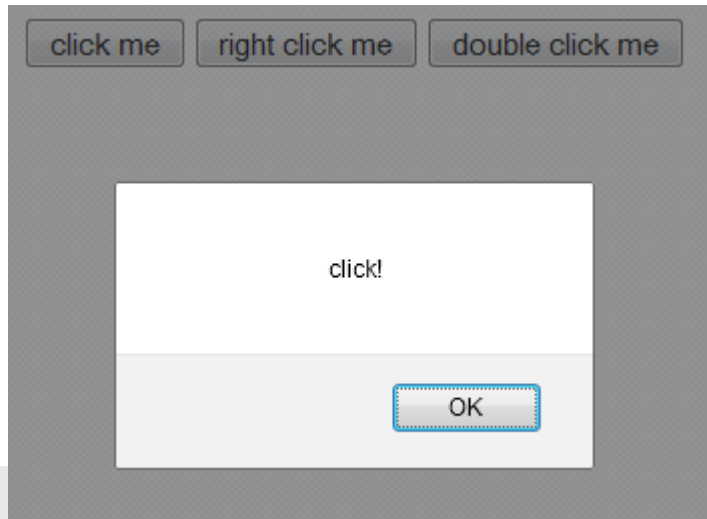
  <input type="button" ondblclick = "dClick()" value="double click
  me" id="btn3">
  <script>

    function Click() {
      alert('click!') //muncul alert pada saat click
    }

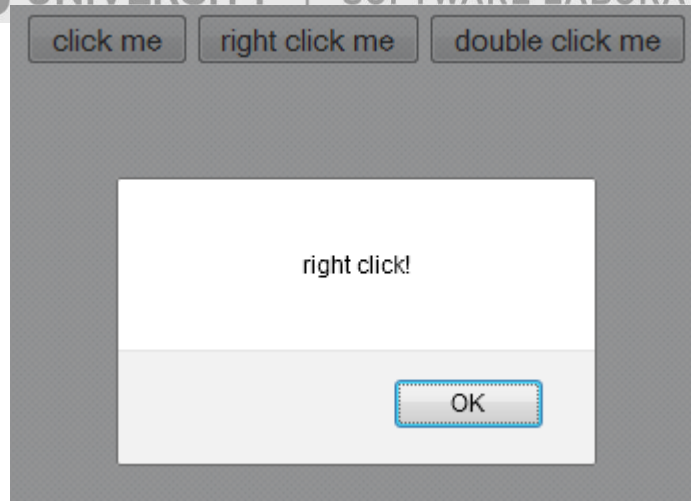
    function ctxMenu() {
      alert('right click!') //muncul alert pada saat right click
    }
  
```

```
function dClick() {  
    alert('double click!')//muncul alert pada saat double click  
}  
  
</script>  
</body>  
...
```

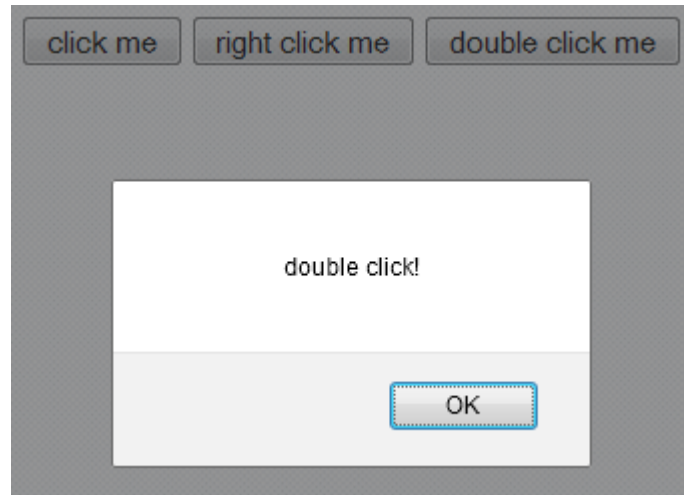
Hasil dari *code*:



**Gambar 7.5** Tampilan saat tombol diklik



**Gambar 7.6** Tampilan saat *right click*



**Gambar 7.7** Tampilan saat *double click*

## 7.2 Keyboard Events

**Keyboard event** adalah *event* yang ter-*trigger* saat suatu aksi dilakukan pada tombol *keyboard* pada suatu elemen. *Keyboard event* ada tiga, yaitu:

Event	Penjelasan
<b>keydown</b>	Ketika suatu tombol pada <i>keyboard</i> ditekan.
<b>keypress</b>	Ketika <i>character key</i> pada <i>keyboard</i> ditekan. <i>Keypress</i> tidak akan ter- <i>trigger</i> pada saat menekan tombol spesial seperti 'Shift', 'Delete' ataupun arah (tergantung <i>browser</i> tertentu).
<b>keyup</b>	Ketika suatu tombol pada <i>keyboard</i> diangkat atau dilepas.

### Contoh *keydown*:

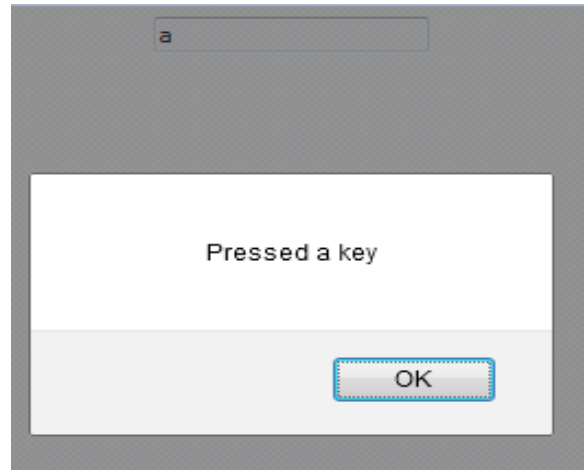
```

...
<input type="text" onkeydown="kDown()">

<script>
    function kDown() {
        alert("Pressed a key");
    }
</script>
...

```

Hasil dari *code*:

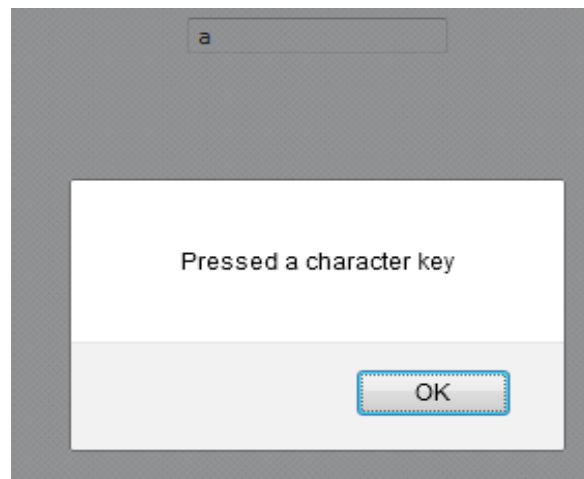


Gambar 7.8 Tampilan saat tombol pada *keyboard* ditekan

Contoh *keypress*:

```
...  
<input type="text" onkeypress="kPress()">  
<script>  
  function kPress() {  
    alert("Pressed a character key");  
  }  
</script>  
...
```

Hasil dari *code*:



Gambar 7.9 Tampilan saat *character key* ditekan

Contoh *keyup*:

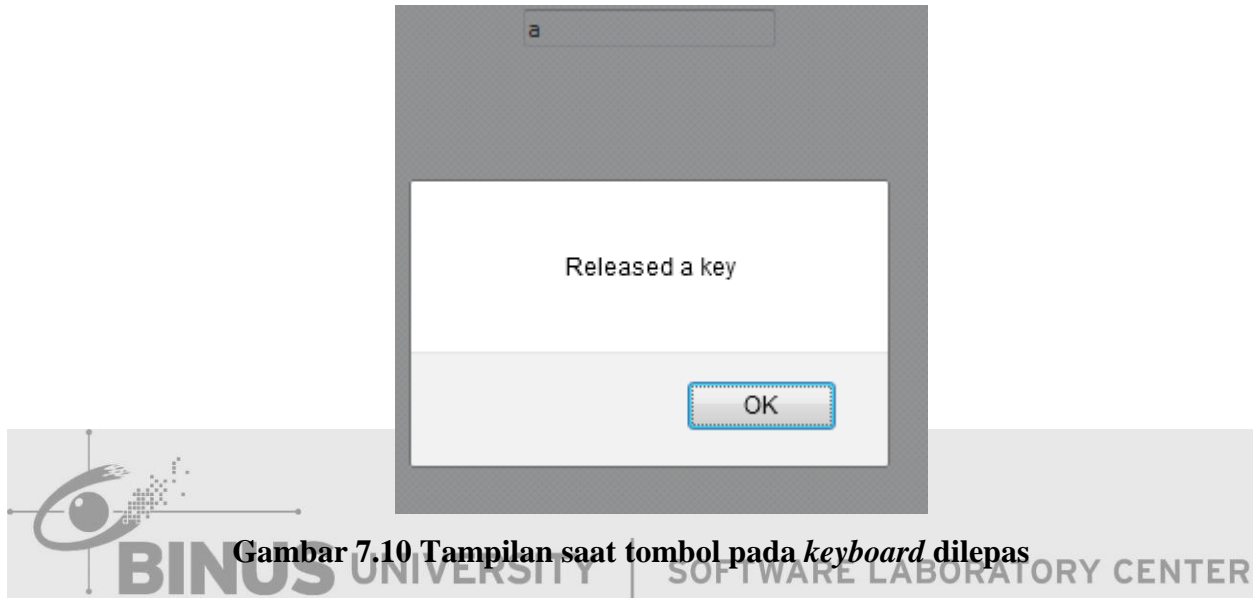
```
...  
<input type="text" onkeyup="kUp()">
```

```

<script>
    function kUp() {
        alert("Released a key");
    }
</script>
...

```

Hasil dari *code*:



Gambar 7.10 Tampilan saat tombol pada *keyboard* dilepas

### 7.3 Canvas Mouse Coordinates

**Canvas Mouse Coordinates** adalah metode untuk mendapatkan posisi *mouse* pada *canvas*. Untuk mendapatkan posisi *mouse* pada *canvas*, dapat dilakukan dengan cara memanggil sebuah fungsi yang mengembalikan koordinat *mouse* berdasarkan posisi dari *mouse* dan posisi dari *canvas*.

Fungsi yang digunakan:

- `getBoundingClientRect()`: untuk mendapatkan posisi dari *canvas*.
- `addEventListener()`: untuk membuat suatu penangkap gerakan yang terjadi pada *canvas*.

**Contoh *canvas mouse coordinates*:**

```

...
<canvas id="myCanvas" width="578" height="200" style="border:1px
solid"></canvas>
<script>
    function writeMessage(canvas, message) { //untuk menulis posisi

```

```

mousenya
    var context = canvas.getContext('2d');
    context.clearRect(0, 0, canvas.width, canvas.height); //
    untuk membersihkan canvasnya
    context.font = '18pt Calibri';
    context.fillStyle = 'black';
    context.fillText(message, 10, 25);
}
function getMousePos(canvas, evt) { // untuk mendapatkan posisi
    dari mouse
    var rect = canvas.getBoundingClientRect();
    return {
        x: evt.clientX - rect.left,
        y: evt.clientY - rect.top
    };
}
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

canvas.addEventListener('mousemove', function(evt) { // memberi
    pendengar untuk event mousemove
    var mousePos = getMousePos(canvas, evt);
    var message = 'Mouse position: ' + mousePos.x + ',' +
    mousePos.y;
    writeMessage(canvas, message);
}, false);
</script>
...

```

Hasil dari code:

Mouse position: 244,124

**Gambar 7.11** Tampilan saat penunjuk *mouse* digerakkan pada *canvas*

Tambahan:

- *evt* adalah *event* yang ditangkap oleh *EventListener*.
- *mousemove* adalah jenis *event* yang ingin ditangkap.

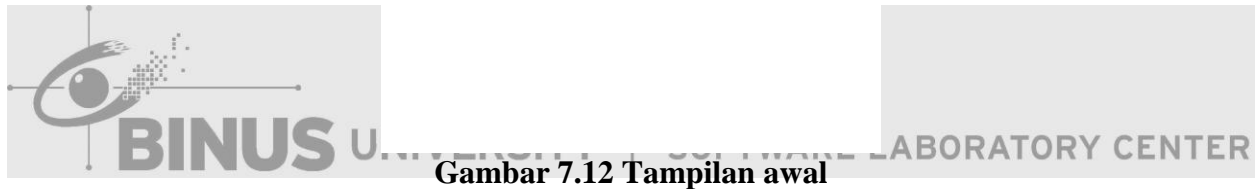


## 7.4 Exercise

Buatlah suatu halaman web yang terdiri dari satu buah **div** dengan ketentuan sebagai berikut:

- **Div** berisi tulisan “**Mouse outside**” (lihat **Gambar 7.12**).
- Saat penunjuk *mouse* berada di dalam div tersebut, ubah tulisan menjadi “**Mouse inside**” dan ubah warna tulisan menjadi warna **merah** (lihat **Gambar 7.13**).
- Saat **div** diklik, ubah tulisan menjadi “**Mouse down**” dan berikan border berwarna **biru** pada **div** (lihat **Gambar 7.14**).
- Saat tombol *mouse* dilepas, ubah tulisan menjadi “**Mouse up**” dan ubah warna border menjadi **hitam** (lihat **Gambar 7.15**).

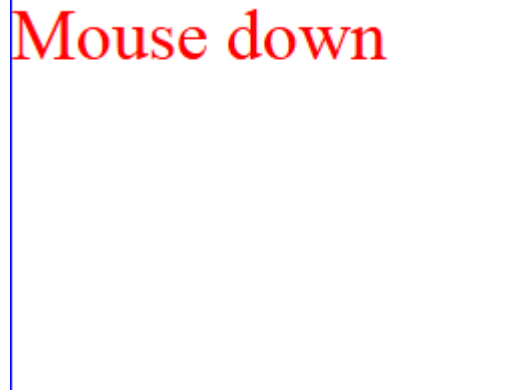
Mouse outside



Gambar 7.12 Tampilan awal

Mouse inside

Gambar 7.13 Tampilan saat penunjuk *mouse* masuk ke area div



Mouse down

Gambar 7.14 Tampilan saat *mouse* diklik pada area **div**



Mouse up

Gambar 7.15 Tampilan saat tombol *mouse* dilepas

Berikut ini cara untuk mengerjakannya:

- a. Buatlah teks dengan tulisan “**Mouse outside**” pada suatu **div**.

```
...
<body>
  <div style="font-size:36px; width:250px; height:200px">Mouse
  outside</div>
</body>
...
```

- b. Kemudian buat *function* untuk semua *event*-nya

```
...
<script>
  function mOver(obj) {
    obj.innerHTML = "Mouse inside"//mengubah isi dalam tag
    object yang diberikan
    obj.style.color = "red"// mengubah warna fontnya
  }
  function mOut(obj) {
    obj.innerHTML = "Mouse outside"
    obj.style.color = "black";
  }
}
```

```

function mDown(obj){
    obj.innerHTML = "Mouse down";
    obj.style.border = "1px solid blue"// memberikan border
    biru di sekeliling
}
function mUp(obj){
    obj.innerHTML = "Mouse up";
    obj.style.border = "1px solid black"// membuat bordernya
    menjadi hitam
}
</script>
...

```

- c. Kemudian tambahkan *event*-nya pada **div** yang sebelumnya sudah dibuat.

```

<html>
<head>
  <head>
    <title>Latihan</title>
    <script>
      function mOver(obj) {
        obj.innerHTML = "Mouse inside"//mengubah isi dalam
        tag object yang diberikan
        obj.style.color = "red"// mengubah warna fontnya
      }
      function mOut(obj) {
        obj.innerHTML = "Mouse outside"
        obj.style.color = "black";
      }
      function mDown(obj){
        obj.innerHTML = "Mouse down";
        obj.style.border = "1px solid blue"// memberikan
        border biru di sekeliling div-nya
      }
      function mUp(obj){
        obj.innerHTML = "Mouse up";
        obj.style.border = "1px solid black"// membuat
        bordernya menjadi warna hitam
      }
    </script>
  </head>

  <body>
    <div onmouseup="mUp(this)" onmousedown="mDown(this)"
    onmouseover="mOver(this)" onmouseout="mOut(this)"
    style="font-size:36px; width:250px; height:200px">Mouse
    outside</div>
  </body>
</html>

```



## 8.1 Local Storage

**Local storage** adalah penyimpanan data pada *browser* mirip dengan *cookie*. Perbedaan antara keduanya adalah *cookie* sendiri mempunyai kapasitas maksimal sebesar 4KB sedangkan *local storage* mempunyai lebih dari 5-10 MB di setiap *browser*. Disamping itu, *local storage* dapat di implementasikan di *web apps* dan *mobile apps*. Kelebihan lainnya adalah mengurangi data yang di-request *client* dari *server* sehingga akan menghemat *bandwidth* dari *server* itu sendiri.

Fungsi-fungsi yang terapat pada *local storage*:

Fungsi	Parameter	Kegunaan
<code>localStorage.setItem(key,value);</code> <code>localStorage.key = value;</code>	<b>key:</b> kata kunci untuk <i>value</i> yang disimpan. <b>value:</b> data yang ingin disimpan.	Menyimpan data ke <i>browser</i> .
<code>localStorage.getItem(key);</code> <code>localStorage.key;</code>	<b>key:</b> kata kunci dari data yang sudah disimpan.	Mengambil data.
<code>localStorage.removeItem(key);</code>	<b>key:</b> kata kunci dari data yang sudah disimpan.	Mehapus data.
<code>localStorage.length;</code>	-	Melihat jumlah dari <i>local storage</i> yang ada.
<code>localStorage.clear();</code>	-	Menghapus semua <i>local storage</i> yang ada.

Cara penggunaan *local storage*:

```
<!DOCTYPE HTML>
<html>
  <body>
    <script type="text/javascript">
      // mengecek apakah sudah ada data didalam local storage
      if( localStorage.hits ){
        // jika sudah ada ditambah 1
        localStorage.hits = Number(localStorage.hits) +1;
      }else{
        // jika belum akan dibuat dan dinisialisasikan menjadi 1
        localStorage.hits = 1;
      }
      document.write("Total Hits :" + localStorage.hits );
    </script>
  </body>
</html>
```

```

</script>
<p>Refresh the page to increase number of hits.</p>
<p>Close the window and open it again and check the result.</p>
</body>
</html>

```

Hasil dari *code* diatas:

Total Hits :1

Refresh the page to increase number of hits.

Close the window and open it again and check the result.

Gambar 8.1 Contoh *local storage*

## 8.2 Web SQL

Selain *local storage*, ada juga penyimpanan *data* dengan cara lain, yaitu **Web SQL**. **Web SQL** memiliki kelebihan yaitu penyimpanan *data* yang berbentuk *database*, sehingga bisa berisi tabel-tabel.

Berikut adalah cara untuk membuat *database* dan tabel:

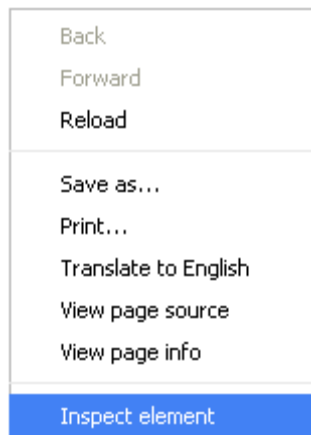
```

<!DOCTYPE HTML>
<html>
  <body>
    <script type="text/javascript">
      //untuk membuat / membuka database
      var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 *
1024);
      db.transaction(function (tx) {
        // membuat tabel baru jika belum ada
        tx.executeSql('CREATE TABLE IF NOT EXISTS Mahasiswa (id ,
name)');
      });
    </script>
  </body>
</html>

```

Untuk melihat Web SQL pada **Google Chrome** berdasarkan hasil dari *code* tersebut:

1. Klik kanan pada halaman web kemudian pilih *Inspect Element*



**Gambar 8.2 Tampilan menu saat melakukan klik kanan**

2. Pilih tab *Resources* kemudian pilih Web SQL.



**Gambar 8.3 Tampilan setelah *inspect element***

Fungsi-fungsi yang digunakan adalah:

Fungsi	Attribute	Kegunaan
<b>openDatabase(name, version, description, size);</b>	<b>name:</b> nama dari <i>database</i> . <b>version:</b> versi dari <i>database</i> . <b>description:</b> deskripsi untuk <i>database</i> . <b>size :</b> ukuran dari <i>database</i> dalam satuan <i>bytes</i> .	Membuat (jika belum ada) atau mengambil <i>database</i> .
<b>transaction(function);</b>	<b>function</b> : fungsi yang berisi fungsi-fungsi SQL yang akan digunakan.	Mengontrol sebuah transaksi SQL.

<b>executeSql(query);</b>	<b>query:</b> perintah-perintah SQL yang ingin di jalankan.	Mengeksekusi <i>query</i> pada SQL.
<b>executeSql(query, array_data, success_function, failed_function);</b>	<b>query:</b> perintah-perintah SQL yang ingin di jalankan. <b>array_data:</b> data yang dipakai di <i>query</i> . <b>success_function:</b> fungsi yang akan dijalankan saat <i>query</i> sukses. <b>failed_function:</b> fungsi yang akan dijalankan saat <i>query</i> gagal.	Mengeksekusi <i>prepared statement</i> pada SQL.

### 8.2.1 Store Data

Contoh penyimpanan data menggunakan Web SQL:

```

<!DOCTYPE HTML>
<html>
  <body>
    <script type="text/javascript">
      //untuk membuat / membuka database
      var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024
        * 1024);
      db.transaction(function (tx) {
        // membuat tabel baru jika belum ada
        tx.executeSql('CREATE TABLE IF NOT EXISTS Mahasiswa
(id , name)');
        // memasukkan data ke tabel
        tx.executeSql('INSERT INTO Mahasiswa (id, name) VALUES
(1, "Andi")');
        tx.executeSql('INSERT INTO Mahasiswa (id, name) VALUES
(2, "Budi")');
      });
    </script>
  </body>
</html>

```

### 8.2.2 Get Data

Contoh mengambil data dari tabel yang telah buat diatas:

```

<!DOCTYPE HTML>
<html>
  <head>
    <script type="text/javascript">

```



```

//untuk membuat / membuka database
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024
* 1024);

db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS
Mahasiswa (id , name)');
    // memasukkan data ke tabel
    tx.executeSql("INSERT INTO Mahasiswa (id, name)
VALUES (1, 'Andi')");
    tx.executeSql("INSERT INTO Mahasiswa (id, name)
VALUES (2, 'Budi')");
    });

db.transaction(function (tx) {
    // mengambil data dari database
    tx.executeSql('SELECT * FROM Mahasiswa', [], function
(tx, results) {
        // mengecek berapa jumlah baris didatabase
        var len = results.rows.length;
        msg = "<p>Found rows: " + len + "</p>";
        document.querySelector('#status').innerHTML += msg;
        for (i = 0; i < len; i++){
            //mengambil item ke-i dan kolom name
            msg = "<p><b>" + results.rows.item(i).name +
"</b></p>";
            document.querySelector('#status').innerHTML += msg;
        }
    }, null);
    });
</script>
</head>
<body>
    <div id="status" name="status"></div>
</body>
</html>

```

Hasil dari *code* diatas:

Found rows: 2

**Adi**

**Budi**

Gambar 8.4 Contoh *get data*

### 8.2.3 Update and Delete data

Contoh meng-*update* dan *delete* data dari tabel yang telah dibuat

```
<!DOCTYPE HTML>
<html>
  <head>
    <script type="text/javascript">
      //untuk membuat / membuka database
      var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024
        * 1024);

      // memasukkan data ke tabel (cukup sekali dijalankan)
      db.transaction(function (tx) {
        tx.executeSql('CREATE TABLE IF NOT EXISTS Mahasiswa
          (id , name)');

        tx.executeSql("INSERT INTO Mahasiswa (id, name) VALUES
          (1, 'Andi')");
        tx.executeSql("INSERT INTO Mahasiswa (id, name) VALUES
          (2, 'Budi')");
      });

      //untuk menghapus data dari database
      function deleteId(id){
        db.transaction(function (tx) {
          tx.executeSql("DELETE FROM Mahasiswa WHERE id =
            "+id);
        });
        refresh();
      }

      //untuk melakukan update dari database
      function updateId(id){
        var name = document.getElementById('name'+id).value;
        db.transaction(function (tx) {
          tx.executeSql('UPDATE Mahasiswa SET name =
```

```

    "+name+" WHERE id = '+id);
    });
    refresh();
}

//untuk mengambil data
function refresh(){
    db.transaction(function (tx) {
        // mengambil data dari database
        tx.executeSql('SELECT * FROM Mahasiswa', [], function
(tx, results) {
            // mengecek berapa jumlah baris didatabase
            var len = results.rows.length;
            msg = "<p>Found rows: " + len + "</p>";
            document.querySelector('#status').innerHTML = msg;
            for (i = 0; i < len; i++){
                //mengambil item ke-i dan kolom name
                msg = "<p><b>" + results.rows.item(i).name +
"</b></p>";
                //membuat button delete untuk setiap datanya
                msg += '<button onclick =
"deleteId('+results.rows.item(i).id+') ">Delete</button>';
                //membuat textbox dan button update untuk setiap
datanya
                msg += '<input type="text"
id="name'+results.rows.item(i).id+'"> <button onclick =
"updateId('+results.rows.item(i).id+')"> Update </button>';
                document.querySelector('#status').innerHTML +=
msg;
            }
        }, null);
    });
}
refresh();
</script>
</head>
<body>
    <div id="status" name="status"></div>
</body>
</html>

```

Hasil dari code diatas:

Found rows: 2

**Agus**

Delete  Update

**Budi**

Delete  Update

### 8.3 Exercise

Buatlah *web* untuk menyimpan data dari *input user*, menampilkan jumlah baris pada data yang ada, dan menampilkan data apa saja yang disimpan. Ketika tombol “**insert**” ditekan, maka data akan bertambah.

Adi  insert

Found rows: 1

**1. Adi**

**Gambar 8.5 Hasil latihan**

Langkah-langkah:

1. Membuat *textbox*, *button* dan *div* untuk menampung hasil.

```
<!DOCTYPE HTML>
<html>
  <head></head>
  <body>
    <input type="text" id="name">
```

```

        <input type="button" onClick="insertData()" value="insert">
        <div id="status" name="status"></div>
    </body>
</html>

```

2. Membuat Web SQL dan *local storage* untuk penyimpanan data.

```

<script type="text/javascript">
    if( !localStorage.MId ){
        localStorage.MId = 1; //untuk counter dari data
    }

    var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024

    db.transaction(function (tx) {
        tx.executeSql('CREATE TABLE IF NOT EXISTS Mahasiswa (id ,
name)');
    });
</script>

```

3. Membuat fungsi untuk mengambil data dari Web SQL.

```

<script type="text/javascript">
    function refreshData(){
        db.transaction(function (tx) {
            tx.executeSql('SELECT * FROM Mahasiswa', [], function (tx,
results) {
                var len = results.rows.length;
                msg = "<p>Found rows: " + len + "</p>";
                document.querySelector('#status').innerHTML = msg;

                for (i = 0; i < len; i++){
                    msg = "<p><b>" + results.rows.item(i).id+ ".
"+results.rows.item(i).name + "</b></p>";
                    document.querySelector('#status').innerHTML += msg;
                }
            }, null);
        });
        refreshData();// dipanggil saat awal membuka page
    }
</script>

```

4. Membuat fungsi untuk melakukan *input data*.

```

<script type="text/javascript">
    function insertData(){
        var name = document.getElementById('name').value;
        db.transaction(function (tx) {
            tx.executeSql('INSERT INTO Mahasiswa (id, name) VALUES (?,

```

```
?')', [Number(localStorage.MId), name]);  
    localStorage.MId = Number(localStorage.MId)+1; //id bertambah  
    refreshData();  
  });  
}  
</script>
```



## 9.1 Introduction HTML5 Audio and Video

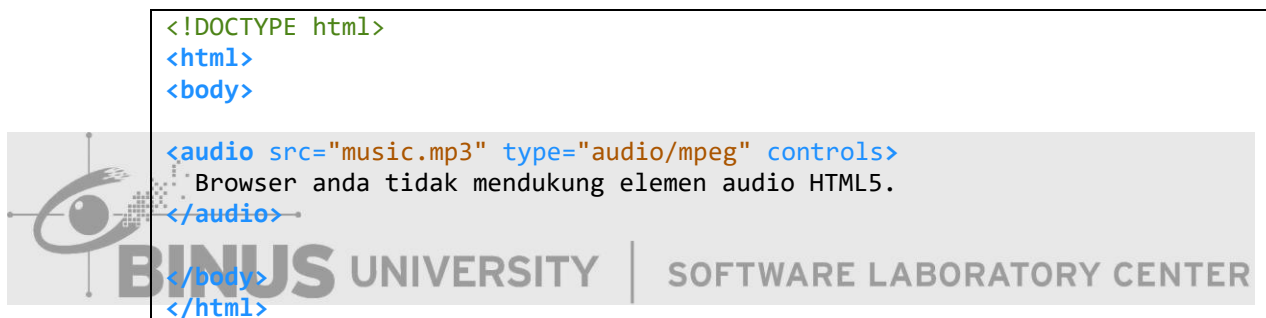
HTML5 memiliki sebuah tag yang dapat mendukung media secara langsung menggunakan tag `<audio>` dan `<video>`, yang memudahkan orang untuk menambah audio dan video kedalam dokumen HTML yang telah dibuat.

### 9.1.1 Audio

Tag `<audio>` pada HTML5 mendukung beberapa format audio seperti:

1. MP3
2. Ogg
3. WAV

Contoh penggunaan tag `<audio>`:



Jika *browser* tidak mendukung tag `<audio>` maka teks yang berada di dalam tag `<audio>` akan muncul.

Hasil dari *code*:



**Gambar 9.1 Penggunaan audio**

Berikut adalah atribut-atribut yang dimiliki oleh tag `<audio>`:

Atribut	Value	Kegunaan
<b>autoplay</b>	-	Audio akan langsung diputar ketika halaman tersebut dibuka.
<b>controls</b>	-	Membuat <i>control</i> untuk audio yang



		dimasukkan seperti <i>play</i> , <i>pause</i> dan <i>volume control</i> .
<b>loop</b>	-	Audio akan langsung diputar kembali setiap audio itu selesai diputar.
<b>muted</b>	-	Mengubah <i>default volume</i> dari audio menjadi <i>mute</i> /tidak bersuara
<b>preload</b>	Dapat diisi dengan “ <b>auto</b> ”, “ <b>metadata</b> ”, atau “ <b>none</b> ”	<p><b>auto:</b> <i>browser</i> akan langsung membuat audio ketika halaman web dibuka.</p> <p><b>metadata:</b> <i>browser</i> akan membuat metadata dari audio saja ketika halaman web dibuka.</p> <p><b>none:</b> <i>browser</i> tidak akan membuat audio ketika halaman web dibuka.</p>
<b>src</b>	Diisi dengan sumber lokasi audio	Menunjukkan sumber/lokasi dari audio yang ingin dimasukkan ke dokumen HTML.

Contoh penggunaan atribut untuk tag `<audio>`:

```

<!DOCTYPE html>
<html>
<body>
ketika page load, musik akan langsung diputar dan ketika music berakhir
akan langsung diputar kembali
<audio src="music.mp3" type="audio/mpeg" autoplay loop controls>
  Browser anda tidak mendukung elemen audio HTML5.
</audio>

</body>
</html>

```

### 9.1.2 Video

Untuk tag `<video>` HTML5 dapat membaca beberapa format video seperti:

1. MP4
2. WebM
3. Ogg

Contoh penggunaan *tag* <video>:

```
<!DOCTYPE html>
<html>
<body>

<video src="video.mp4" type="video/mp4" controls>
  Browser anda tidak mendukung elemen audio HTML5.
</video>

</body>
</html>
```

Jika *browser* tidak mendukung *tag* <video> maka teks yang berada di dalam *tag* <video> akan muncul.

Hasil dari *code*:



**Gambar 9.2 Penggunaan video**

Berikut adalah atribut-atribut yang dimiliki oleh *tag* <audio>:

Atribut	Value	Kegunaan
<b>autoplay</b>	-	Video akan langsung diputar ketika halaman tersebut dibuka.
<b>controls</b>	-	Membuat <i>control</i> untuk video yang

		dimasukkan seperti <i>play</i> , <i>pause</i> dan <i>volume control</i> .
<b>height</b>	ukuran (secara default dalam piksel)	Menentukan tinggi dari <i>video player</i> .
<b>loop</b>	-	Video akan langsung diputar kembali setiap video itu selesai diputar.
<b>muted</b>	-	Mengubah <i>default volume</i> dari video menjadi <i>mute</i> /tidak bersuara.
<b>poster</b>	<i>URL</i>	Menunjukkan gambar pada <i>video player</i> ketika video sedang di- <i>download</i> atau sebelum user memencet tombol <i>play</i> .
<b>preload</b>	Dapat diisi dengan “ <b>auto</b> ”, “ <b>metadata</b> ”, atau “ <b>none</b> ”	<p><b>auto:</b> <i>browser</i> akan langsung membuat video ketika halaman web dibuka.</p> <p><b>metadata:</b> <i>browser</i> akan membuat metadata dari video saja ketika halaman web dibuka.</p> <p><b>none:</b> <i>browser</i> tidak akan membuat video ketika halaman web dibuka.</p>
<b>src</b>	Diisi dengan sumber lokasi video	Menunjukkan sumber/lokasi dari video yang ingin dimasukkan ke dokumen HTML.
<b>width</b>	ukuran (secara default dalam piksel)	Menentukan lebar dari <i>video player</i> .

Contoh penggunaan atribut untuk tag `<video>`:

```
<!DOCTYPE html>
<html>
<body>
mengatur agar ukuran video menjadi 100x100
<video src="video.mp4" type="video/mp4" width="100px" height="100px"
controls>
  Browser anda tidak mendukung elemen audio HTML5.
</video>
</body>
</html>
```

## 9.2 Adding User Control

Untuk menambahkan *user control*, dapat menggunakan atribut yang sudah disediakan pada HTML5 atau menggunakan *JavaScript*.

Contoh dengan menggunakan HTML5:

```
...
<audio src="music.mp3" type="audio/mpeg" controls></audio>
...
```

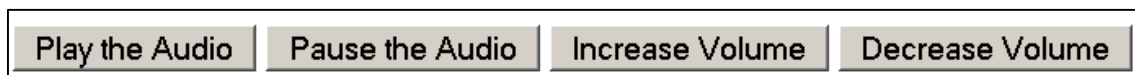
Untuk *JavaScript*, sudah disediakan fungsi yang dapat digunakan:

Fungsi	Kegunaan
<b>play();</b>	Untuk memutar audio atau video.
<b>pause();</b>	Untuk melakukan <i>pause</i> pada audio atau video.
<b>volume</b>	Mengatur <i>volume</i> .
<b>currentTime</b>	Mengatur posisi waktu dari audio atau video.

Contoh menggunakan *JavaScript*:

```
...
<audio id="demo" src="music.mp3" type="audio/mpeg"></audio>
<div>
  //Untuk menjalankan audio
  <button onclick="document.getElementById('demo').play()">Play the
Audio</button>
  //Untuk melakukan pause pada audio
  <button onclick="document.getElementById('demo').pause()">Pause the
Audio</button>
  //Untuk membesarkan volume
  <button onclick="document.getElementById('demo').volume+=0.1">Increase
Volume</button>
  //Untuk mengecilkan volume
  <button onclick="document.getElementById('demo').volume-=0.1">Decrease
Volume</button>
</div>
...
```

Hasil dari kode diatas:



**Gambar 9.3 Mengontrol audio menggunakan *JavaScript***

### 9.3 Preloading Audio and Video

Atribut *preload* digunakan untuk menentukan bagaimana suatu video atau audio dimuat ketika suatu halaman *web* dibuka.

*Preload* dapat diubah menjadi:

Value	Keterangan
<b>auto</b>	<i>browser</i> akan langsung memuat audio ketika halaman <i>web</i> dibuka.
<b>metadata</b>	<i>browser</i> akan memuat metadata saja ketika halaman <i>web</i> dibuka
<b>none</b>	<i>browser</i> tidak akan memuat audio ketika halaman <i>web</i> dibuka.

Contoh penggunaan *preload*:

```
<!DOCTYPE html>
<html>
<body>
//harus menekan tombol play terlebih dahulu untuk menjalankan
<video width="320" height="240" controls preload="none">
  <source src="movie.mp4" type="video/mp4">
  Browser anda tidak mendukung elemen video HTML5.
</video>
//video akan langsung dimuat ketika
<video width="320" height="240" controls preload="auto">
  <source src="movie.mp4" type="video/mp4">
  Browser anda tidak mendukung elemen video HTML5.
</video>
<p><b>Tambahan:</b> Attribute preload tidak didukung oleh Internet Explorer.</p>
</body>
</html>
```

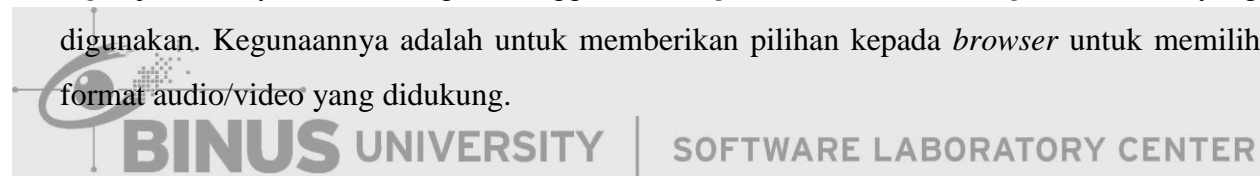
Hasil dari *code*:



**Gambar 9.4** Contoh penggunaan *preload*

## 9.4 Specifying Multiple Source Files

Pada HTML5 dapat dicantumkan lebih dari satu buah *source* audio/video dalam satu *tag* saja. Caranya adalah dengan menggunakan *tag* `<source>` didalam *tag* audio/video yang digunakan. Kegunaannya adalah untuk memberikan pilihan kepada *browser* untuk memilih format audio/video yang didukung.



Contoh penggunaan *multiple source*:

```
...
<video controls>
  <source src="video.ogg" type="video/ogg">
  <source src="video.mp4" type="video/mp4">
  Browser anda tidak mendukung tag video
</video>
...
```

## 9.5 Controlling Playback (volumes, audio events, customer UI controls)

### 9.5.1 Volume

Pengaturan *volume* di HTML5 dapat dilakukan melalui dua cara yaitu melalui atribut *controls* pada HTML5 dan dengan menggunakan *JavaScript*. Jika kita menggunakan *JavaScript*, properti yang digunakan adalah:

Atribut	Nilai	Kegunaan
<b>volume</b>	Nilai numerik	Menentukan <i>volume</i> dari

		audio atau video
<b>muted</b>	<b>Boolean</b> berupa <b>true</b> atau <b>false</b>	Apakah audio di- <i>mute</i> atau tidak

Contoh penggunaan *volume*:

```

...
<audio id="musicplayer" src="music1.mp3" controls></audio>
<br>
<button
onclick="document.getElementById('musicplayer').volume+=0.1">+</button>
<button onclick="document.getElementById('musicplayer').volume-=0.1">-
</button>
<style>
    button{
        border-radius: 13px;
        font-family: Arial;
        color: #ffffff;
        font-size: 10px;
        background: #3498db;
        padding: 5px 10px 5px 10px;
        text-decoration: none;
    }
</style>
...

```

Jika tombol “+” ditekan, maka *volume* akan bertambah, dan jika tombol “-” ditekan maka volume akan berkurang.

Hasil dari *code* tersebut:



**Gambar 9.5** Contoh penggunaan *volume*

### 9.5.2 Audio/Video Events

Audio atau video pada HTML5 juga memiliki *events* yang dapat digunakan menggunakan *JavaScript*. Berikut adalah beberapa *events* ada:

Event	Penjelasan
<b>abort</b>	Ketika <i>loading</i> dari audio/video dibatalkan
<b>canplay</b>	Ketika <i>browser</i> dapat memulai memutar audio/video

<b>canplaythrough</b>	Ketika <i>browser</i> dapat memulai memutar audio/video tanpa harus <i>buffering</i> lagi
<b>durationchange</b>	Ketika durasi dari audio/video diubah
<b>emptied</b>	Ketika <i>playlist</i> sedang kosong
<b>ended</b>	Ketika <i>playlist</i> sudah berakhir
<b>error</b>	Ketika terjadi <i>error</i> saat memuat audio/video
<b>loadeddata</b>	Ketika <i>browser</i> sudah memuat audio/video
<b>loadedmetadata</b>	Ketika <i>browser</i> sudah memuat <i>meta data</i> dari audio/video
<b>loadstart</b>	Ketika <i>browser</i> bari ingin memuan audio/video
<b>pause</b>	Ketika audio/video di- <i>pause</i>
<b>play</b>	Ketika audio/video di- <i>play</i>
<b>playing</b>	Ketika audio/video sedang di- <i>play</i>
<b>progress</b>	Ketika <i>browser</i> sedang men- <i>download</i> audio/video
<b>ratechange</b>	Ketika kecepatan audio/video diubah
<b>seeked</b>	Ketika <i>user</i> selesai mengganti posisi audio/video yang sedang diputar
<b>seeking</b>	Ketika <i>user</i> sedang mengganti posisi audio/video yang sedang diputar
<b>stalled</b>	Ketika <i>browser</i> sedang mencoba mengambil data dari audio/video tapi data tidak tersedia
<b>suspend</b>	Ketika <i>browser</i> dengan sengaja tidak mengambil data audio/video
<b>timeupdate</b>	Ketika posisi dari durasi audio/video diubah
<b>volumechange</b>	Ketika i diubah
<b>waiting</b>	Ketika audio/video sedang <i>buffering</i> /membaca data selanjutnya

Contoh penggunaan *audio/video event*:

```

...
<audio id="musicplayer" src="music1.mp3" controls
onended="alertMe()"></audio>
<script>
    function alertMe(){//ketika selesai diputar muncul alert
        alert("audio selesai diputar");
    }

```

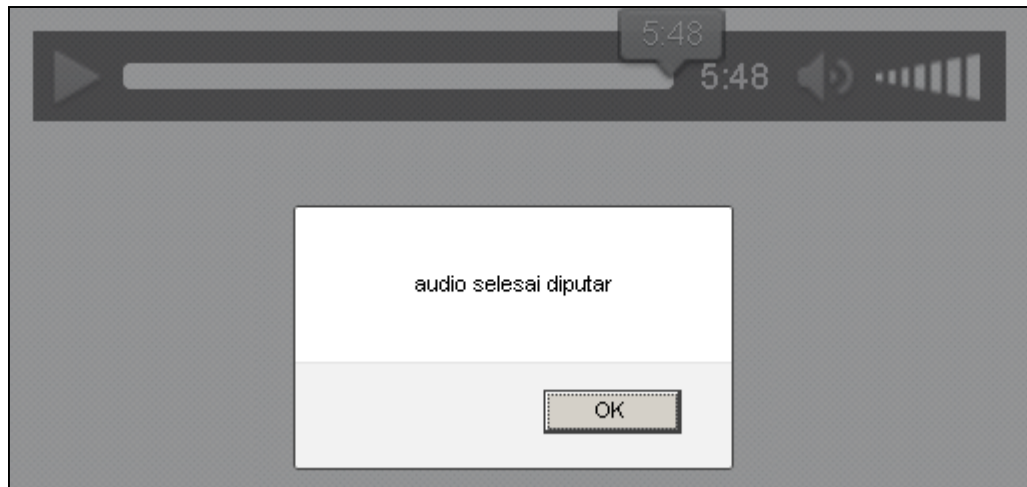


```

</script>
...

```

Hasil dari *code* tersebut:



**Gambar 9.6** Contoh penggunaan *event* pada audio

### 9.5.3 Custom UI Controls

Tampilan dari *controls* pada audio dapat diubah sesuai dengan keinginan. Hanya perlu menerima status dari pemutaran audio tersebut, dan kemudian menampilkannya pada dokumen HTML yang ada. Agar lebih menarik, dapat ditambahkan CSS. Selain itu, dapat juga membuat *playlist* sederhana pada dokumen HTML.

Contoh membuat *playlist* sederhana:

```

...
<style>
    #audioplayer{
        width:200;
        border: solid;
        padding:10;
    }
    #prButton,#pButton,#nButton{
        border-radius: 13px;
        font-family: Arial;
        color: #ffffff;
        font-size: 10px;
        background: #3498db;
        padding: 5px 10px 5px 10px;
        text-decoration: none;
    }
</style>
<audio id="music" ontimeupdate="updateTime()">
    <source src="music1.mp3">
</audio>

```

```

<div id="audioplayer">
  <p id="NP">Now Playing : music1.mp3</p>
  <p id="duration">00:00</p>
  <button id="prButton" class="prev" onclick="prev()">Prev</button>
  <button id="pButton" class="play" onclick="play()">Play</button>
  <button id="nButton" class="next" onclick="next()">Next</button>
</div>
<script>
  var music = document.getElementById('music');
  var pButton = document.getElementById('pButton');
  var playlist=["music1.mp3","music2.mp3","music3.mp3"];
  var nowplaying = document.getElementById('NP');
  var cntr = 0;
  //untuk forming waktunya
  function formatSecondsAsTime(secs, format) {
    var hr = Math.floor(secs / 3600);
    var min = Math.floor((secs - (hr * 3600))/60);
    var sec = Math.floor(secs - (hr * 3600) - (min * 60));

    if (min < 10){
      min = "0" + min;
    }
    if (sec < 10){
      sec = "0" + sec;
    }

    return min + ':' + sec;
  }
  //untuk display durasi sekarang
  function updateTime(){
    duration.innerHTML = formatSecondsAsTime(music.currentTime);
  }
  function next(){
    if(cntr<playlist.length-1){//jika masih ada lagu dilist
      cntr++;//majukan counter
    }
    else{
      cntr=0;//pilih lagu pertama
    }
    music.src= playlist[cntr];
    music.load();//mereLoad lagu
    play();
  }
  function prev(){
    if(cntr!=0){//jika bukan lagu pertama
      cntr--;//mundurkan counter
    }
    else{
      cntr=playlist.length-1;//pilih lagu terakhir
    }
    music.src= playlist[cntr];
    music.load();//mereLoad lagu
    play();
  }
}

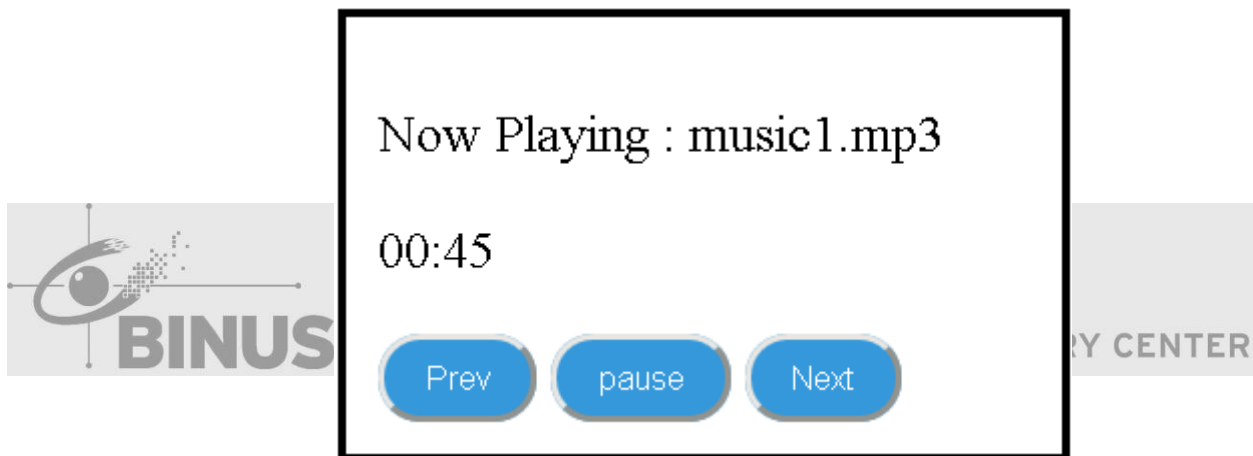
```

```

function play() {
    nowplaying.innerHTML = "Now Playing : "+playlist[ctr];
    // play audio
    if (music.paused) {
        music.play();
        pButton.innerHTML = "pause";//mengubah tulisan
    } else { // pause audio
        music.pause();
        pButton.innerHTML = "play";//mengubah tulisan menjadi
    }
}
</script>
...

```

Hasil dari *code* tersebut:



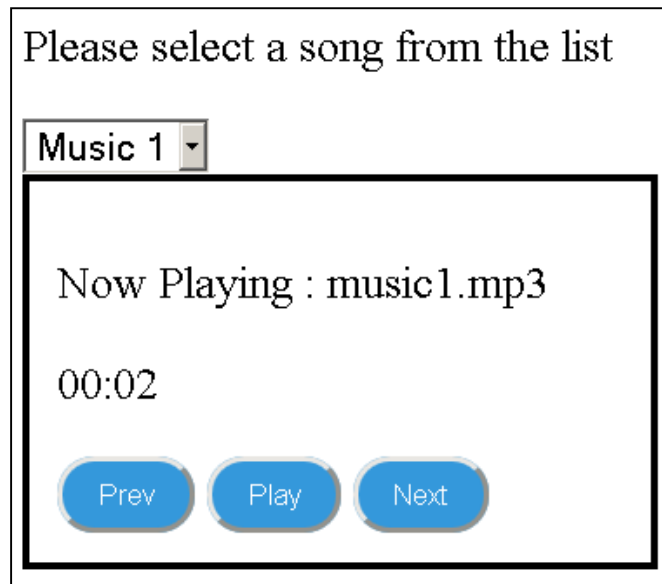
**Gambar 9.7** Contoh membuat *playlist* sederhana

## 9.6 Exercise

Buatlah suatu pemutar musik yang dapat memainkan suatu lagu tertentu. Pemutar musik ini memiliki beberapa tombol untuk mengontrol musik yang dimainkan. Berikut beberapa kontrol yang diperlukan:

1. Komponen untuk memilih lagu yang ingin dimainkan.
2. Nama dokumen musik yang sedang dimainkan.
3. Durasi musik yang sedang dimainkan.
4. Tombol “**Prev**” untuk memainkan musik sebelumnya.
5. Tombol “**Play**” untuk memainkan musik.
6. Tombol “**Next**” untuk memainkan musik selanjutnya.

Berikut adalah tampilan dari pemutar musik tersebut:



**Gambar 9.8 Hasil pemutar musik**

Berikut ini adalah langkah-langkah untuk mengerjakan:

1. Membuat HTML untuk *audio player*, *nowplaying*, list lagu, dan tombol.

```
<p>Please select a song from the list</p>
<select name=mylist id=mylist onChange="play()">
  <option id="0" value="0">Music 1</option>
  <option id="1" value="1">Music 2</option>
  <option id="2" value="2">Music 3</option>
</select>

<audio id="music" ontimeupdate="updateTime()">
  <source src="music1.mp3">
</audio>
<div id="audioplayer">
  <p id="NP">Now Playing : music1.mp3</p>
  <p id="duration">00:00</p>
  <button id="prButton" class="prev" onclick="prev()">Prev</button>
  <button id="pButton" class="play" onclick="play()">Play</button>
  <button id="nButton" class="next" onclick="next()">Next</button>
</div>
...
```

2. Membuat CSS untuk desain pemutar musik.

```
<style>
  #audioplayer{
    width:200;
    border: solid;
    padding:10;
  }
```

```

#prButton,#pButton,#nButton{
    border-radius: 13px;
    font-family: Arial;
    color: #ffffff;
    font-size: 10px;
    background: #3498db;
    padding: 5px 10px 5px 10px;
    text-decoration: none;
}
</style>

```

### 3. Membuat *JavaScript* untuk mengontrol musik.

```

<script>

var audioURL = document.getElementById('mylist');
var music = document.getElementById('music');
var pButton = document.getElementById('pButton');
var playlist=["music1.mp3","music2.mp3","music3.mp3"];
var nowplaying = document.getElementById('NP');
var cntr = 0;
//untuk formatting waktunya
function formatSecondsAsTime(secs, format) {
    var hr = Math.floor(secs / 3600);
    var min = Math.floor((secs - (hr * 3600))/60);
    var sec = Math.floor(secs - (hr * 3600) - (min * 60));

    if (min < 10){
        min = "0" + min;
    }
    if (sec < 10){
        sec = "0" + sec;
    }

    return min + ':' + sec;
}
//untuk display waktu sekarang
function updateTime(){
    duration.innerHTML = formatSecondsAsTime(music.currentTime);
}
function next(){
    if(cntr<playlist.length-1){//jika masih ada lagu dilist
        cntr++;//majukan counter
    }
    else{
        cntr=0;//pilih lagu pertama
    }
    audioURL.getElementsByTagName('option')[cntr].selected =
'selected';
    music.src= playlist[cntr];
    music.load();//mere-load lagu
    play();
}
function prev(){

```

```

        if(cntr!=0){//jika bukan lagu pertama
            cntr--;//mundurkan counter
        }
        else{
            cntr=playlist.length-1;//pilih lagu terakhir
        }
        audioURL.getElementsByTagName('option')[cntr].selected =
'selected';
        music.src= playlist[cntr];
        music.load();//mere-load lagu
        play();
    }

    function play() {
        if(audioURL.value!=cntr){//mengecek apakah listnya berubah apa
            cntr = audioURL.value;
            music.src= playlist[cntr];
            music.load();
        }
        nowplaying.innerHTML = "Now Playing : "+playlist[cntr];
        // play audio
        if (music.paused) {
            music.play();
            pButton.innerHTML = "pause";//ubah jadi pause
        } else { // pause audio
            music.pause();
            pButton.innerHTML = "play";//ubah jadi play
        }
    }
}
</script>

```

#### 4. Hasil akhir:

```

...
<p>Please select a song from the list</p>
<select name=mylist id=mylist onChange="play()">
<option id="0" value="0">Music 1</option>
<option id="1" value="1">Music 2</option>
<option id="2" value="2">Music 3</option>
</select>

<audio id="music" ontimeupdate="updateTime()">
    <source src="music1.mp3">
</audio>
<div id="audioplayer">
    <p id="NP">Now Playing : music1.mp3</p>
    <p id="duration">00:00</p>
    <button id="prButton" class="prev" onclick="prev()">Prev</button>
    <button id="pButton" class="play" onclick="play()">Play</button>
    <button id="nButton" class="next" onclick="next()">Next</button>
</div>
<style>

```

```

#audioplayer{
    width:200;
    border: solid;
    padding:10;
}
#prButton,#pButton,#nButton{
    border-radius: 13px;
    font-family: Arial;
    color: #ffffff;
    font-size: 10px;
    background: #3498db;
    padding: 5px 10px 5px 10px;
    text-decoration: none;
}
</style>
<script>

var audioURL = document.getElementById('mylist');
var music = document.getElementById('music');
var pButton = document.getElementById('pButton');
var playlist=["music1.mp3","music2.mp3","music3.mp3"];
var nowplaying = document.getElementById('NP');
var cntr = 0;
//untuk formatting waktunya
function formatSecondsAsTime(secs, format) {
    var hr = Math.floor(secs / 3600);
    var min = Math.floor((secs - (hr * 3600))/60);
    var sec = Math.floor(secs - (hr * 3600) - (min * 60));

    if (min < 10){
        min = "0" + min;
    }
    if (sec < 10){
        sec = "0" + sec;
    }

    return min + ':' + sec;
}
//untuk display waktu sekarang
function updateTime(){
    duration.innerHTML = formatSecondsAsTime(music.currentTime);
}
function next(){
    if(cntr<playlist.length-1){//jika masih ada lagu dilist
        cntr++;//majukan counter
    }
    else{
        cntr=0;//pilih lagu pertama
    }
    audioURL.getElementsByTagName('option')[cntr].selected =
'selected';
    music.src= playlist[cntr];
    music.load();//mere-load lagu
    play();
}

```

```

function prev(){
    if(cntr!=0){//jika bukan lagu pertama
        cntr--;//mundurkan counter
    }
    else{
        cntr=playlist.length-1;//pilih lagu terakhir
    }
    audioURL.getElementsByTagName('option')[cntr].selected =
'selected';
    music.src= playlist[cntr];
    music.load();//mere-load lagu
    play();
}

function play() {
    if(audioURL.value!=cntr){//mengecek apakah listnya berubah apa
tidak
        cntr = audioURL.value;
        music.src= playlist[cntr];
        music.load();
    }
    nowplaying.innerHTML = "Now Playing : "+playlist[cntr];
    // play audio
    if (music.paused) {
        music.play();
        pButton.innerHTML = "pause";//ubah jadi pause
    } else { // pause audio
        music.pause();
        pButton.innerHTML = "play";//ubah jadi play
    }
}
</script>

```