Sistem Aplikasi Travel
dengan AngularJS & Codeigniter

Agung Julisman

CV. LOKOMEDIA

Aplikasi Travel dengan AngularJS & Codeigniter

Perpustakaan Nasional : Katalog Dalam Terbitan (KDT)
Penulis : Agung Julisman
Sistem Aplikasi Travel dengan Angular JS dan Codeigniter
- Cet. I. - Yogyakarta : Penerbit Lokomedia, 2014
164 halaman; 14 x 21 cm
ISBN : 978-602-14306-5-1

Penerbit Lokomedia,
Cetakan Pertama : Agustus 2014

Editor : Lukmanul Hakim
Cover : Subkhan Anshori
Layout : Lukmanul Hakim

Diterbitkan pertama kali oleh :
CV. LOKOMEDIA
Jl. Jambon, Perum. Pesona Alam Hijau 2 Kav. B-4, Kricak
Yogyakarta 55242.

email : redaksi@bukulokomedia.com
website : www.bukulokomedia.com

KATA PENGANTAR

Perkembangan teknologi pemrograman di sisi client mengalami perkembangan yang sangat cepat. Berbeda dari tahun-tahun sebelumnya, teknologi di sisi client cenderung datar dan perkembangan terjadi di sisi server, tapi tampaknya zaman sudah berbalik ke sisi client, hal ini dikarenakan banyaknya pengguna device mobile yang cenderung menggunakan teknologi di sisi client, sekarang proses bisnis bisa dikelola di sisi client. Teknologi di sisi server sekarang hanya digunakan sebagai service untuk mengelola data.

Buku ini mengajak para pembaca untuk memahami pemrograman di sisi client dengan menggunakan AngularJS dan CodeIgniter di sisi server sebagai web service.dengan studi kasus sistem aplikasi travel yang saat ini sangat diminati.

Puji syukur ke hadirat Allah SWT atas berkat dan rahmat-Nya, sehingga penulis dapat menyelesaikan buku ini, dan penulis menyadari masih banyak kekurangan dalam buku ini, kritik dan saran dapat langung disampaikan ke penulis, penulis menyediakan fasilitas free konsultasi mengenai buku ini melalui email.

Bandung, 3 Maret 2014

Agung Julisman

Web : http://julisman.com

Blog : http://blog.julisman.com

Email : agung.julisman@yahoo.com

Halaman ini Sengaja Dikosongkan

www.bukulokomedia.com

DAFTAR ISI

BAB I

Pemrograman Javascript

BAB 1
Pemrograman JavaScript

Dalam Bab 1 ini, kita akan membahas tentang dasar -dasar dari pemrograman di Javascript.

1.1 Apa Itu JavaScript?

Javascript pertama kali dikembangkan oleh Brendan Eich dari Netscape dibawah nama Mocha, yang nantinya namanya diganti menjadi LiveScript, dan akhirnya menjadi JavaScript. JavaScript merupakan **bahasa yang berorientasi pada objek atau di kenal dengan istilah OOP**.

1.2 Komentar dalam Kode JavaScript

Di dalam Javascript kita bisa menuliskan komentar, komentar ini sangat diperlukan **untuk menjelaskan sebuah logika yang kita buat** apalagi kalau kita bekerja dalam sebuah tim, agar kode kita bisa di pahami oleh programmer lain sebaiknya kita selalu memberikan komentar terhadap fungsi/logika yang kita buat.

Ada dua jenis komentar yang diperbolehkan yaitu dengan

- Tanda // dengan **batas akhir baris**.
- /* */ selama kode/tulisan berada dalam blok /* */ ini akan dianggap sebagai komentar oleh Javascript.

1.3 Tipe Data Primitif, Variabel, Operator

1.3.1 Tipe Data Primitif

Tipe data primitif adalah **tipe data dasar pada suatu bahasa pemrograman**, sedangkan tipe data sendiri adalah **jenis data yang ditangani oleh suatu bahasa pemrograman**.

Berikut tipe data primitif dalam Javascript:

- **Number**, contoh: 32 , 3.14
- **String**, contoh: “Halo Lokomedia”
- **Boolean**, contoh: true, false
- **null**, kata kunci khusus yang menunjukan nilai null.
- **undefined**, kata kunci yang menunjukkan nilai yang tidak terdefinisi

Javascript memiliki tipe data dinamis. Ini berarti bahwa **setiap variabel yang sama dapat digunakan sebagai jenis yang berbeda dan tidak perlu menetukan tipe data** terlebih dahulu pada variabel ketika kita membuat suatu variabel. Contoh:

```
var a;                  // sekarang tipe data variabel undefined
var a = 23              // sekarang tipe data variabel Number
var a = “Lokomedia”     // sekarang tipe data variabel String
```

1.3.1.1 Typeof

Jika kita **ingin mengetahui tipe data dari suatu variabel atau nilai**, kita dapat menggunakan operator khusus ***typeof***. Operator ini mengembalikan sebuah string yang mewakili tipe data. Nilai-nilai kembaliannya (return) bisa menjadi number, string, boolean, undefined, objek atau fungsi. Contoh:

```
var a =  1 // tipe data a di isi nilai 1
typeof(a)  // Number
var a = “JavaScript”  // tipe data a di isi nilai “JavaScript”
typeof(a)             // String
```

1.3.1.2 Konversi String ke Number

Dalam beberapa kasus, kita terkadang harus mengubah tipe data String ke Number.

```
var a = “1.2”
var b = “2”
var c = a + b
```

Maka variabel c akan berisi 1.22, dalam kasus ini karena tipe data a dan b adalah string, maka ketika dilakukan operator penambahan hasilnya akan menggabungkan kedua buah string tersebut.

Untuk mengkonversinya kita menggunakan fungsi yang telah disediakan oleh Javascript, yaitu `parseInt()`dan `parseFloat()`. Jika kita menggunakan fungsi `parseInt()` maka data yang akan di konversi akan menjadi bilangan bulat

sedangkan jika kita menggunakan fungsi **parseFloat()** data yang di konversi akan menjadi bilangan decimal.

```
var c = parseInt(a) + parseInt(b)     // maka nilai c = 3
var c = parseFloat(a) + parseFloat(b) // maka nilai c = 3.2
```

1.3.1.3 Karakter Spesial didalam String

Beberapa karakter memerlukan notasi ***backslash*** di Javascript. Dalam beberapa kasus terkadang kita harus menambahkan karakter spesial. Berikut contoh beberapa penambahan karakter spesial di Javascript:

- Menambahkan karakter **\t**

  `Nama :\tnizam\t\tUmur :\t19` // Hasilnya → Nama : nizam Umur : 19

- Menambah tanda kutip dua (“string”)

  `Sistem Informasi \”SMA 19\”` // Hasilnya → Sistem Informasi ”SMA 19”

- Menambah *backslash* (\)

  `No hp\\ No tlp` // Hasilnya → No hp \ No tlp

- Pindah Baris \n

  ```
  Nama Depan : Nizam\n Nama Belakang : Ibrahim
  ```

  Hasilnya:

  // Nama Depan : Nizam

  // Nama Belakang : Ibrahim

1.3.2 Variabel

Javascript menggunakan variabel untuk menyimpan, mengelola data dan bukan merupakan data sebenarnya. Javascript merupakan bahasa yang *case sensitive* (membedakan huruf besar dengan huruf kecil, A tidak sama dengan a)

```
var tinggi = 7
var Tinggi = 9  // tinggi == Tinggi (false)
```

Keterangan:

var (variabel keyword), **tinggi** (variabel nama), **=** (*Assignment Operator*), **7** (nilai dari variabel).

Ada beberapa aturan dalam penamaan variabel, contohnya:

Contoh	Keterangan
var nama depan	×, **tidak boleh** ada spasi.
var 9nama, var @angka, var _+angka	×, **tidak boleh** diawali angka, simbol, operator.
var nama, var Nama	☑, **boleh** diawali huruf besar dan huruf kecil.
var nama_depan	☑, **boleh** menggunakan garis bawah.

1.3.3 Operator

Operator mengambil satu atau lebih dari nilai, melakukan operasi, dan mengembalikan nilai. Sintaksnya:

```
operand1 operator operand2
```

Contoh: 1 + 9 atau x * y

Berikut jenis-jenis operator dalam JavaScript:

1.3.3.1 Bitwise Operator

Operator bitwise memperlakukan mereka sebagai operan satu set 32 bit (nol dan satu), bukan sebagai desimal, heksdesimal, atau angka oktal. Sebagai contoh, angka desimal sembilan memiliki representasi biner 1001. Operator bitwise melakukan operasi mereka pada representasi biner seperti itu, tapi mereka kembalikan nilai-nilai numerik Javascript standar. Berikut tabel Operator Bitwise:

Operator	Contoh
Bitwise AND	a & b
Bitwise OR	a | b
Bitwise XOR	a ^ b
Bitwise NOT	~ a

1.3.3.2 Comparison Operator

Sebuah operator perbandingan membandingkan operan dan mengembalikan nilai berdasarkan apakah perbandingan yang kita bandingkan itu benar. Operan dapat berupa number, string, logic, atau nilai-nilai objek.

Berikut tabel Operator Perbandingan:

Operator	Keterangan
==	Mengembalikan nilai **true** jika operan sama.
===	Mengembalikan nilai **true** jika operan sama dan tipe data yang sama.
!=	Mengembalikan nilai **true** jika operan tidak sama.
!===	Mengembalikan nilai **true** jika operan tidak sama dan tipe data tidak sama.
>	Mengembalikan nilai **true** jika operan kiri lebih besar dari operan kanan.
>=	Mengembalikan nilai **true** jika operan kiri lebih besar atau sama dengan operan kanan.
<	Mengembalikan nilai **true** jika operan kanan lebih besar dari operan kiri.
<=	Mengembalikan nilai **true** jika operan kanan lebih besar atau sama dengan operan kiri.

Setelah kita lihat tabel operator perbandingan, muncul pertanyaan apa sih bedanya operator == dengan ===? Seperti yang sudah dijelaskan perbedaannya jika operator == tidak membandingkan tipe data dalam operannya sedangkan operator === **membandingkan nilai dan tipe datanya**. Contoh:

```
var a = "2"
var b = 2
a == b      // true
a === b     // false

var a = null
var b;      // nilai b sekarang undefined
a == b      // true
a === b     // false
```

```
var a = “”  // string kosong
var b = 0
a == b      // true
a === b     // false

var a = “”  // string kosong
var b = false
a == b      // true
a === b     // false
```

Sekarang sudah cukup jelaskan perbedaan antara operator = = dengan = = =, begitu juga dengan operator ! = dengan ! = = =.

1.3.3.3 Assignment Operator

Shorthand operator	Keterangan
x += y	x = x + y
x -= y	x = x – y
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y
x <<= y	x = x << y
x >>= y	x = x >> y
x >>>= y	x = x >>> y
x &= y	x = x & y
x ^= y	x = x ^ y
x |= y	x = x | y

1.3.3.4 Arithmetic Operator

Arithmetic operator mengambil nilai numerik (baik literal maupun variabel) sebagai operan mereka dan mengembalikan nilai numerik tunggal. Operator ini

bekerja seperti yang mereka lakukan dalam kebanyakan bahasapemrograman lain, dimana jika digunakan dengan angka floating point (khususnya, perhatikan divisi yang dengan nol menghasilkan Infinity). Contoh:

```
1 / 2                      // maka akan mengenghasilkan 0.5
(1 / 2 == 1.0 / 2.0)       // maka akan mengembalikan true
```

Berikut tabel Operator Aritmatika:

Operator	Keterangan
+	Menambahkan operan.
-	Mengurangi operan.
*	Mengalikan operan.
/	Membagi operan.
%	Mengembalikan sisa integer dari hasil membagi dua operan.
++	Menambahkan salah satu operannya.
--	Mengurangi salah satu operannya.

1.3.3.5 String Operator

Selain operator perbandingan, yang dapat digunakan pada nilai string dengan operator (+) *concatenation operator* adalah menggabungkan dua buah string, contoh:

```
var a = "saya"
var b = "budi"
var c = a + c  // maka nilai dari variabel c adalah "sayabudi"
```

1.3.3.6 Logical Operator

Operator logika biasanya digunakan dengan Boolean (logic), ketika mereka mengembalikan nilai Boolean. Namun, && dan | | operator benar-benar mengembalikan nilai dari salah satu operan yang ditentukan, jadi jika operator ini digunakan dengan nilai-nilai non-Boolean, mereka dapat mengembalikan nilai non-Boolean.

Berikut tabel Operator Logika

Operator	Keterangan
&&	(Logical AND) Pengembalian expr1 jika dapat dikonversi ke **false**, jika tidak, kembali ke expr2. Dengan demikian, bila digunakan dengan nilai-nilai Boolean, && mengembalikan nilai **true** jika kedua operan adalah **true**, jika tidak, kembali **false**.
||	(Logical OR) Pengembalian expr1 jika dapat dikonversi menjadi **true**, jika tidak, kembali ke expr2. Dengan demikian, bila digunakan dengan nilai-nilai Boolean,|| mengembalikan nilai true jika salah satu operan true, jika keduannya salah, kembali false.
!	(Logical NOT) Pengembalian false jikaoperan tunggal dapat dikonversi menjadi true, jika tidak, mengembalikan nilai true.

Contoh operator && (logical AND)

```
var a = true&&true          // mengembalikan true
var a = true&&false         // mengembalikan false
var a = false&&false        // mengembalikan false
```

Contoh operator || (logical OR)

```
var b = true || true        // mengembalikan true
var a = true || false       // mengembalikan true
var a = false || false      // mengembalikan false
```

Contoh operator ! (logical NOT)

```
var c =!true                // mengembalikan false
var c =!false               // mengembalikan true
```

1.3.3.7 Special Operator

Dalam Javascript banyak terdapat special operator, penulis akan memberikan salah satu contohnya yaitu **conditional operator**, karena ini sering di gunakan.

```
Condition ? val1 : val2 // cara penggunaan

var status = (age > 17) ? "dewasa" : "anak - anak";
```

Keterangan:

(age > 17) adalah sebuah kondisi dimana jika kondisi ini **true**maka variabel status

akan disi dengan var1 yaitu **dewasa**, sebaliknya jika kondisi bernilai false maka variabel status akan di isi dengan var2 yaitu **anak-anak**.

1.4 Kondisi dan Perulangan

Sangat sering ketika kita menulis kode program, kita ingin melakukan tindakan yang berbeda untuk keputusan yang berbeda. Untuk kasus yang seperti ini kita bisa menggunakan pernyataan kondisi atau yang lebih dikenal dengan nama **Conditional Statements**.

Berikut conditional statements yang ada di Javascript.

- **if statements**, digunakan untuk mengeksekusi beberapa kode dan jika kondisi adalah **true**.

```
if( nilai < 50){
    index = "D";
}
```

- **if … else statements**, digunakan untuk mengeksekusi beberapa kode jika kondisi **true** dan kode lain jika kondisi **false**.

```
if (nilai < 50){
    index = "D";
} else{
    index = "A";
}
```

- **if … else if …else statements**, digunakan untuk memilih salah satu dari banyak blok kode yang akan di eksekusi.

```
if (nilai < 50){
    index = "D";
}else if(nilai >= 50 && nilai < 70){
    index ="C";
}else if(nilai >=70 && nilai < 80){
    index = "B";
}else{
    index ="A";
}
```

- **switch statements**, digunakan untuk memilih salah satu dari banyak bok kode yang akan di eksekusi.

```
switch(nilai){
case "A":
     x = "Sangat baik";
```

```
        break;
    case "B":
        x = "Baik";
        break;
    case "C":
        x = "Cukup";
        break;
    case "D":
        x = "Kurang";
        break;
    }
```

Selanjutnya, Javascript juga mempunyai logika perulangan. Dalam Javascript ada empat jenis perulangan, yaitu:

- **while**, jenis yang paling sederhana.

```
var i = 0;
while(i < 10){
    console.log(i);
    i++;
}
```

Keterangan: Jika dilihat dalam console akan menghasilkan **0…9**. Pada perulangan while akan terus melakukan perulangan selama kondisi **true**.

- **do - while**, variasi dari perulangan while.

```
var i = 11;
do {
i++
console.log(i);
}
while(i < 10)
```

Keterangan: Jika dilihat dalam console akan menghasilkan **12**. Pada perulangan do - while, statement **do** di-ikuti oleh sebuah blok kode dan **kondisi setelah blok**. Ini berarti bahwa blok kode akan selalu dijalankan, setidaknya sekali, sebelum kondisi di evaluasi.

- **for**, perulangan ini paling sering digunakan ketika kita akan melakukan perulangan, penulis juga sering menggunakan for untuk melakukan perulangan karena hanya **membutuhkan sedikit sintak**.

```
for( var i = 0;i<10;i+){
    console.log(i);
}
```

- **for - in**, biasannya digunakan untuk melihat elemen array atau objek, tujuan

perulangan ini hanya untuk melihat informasi saja.

```
var a = ["a","b","c"];     // membuat array
var x = "";                // membuat string kosong

for(i in a){       // lakukan perulangan terhadap array a
 x += "index : " +  i + ", mempunyai nilai : " + a[i] + "\n";
 // var x akan menampung hasil pengulangan
}
```

Keterangan:

Hasil perulangannya sebagai berikut:

index : 0 mempunyai nilai a

index : 1 mempunyai nilai b

index : 2 mempunyai nilai c

1.5 Array

Sekarang kitaakan membahas tentang array di JavaScript. Apa sih itu array? **Array adalah variabel khusus yang dapat menyimpan lebih dari satu nilai pada satu waktu**. Contoh jika kita memiliki daftar teman yang akan kita tampung ke dalam satu variabel bernama teman, mungkin kita akan menyimpannya seperti ini:

```
var teman1 = "agung"
var teman2 = "nicky"
var teman3 = "adit"
var teman4 = "jaka"
```

Namun bagaimana jika kita akan melakukan perulangan (loop) dengan variabel teman dan menemukan dengan spesifikasi tertentu dan jumlah teman kita berjumlah lebih dari 100? Nah solusinya adalah dengan array, sebuah array dapat menyimpan banyak nilai dan kita dapat mengaksess nilai dengan mengacu pada nomor index sebuah array.

Array dapat dibuat dengan tiga cara, yaitu:

- Regular

```
var teman = new array();
teman[0] = "agung";
teman[1] = "nicky";
teman[2] = "adit";
teman[3] = "jaka";
```

- Condensed

```
var teman = new array("agung","nicky","adit","jaka");
```

- Literal

```
var teman = ["agung","nicky","adit","jaka"];
```

Untuk mengakses elemen array, kita harus menentukan indeks dari elemen di dalam tanda kurung siku. Jadi [0] memberikan elemen pertama dari array. Ingat index dari elemen array selalu berawal dari **0** bukan **1**. Contohnya kita akan memanggil teman kita yang bernama nicky, kodenya:

```
teman[1] // memanggil index ke 1 dari array teman yaitu nicky
```

Selanjutnya, untuk menambah, mengedit dan menghapus data yang ditampung dalam array, kita bisa menggunakan **index sebagai key**, sedangkan untuk menghapus elemen pada array kita bisa menggunakan perintah **delete**. Contoh:

```
// tambah
var teman[4] ="ruly"  // array teman mempunyai index ke 4
                      // dengan nilai "ruly"

// edit
var teman[0] = "julisman" // sekarang  array teman yang ke 0
                          // diganti nilainya menjadi julisman
                          // yang tadinya agung.

// hapus
delete teman[2]
// sekarang array teman yang ke 2 menjadi undefined
// ["julisman", "nicky", undefined, "adit", "jaka", "ruly"]
```

1.6 Fungsi

Fungsi adalah **satu blok kode yang melakukan tugas atau menghitung suatu nilai**. Dengan menggunakan fungsi kita akan **mengurangi beberapa kode yang di pakai secara berulang**.

Misalnya kita akan membuat fungsi faktorial, sebelum membuat fungsi faktorial kita harus mengetahui terlebih dahulu aturan untuk menghitung suatu faktorial ke n suatu bilangan, **faktorial** dari bilangan asli ke *n* adalah hasil perkalian antara bilangan bulat positif yang kurang dari atau sama dengan *n*.

```
function faktorial(n){
if (n < 2 )
     return 1;
else
```

```
        return(n * faktorial(n -1));
    }
```

Keterangan:

- **`function faktorial(n)`**, membuat fungsi dengan nama faktorial dengan parameter n.
- **`if (n < 2)`**, disini kita mempunyai kondisi, jika n kurang dari 2, maka fungsi ini akan mengembalikan nilai 1.
- **`else return(n * factorial(n -1));`** jika kondisi lebih dari atau sama dengan 2, maka fungsi ini akan melakukan kondisi ini n * fungsi dirinya sendiri di kurangi 1.

Untuk menggunakan fungsi yang kita buat, kita cukup memanggil nama fungsinya saja, **faktorial(3)**, ini akan menghasilkan **6**.

Dengan membuat fungsi, ketika kita membutuhkan perhitungan factorial kita cukup memanggil fungsi yang telah kita buat tadi saja **faktorial(n)**, jadi dengan membuat fungsi kita tidak akan membuat script yang sama secara berulang.

1.7 Parameter

Ketika kita membuat suatu fungsi, kita dapat menentukan parameter apa saja yang akan kita masukkan ke dalam sebuah fungsi, ini menandakan bahwa fungsi yang kita buat memiliki parameter yang statis, artinya ketika kita memilki sebuah fungsi kita harus memasukkan parameter sesuai dengan parameter di fungsi yang kita buat, contoh:

```
function tambah(a,b){
     return a + b;
}
```

Keterangan:

function tambah(a,b), membuat fungsi dengan nama tambah dengan dua parameter, yaitu a dan b, ketika kita memanggil fungsi tambah kita harus memasukan parameter sebanyak parameter yang telah kita set.

Apabila kita panggil **tambah(2)** akan error, karena kita tidak mendefinisikan parameter b. Atau ketika kita panggil **tambah(2,3,4)**, kode tidak akan error tapi hasilnya tidak akan sesuai karena yang akan di tambahkan hanya **2** dan **3** saja.

Sekarang muncul pertanyaan bagaimana kita membuat fungsi tanpa kita

mendefinisikan parameter dimana ketika kita memanggil fungsi **parametenya dinamis**? Pada JavaScript sudah ada solusinya menggunakan **arguments**, arguments array akan dibuat secara otomatis didalam setiap fungsi yang hanya mengembalikan parameter apa saja yang dikirim ke sebuah fungsi, contoh:

```
function tambah(){
hasil = 0;
var panjang_array_arguments = arguments.length;

for(var i = 0; i < panjang_array_arguments; i++){
     hasil += arguments[i];
}
return hasil;
}
```

Keterangan:

- **`hasil = 0;`** membuat variabel **hasil** dengan tipe data number.
- **`var panjang_array_arguments = arguments.length;`** membuat variabel **panjang_array_arguments** dengan jumlah array yang ada di arguments dengan bantuan key **length**.
- **`for(var i = 0; i < panjang_array_arguments; i++)`**, lakukan perulangan sebanyak jumlah data array yang berada di arguments.
- **`hasil += arguments[i]`**, masih inget sama Assigment Operator +=? Kalau udah lupa baca lagi ya he he he..

Untuk menggunakannya, misalnya **tambah(2,3,4)**, dimana didalam fungsi tambah, arguments akan membentuk array secara otomatis sebagai berikut:

arguments[0] = 2;

arguments[1] = 3;

arguments[2] = 4;

Maka ketika kita memanggil fugsi **tambah**(2,3,4) akan mengembalikan nilai **9**.

1.8 Object

Penggunaan object dalam pemrograman sangat penting, **sebuah object adalah representasi dari sesuatu** dan representasi ini dinyatakan dengan bantuan bahasa pemrograman hal ini bisa benda, makhluk hidup, atau apapun.

Sebagai contoh kita akan menggambarkan object mobil, kita bisa melihat

bahwa mobil memiliki karakteristik tertentu misalnya warna, merek, jenis mobil dan masih banyak lagi dan kemampuan mobil ini bisa menghidupkan lampu, berjalan, menyalakan AC dan masih banyak lagi. Di dalam object programming, **karakteristik merupakan properties** dan **kemampuannya disebut methods**.

Di Javascript sendiri, desain dari object dibuat sangat sederhana, dimana **sebuah object adalah kumpulan-kumpulan properties**, sebuah nilai dari properties bisa menjadi fungsi, dalam hal ini properties bisa disebut sebagai methods. Contoh kita akan membuat sebuah object sebuah mobil.

Contoh 1:

```
// definisikan tipe data mobil sebagai objek
var mobil = new Object();
mobil.warna = "Merah";      // set properties warna
mobil.merek = "Honda"       // set properties merek
mobil.tahun = 2013          // set properties tahun
```

Contoh 2:

```
var mobil = {}; // definisikan tipe data mobil sebagai objek
mobil = {
            'warna' : 'Merah',     // set properties warna
            'merek' : 'Honda',     // set properties merek
            'tahun'  : 2013        // set properties tahun
}
```