

Pengantar JavaScript

Andry Alamsyah
alamsyah@tiscali.fr

Semenitpun Internet tidak akan berhenti berkembang , demikian juga dengan bahasa pemrograman berbasis web yang akan selalu berkembang, termasuk juga JavaScript, yang merupakan bahasa pengenalan awal sebelum kita benar benar serius terjun dan berkecimpung di dalam pengembangan aplikasi web. Pada artikel Pengantar JavaScript ini kita akan belajar dasar dasar pengenalan struktur bahasa dan cara kerja JavaScript dalam suatu dokumen HTML

1. Pengenalan Javascript

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

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

Javascript bergantung kepada browser(navigator) yang memanggil halaman web yang berisi skrip skrip dari Javascript dan tentu saja terselip di dalam dokumen HTML. Javascript juga tidak memerlukan kompilator atau penterjemah khusus untuk menjalankannya (pada kenyataannya kompilator Javascript sendiri sudah termasuk di dalam browser tersebut). Lain halnya dengan bahasa "*Java*" (dengan mana JavaScript selalu di banding bandingkan) yang memerlukan kompilator khusus untuk menterjemahkannya di sisi user/klien.

Tabel daftar navigator dan versi dari Javascript :

Versi Javascript	Browser
Javascript 1.2	Netscape Navigator 4.0/4.05, Internet Explorer 4.0
Javascript 1.1	Netscape Navigator 3.0
Javascript 1.0	Netscape Navigator 2.0, Internet Explorer 3.0
Javascript 1.5	Netscape Navigator 6.0
Javascript 1.3	Netscape Navigator 4.06, Internet Explorer 5.0
Javascript 1.4	Netscape Navigator 6.0, Internet Explorer 5.5

Janganlah anda bingung dengan perbandingan antara Javascript dan Java, karena berdasarkan pengalaman yang saya peroleh baik dari milist milist maupun forum forum banyak yang belum bisa menemukan perbedaan jelas antara keduanya, oleh karena itu saya akan jelaskan berikut ini.

Javascript merupakan suatu bahasa yang perkembangannya lambat di bandingkan dengan Java yang berkembang sangat cepat. Di Javascript kita tidak mungkin menyembunyikan kode skrip yang kita tulis, kode langsung di tulis di dalam dokumen HTML dan sangat mudah terlihat, sedangkan di Java, kode sudah berbentuk setengah terkompilasi (dalam bentuk applet) dan tidak mungkin terlihat dari dalam dokumen HTML, satu mesin virtual di sisi user yang bertanggung jawab untuk menterjemahkan program di dalam applet tersebut setiap kali halaman HTML yang memuat applet tersebut dipanggil oleh browser. Dibandingkan dengan applet java yang cukup lambat dibuka oleh browser, bisa kita katakan bahwa Javascript cukup cepat di panggil(di load) oleh navigator.

JavaScript sendiri merupakan bahasa yang mudah dipahami, dalam artian diperlukan skill novice atau dasar untuk mengerti bahasa ini, jika anda sudah terbiasa dan mengenal konsep bahasa pemrograman visual, maupun Java ataupun C, akan sangat mudah untuk memahami konsep Javascript.

Berikut ini satu tabel yang berisi beberapa perbandingan mendasar antara Java dan JavaScript :

JavaScript	Java
Bahasa yang diinterpretasikan langsung oleh browser	Bahasa yang setengah terkompilasi dan memerlukan Java Virtual Machine untuk menterjemahkannya
Kode terintegrasi dengan HTML	Kode(applet) terpisah dari dokumen HTML, dipanggil pada saat membuka dokumen HTML
Bahasa dengan karakteristik yang terbatas	Bahasa dengan karakteristik yang luas (pendeklarasian jenis variabel)
Hubungan dinamis, referensi dari obyek diverifikasi pada saat loading.	Hubungan statis, obyek harus ada pada saat program di loading (di kompilasi)
Kode program bisa di akses	Kode program tersembunyi

JavaScript adalah bahasa yang “*case sensitive*” artinya membedakan penamaan variabel dan fungsi yang menggunakan huruf besar dan huruf kecil, contoh variabel atau fungsi dengan nama *TEST* berbeda dengan variabel dengan nama *test*. Dan yang terakhir seperti bahasa Java ataupun C, setiap instruksi diakhiri dengan karakter titik koma (;).

1.1 Bentuk skrip dari Javascript

Skrip dari JavaScript terletak di dalam dokumen HTML. Kode tersebut tidak akan terlihat dari dalam jendela navigator anda, karena diantara tag (kalau anda mengerti HTML pasti tahu dengan istilah ini) tertentu yang memerintahkan navigator untuk memperlakukan bahwa skrip tersebut adalah skrip dari JavaScript. Contoh dari skrip yang menunjukkan bahwa skrip tersebut adalah skrip dari JavaScript adalah sebagai berikut :

```
<SCRIPT language="Javascript">  
letakkan script anda disini  
</SCRIPT>
```

1.2 Memberikan komentar (dan membuat skrip tidak tereksekusi)

Sering kali pada navigator versi lama, sebelum adanya JavaScript, tidak mengenal tag tersebut dan akan melewatkannya untuk di baca. Contoh kode diatas tidak akan terlihat di navigator kita, akan tetapi dia akan menampilkan jendela peringatan (berupa kotak dialog) karena skrip tersebut tidak lengkap dan akan merusak dokumen HTML yang sudah kita buat dengan bagusnya. Untuk itu maka kita tambahkan tag komentar agar supaya skripnya tidak dibaca sebagai skrip, akan tetapi di baca sebagai komentar dan tidak akan dieksekusi sebagai program. Contohnya adalah sebagai berikut :

```
<SCRIPT language="Javascript">  
<!--  
letakkan script anda disini  
// -->  
</SCRIPT>
```

Seperti dalam banyak bahasa pemrograman lainnya, sangat dianjurkan untuk menambahkan komentar komentar di dalam skrip atau kode program yang kita bikin. kegunaannya antara lain adalah :

- Mengingatkan kita akan bagian bagian khusus di dalam skrip tersebut, jika kita ingin merubah sesuatu di dalamnya, mungkin beberapa bulan kemudian dan kita sudah lupa dengan detail dan alur dari skrip tersebut.
- Membuat orang yang tidak mengerti skrip anda, menjadi mengerti dengan petunjuk petunjuk yang anda bikin melalui komentar komentar, kecuali anda tidak mau mensharing program yang anda miliki ..

Untuk menulis komentar di JavaScript, kita bisa menggunakan cara yang sama dengan aturan yang ada di bahasa C/C++ ataupun Java.

- Untuk menulis komentar dalam satu baris kita gunakan karakter dobel slash.

```
// semua karakter di belakang // tidak akan di eksekusi
```

- Untuk menulis komentar yang terdiri dari beberapa baris kita gunakan karakter /* dan */

```
/* Semua baris antara 2 tanda tersebut  
tidak akan di eksekusi oleh  
kompilator */
```



Perhatikan perbedaan tag komentar untuk bahasa HTML (diantaranya untuk menyembunyikan skrip dari beberapa browser versi lama) dan tag komentar JavaScript (untuk keperluan dokumentasi skrip).

1.3 Contoh program JavaScript

Pada contoh berikut ini adalah contoh skrip JavaScript didalam suatu dokumen HTML, disini kita akan membuat satu program untuk menampilkan satu kotak dialog (dijelaskan lebih lanjut di bab 10) pada saat kita membuka dokumen HTML

```
<HTML>  
<HEAD>  
<TITLE>Contoh Program Javascript</TITLE>
```

```
</HEAD>  
<BODY>  
  
<SCRIPT language="Javascript">  
<!--  
alert("Hallo !");  
// -->  
</SCRIPT>  
  
</BODY>  
</HTML>
```

1.4 Meletakkan JavaScript dalam dokumen HTML

Ada beberapa cara untuk meletakkan kode JavaScript di dalam dokumen/halaman HTML

- **Menggunakan tag <SCRIPT>**

Dengan menggunakan cara ini, pada saat mengakses satu halaman HTML kita harus menunggu sampai proses pemanggilan halaman itu selesai sepenuhnya, sebelum kita menjalankan program JavaScript tersebut. Proses eksekusi kode HTML untuk menampilkan satu halaman dilakukan dari atas ke bawah, semakin banyak user yang mengakses halaman ini (dan seringnya gak sabar ...) dapat mengganggu proses pemanggilan tersebut. Pada kasus dimana pemanggilan suatu fungsi JavaScript terjadi sebelum proses pemanggilan kode JavaScript selesai dilakukan oleh navigator, maka akan timbul pesan error.

Oleh karena itu untuk menghindari kejadian diatas, maka pada umumnya kita meletakkan tag <SCRIPT> diantara bagian kepala dari dokumen HTML, yaitu bagian antara tag <HEAD> dan </HEAD>. Pemanggilan fungsi JavaScript (atau disebut juga *event*) diletakkan di bagian badan dokumen HTML atau bisa kita sebut diantara tag <BODY> dan </BODY>.

Keterangan tambahan di dalam tag <SCRIPT> menunjukkan jenis bahasa yang digunakan dan versinya, contohnya "JavaScript", "JavaScript1.1", "JavaScript1.2" untuk bahasa JavaScript atau bahasa lainnya, contohnya "VBScript".

Jika kita ingin menggunakan beberapa versi JavaScript di dalam satu halaman HTML (untuk menyesuaikan dengan kompatibilitas navigator) , maka kita hanya perlu meletakkan kode kode JavaScript tersebut (berdasarkan versinya) kedalam beberapa tag <SCRIPT> dengan mencantumkan versi JavaScriptnya.

- **Menggunakan file ekstern**

Cara berikutnya adalah menuliskan kode program JavaScript dalam suatu file teks dan kemudian file teks yang berisi kode JavaScript di panggil dari dalam dokumen HTML (khusus Netscape mulai versi 3 keatas). Kode yang kita sisipkan kedalam dokumen HTML adalah sebagai berikut :

```
<SCRIPT LANGUAGE="Javascript" SRC="url/file.js"> </SCRIPT>
```

dimana url/file.js adalah lokasi dan nama file yang berisi kode JavaScript, jika perintah tambahan SRC tidak disertakan maka tag Script akan mencari kode yang terletak di dalam tag Script.

- Melalui event tertentu

Event adalah sebutan dari satu action yang dilakukan oleh user, contohnya seperti klik tombol mouse, pembahasan lebih lanjut ada di bab 5. Kodenya dapat di tulis sebagai berikut :

```
<tag eventHandler="kode Javascript yang akan dimasukkan">
```

dimana `eventHandler` adalah nama dari event tersebut.

2. Obyek

2.1 Tentang Obyek

Untuk selanjutnya mari kita masuk kebagian berikutnya, yaitu bagian yang berbicara tentang obyek, tujuan dari bagian ini tidaklah muluk muluk sampai ke pengetahuan tentang *pemrograman orientasi obyek* akan tetapi sekedar pengenalan tentang obyek, dan memberikan ide kepada anda apakah yang disebut obyek itu ? ,yang merupakan konsep penting di dalam pembuatan kode JavaScript.

JavaScript memperlakukan elemen elemen yang tampil di jendela navigator kita sebagai suatu obyek , yang artinya adalah elemen yang :

- Diklasifikasikan berdasarkan hirarki/tingkatan khusus sehingga kita bisa mengetahui dimana letak/lokasi obyek itu sebenarnya.
- Dimana kita mengasosiasikan dengan kondisi atau sifat sifat khusus (properti)

Pengertian diatas mungkin sedikit membingungkan bagi anda, tapi marilah kita lihat contoh berikut ini untuk memperjelas. Misal kita bayangkan ada satu pohon yang terletak di dalam satu kebun, dimana pohon itu mempunyai banyak dahan, dan di salah satu dahannya terdapat sarang burung. Mari kita lihat hirarkinya sebagai berikut.

- Kebun
 - Pohon
 - Dahan
 - Daun
 - Sarang Burung
 - Panjang = 20
 - Warna = kuning
 - Tinggi = 4
 - Batang
 - Akar
 - Kolam
 - Sangkar Ternak
 - Ayam
 - Bebek
 - Gudang
 - Pintu

- Atap
 - Sarang Burung
 - Panjang =15
 - Warna = coklat
 - Tinggi = 6
- Jendela

Sarang burung yang berada di atas pohon dapat di tuliskan sebagai berikut :

`Kebun.Pohon.Dahan.Sarang Burung`

Sedangkan Sarang burung yang terletak diatas atap gudang, kita tuliskan sebagai berikut :

`Kebun.Gudang.Atap.Sarang Burung`

Bayangkan sekarang kita ingin mengecat atau mengganti warna sarang burung yang terletak di atas pohon, maka kita cukup mengetik perintah berikut ini

`Kebun.Pohon.Dahan.Sarang Burung.warna=hijau`

Nah begitulah ilustrasi tentang bagaimana kita memperlakukan obyek di dalam JavaScript, perbedaannya adalah bukan kebun sebagai bentuk obyek kita, akan tetapi navigator kitalah sebagai obyek utamanya.

2.2 Obyek dari navigator (browser)

JavaScript membagi satu halaman Navigator dalam berbagai obyek obyek, dengan tujuan untuk memudahkan kita untuk mengakses salah satu dari mereka dan memanipulasinya dengan cara merubah sifat/kondisi (properti) mereka.

Kita mulai dari obyek yang paling besar diantara semuanya (yang berisi semua obyek lainnya), kemudian kita turun berdasarkan tingkatan atau hirarkinya sampai kepada obyek yang kita inginkan.

- Obyek paling besar adalah obyek jendela (**window**) dari navigator.
- Di dalam obyek jendela, ada satu obyek yang ditampilkan dalam bentuk sebuah halaman, kita sebut obyek dokumen atau **document**
- Halaman itu berisi banyak obyek seperti, formula, text, image dan lain lainnya..

Untuk mengakses satu obyek, kita harus mengakses terlebih dahulu obyek yang paling besar(dalam hal ini obyek **window**) . ambil sebagai contoh satu obyek (yang bernama *checkbox*) dan satu *textfield* berikut ini



- Form untuk untuk contoh ini kita beri nama *form1* tujuannya adalah utuk membedakan dengan form form lainnya di dalam satu dokumen, dan dituliskan dalam kode berikut ini :
`window.document.forms["form1"]`
- Tombol checkbox, dan kita beri nama *checkbox1* dituliskan dalam bentuk berikut ini :
`window.document.forms["form1"].checkbox1`
- TextField disini kita beri nama *tetxfield1* dan dituliskan dalam bentuk berikut ini :
`window.document.forms["form1"].textfield1`

Tombol checkbox mempunyai nilai kondisi(properti) *checked* , yang akan memberikan nilai 1 jika tombol itu di pilih (checked), dan nilai 0 pada kasus sebaliknya.

Sedangkan kode dari form *form1* di dalam kode HTML adalah sebagai berikut :

```
<form name="form1">  
<br><input type="checkbox" name="check_box" onClick="ModifField();return true;">  
<br><input type='TEXT' name='text_field' value=test javascript' size='24'></form>
```

dan fungsi JavaScript yang berhubungan dengan *checkbox* tersebut adalah ..

```
<script language="Javascript">  
<!--  
function ModifField()  
{  
    if (document.forms["form1"].check_box.checked)  
    {document.forms["form1"].text_field.value="checkbox dipilih"}  
    else  
    {document.forms["form1"].text_field.value="checkbox tidak dipilih"}  
}  
// -->  
</script>
```

2.3 Obyek standart JavaScript

Selain obyek navigator di subbab 2.2, kita juga mengenal obyek standart dari JavaScript. Obyek obyek ini distandarisasikan oleh asosiasi ECMA (European Computer Manufacturer Association). Berikut ini adalah daftar obyek standart JavaScript

Nama Obyek	Keterangan
Array	obyek array memungkinkan kita untuk membuat tabel. dia mempunyai berbagai metoda untuk menambahkan, menghapus, atau juga mengambil elemen elemen dari suatu tabel dan juga mengurutkan elemen elemen tersebut.
Boolean	obyek boolean memungkinkan kita untuk membuat nilai boolean, dalam artian elemen yang mempunyai dua kondisi : <i>benar</i> atau <i>salah</i>
Date	obyek data memungkinkan kita untuk membuat dan memanipulasi tanggal dan juga durasi waktu.
Function	obyek function memungkinkan kita untuk membuat fungsi yang sesuai dengan kebutuhan (personalized)
Math	obyek math memungkinkan kita untuk memanipulasi fungsi fungsi matematika, seperti contohnya fungsi trigonometri
Number	obyek number memungkinkan kita untuk membuat operasi operasi dasar terhadap bilangan
RegExp	obyek regexp memungkinkan kita untuk membuat satu ekspresi umum(regular expression). ekspresi ini berguna untuk melakukan operasi operasi yang lebih canggih terhadap variabel jenis <i>String</i>
String	obyek string menyediakan banyak jenis metoda yang memungkinkan kita untuk memanipulasi variabel jenis <i>String</i>

2.3.1 obyek Array

obyek array adalah satu obyek yang memungkinkan kita untuk membuat dan memanipulasi tabel, berikut ini adalah sintaks untuk membuat tabel :

```
var x = new Array(elemen1[, elemen2, ...]);
```

jika tidak ada elemen yang disebutkan dalam parameter, tabel itu akan menjadi tabel kosong pada saat pembuatannya, sebaliknya jika elemen diisi, maka isi tabel akan di inisialisasi oleh nilai dari elemen tersebut. Sebagai tambahan obyek array mempunyai dua karakteristik properti yaitu properti *input* dan *length*.

Berikut ini adalah tabel daftar beberapa metoda standart dari obyek array

Metoda	Keterangan
concat (tab1,tab2[,tab3, ...])	memungkinkan kita untuk menambahkan (concatenate) banyak tabel, dalam artian membuat satu tabel dari beberapa tabel yang berbeda yang dilewatkan sebagai parameter metoda.
join (tabel) atau Table.join()	mengirimkan satu variabel string yang berisi semua elemen dari tabel
pop (tabel) atau Table.pop()	menghapus elemen terakhir dari tabel.
Table.push (nilai1[,nilai2,])	menambahkan satu atau beberapa elemen ke tabel
Table.reverse()	membalikkan (inverse) urutan elemen di tabel
Table.shift()	menghapus elemen pertama dari tabel
Table.slice()	mengirimkan satu tabel yang berisi sebagian elemen dari tabel utama
Table.splice()	menambahkan dan mengurangi elemen elemen tabel
Table.sort()	mengurutkan elemen elemen tabel
Table.unshift (nilai1[,nilai2, ...])	mengirimkan kembali kode sumber yang memungkinkan kita untuk membuat obyek array.
Table.toString()	mengirimkan kembali variabel string yang berhubungan dengan instruksi pembuatan obyek array
Table.unshift()	menambahkan kembali satu atau beberapa elemen pada bagian awal dari tabel
Table.valueOf()	mengembalikan nilai dari obyek array dimana obyek array itu yang jadi referensi dari tabel tersebut.

2.3.2 obyek boolean

obyek boolean adalah obyek standart dari JavaScript yang memungkinkan kita untuk memanipulasi nilai nilai jenis boolean. Berikut ini adalah sintaks yang digunakan untuk membuat obyek boolan :

```
var x = new Boolean(parameter)
```

parameter bisa berupa bisa berupa satu nilai (*True* atau *False*) atau bisa juga satu ekspresi, yang mana ekspresi akan di perhitungkan sebagai nilai boolean. Jika tidak ada nilai parameter yang dilewatkan atau nilai *0* atau string *kosong* atau *null* atau *undefined* atau juga *NaN*(*Not a Number*), nilai boolean akan diinisialisasikan ke *False*. Sebaliknya obyek boolean akan mempunyai nilai *True*

Berikut ini adalah tabel daftar beberapa metoda standart dari obyek boolean

Metoda	Keterangan
toSource()	Metoda ini mengirimkan kembali kode sumber yang memungkinkan kita untuk membuat obyek boolean
toString()	Metoda ini mengirimkan kembali string yang berhubungan dengan instruksi untuk membuat obyek boolean
valueOf()	Metoda ini mengembalikan nilai asal dari obyek boolean

2.3.3 obyek date

obyek date memungkinkan kita untuk bekerja dengan semua variabel yang berhubungan dengan penanggalan dan juga manajemen waktu (durasi waktu). Sintaks untuk membuat obyek *date* adalah berikut ini :

- `Nama_dari_obyek = new Date()`
sintaks ini memungkinkan kita untuk menyimpan tanggal dan jam saat ini.
- `Nama_dari_obyek = new Date("hari, bulan tanggal tahun jam:menit:detik")`
parameter berbentuk string dengan batas batas pemisah seperti format diatas.
- `Nama_dari_obyek = new Date(tahun, bulan, hari)`
parameter adalah 3 integer yang dipisahkan oleh tanda koma
- `Nama_dari_obyek = new Date(tahun, bulan, hari, jam, menit, detik[,perseribudetik])`
parameter adalah 6 integer yang dipisahkan oleh tanda koma

JavaScript menyimpan tanggal dalam bentuk string yang berisi hari, bulan, tahun, jam, menit, dan detik. Meskipun demikian sangat sulit bagi kita untuk bisa mengakses satu elemen waktu diatas dengan menggunakan obyek string (sub bab 2.3.5), karena setiap elemen mempunyai ukuran yang berbeda beda.. Sebaliknya obyek date memungkinkan kita untuk mengakses dan memodifikasi satu elemen tersebut.

Berikut ini adalah tabel beberapa metoda standart dari obyek date

Metoda	Keterangan	Jenis nilai hasil
getDate()	untuk memperoleh angka yang berkorespondensi dengan nomer hari dalam satu bulan.	hasilnya adalah satu integer dengan nilai antara 1 s/d 31 yang berkorespondensi dengan nomer hari dalam satu bulan
getDay()	untuk memperoleh angka yang berkorespondensi dengan nomer hari dalam satu minggunya	hasil adalah integer yang berhubungan dengan nomer hari dalam seminggu <ul style="list-style-type: none"> • 0 : minggu • 1 : senin • ...
getFullYear()	untuk memperoleh angka yang berkorespondensi dengan tahun dalam 4 bilangan penuh	hasilnya adalah integer yang berhubungan dengan tahun yang ditanyakan dengan format XXXX
getHours()	untuk memperoleh angka yang berkorespondensi dengan satuan jam	hasilnya adalah integer dengan nilai antara 0 sampai 23 yang berhubungan dengan obyek date
getMilliseconds()	untuk memperoleh angka yang berkorespondensi dengan	obyek hasil adalah integer antara 0 sampai 999 yang

	satuan perseribu detik	berkorespondensi dengan perseribudetik dari obyek yang dilewatkan di parameter.
getMinutes()	untuk memperoleh angka yang berkorespondensi dengan satuan menit	obyek hasil adalah integer antara 0 sampai 59 yang berkorespondensi dengan menit dari obyek yang dilewatkan di parameter
getMonth()	untuk memperoleh angka yang berkorespondensi dengan nomer bulan dalam setahun	hasil adalah integer yang berhubungan dengan nomer bulan dalam setahun <ul style="list-style-type: none"> • 0 : januari • 1 : februari •
getSeconds()	untuk memperoleh angka yang berkorespondensi dengan satuan detik	hasil adalah integer antara 0 sampai 59 yang berkorespondensi dengan detik dari obyek yang dilewatkan di parameter
getTime()	untuk memperoleh jumlah detik sejak tanggal 1 januari 1970	hasilnya adalah integer, metoda ini sangat berguna untuk melewati satu tanggal ke tanggal yang lain, atau juga menambahkan dua tanggal, etc
getTimeZoneOffset()	untuk memperoleh perbedaan waktu, antara waktu lokal dan GMT	hasilnya adalah integer yang berkorespondensi berapa menit perbedaan dengan GMT
getYear()	untuk memperoleh nilai tahun dalam 2 bilangan dari obyek date	hasilnya adalah integer yang berkorespondensi dengan tahun XX
toGMTString()	untuk mengkonversi satu tanggal menjadi satu string dengan format GMT	hasilnya adalah string dengan format Wed, 3 dec 2003: 15: 15: 20 GMT
toLocalString()	untuk mengkonversi satu tanggal menjadi satu string dengan format local	hasilnya adalah string dengan format tergantung dari sistem local sebagai contoh : 3/12/03 15: 15: 20
setDate(X)	untuk memberikan angka yang berkorespondensi dengan nomer hari dalam satu bulan	parameter adalah integer antara 1 dan 31 yang berkorespondensi dengan nomer hari dalam satu bulan
setDay(X)	untuk memberikan angka yang berkorespondensi dengan nomer hari dalam satu minggu	parameter adalah integer yang berkorespondensi dengan nomer hari dalam seminggu: <ul style="list-style-type: none"> • 0 : minggu • 1 : senin • ...
setHours(X)	untuk memberikan nilai jam	parameter adalah integer antara 0 sampai 23 yang berkorespondensi dengan angka jam
setMonth(X)	untuk memberikan angka yang berkorespondensi dengan nomer bulan	parameter adalah integer antara 0 sampai 11 yang berkorespondensi dengan nomer bulan <ul style="list-style-type: none"> • 0 : januari • 1 : februari •

setTime(X)	untuk menentukan tanggal	parameter adalah integer yang berkorespondensi dengan jumlah detik sejak tanggal 1 januari 1970
-------------------	--------------------------	---

2.3.4 obyek math

obyek math adalah suatu obyek yang mempunyai banyak metoda dan properti untuk memanipulasi bilangan bilangan dan juga fungsi fungsi matematika. Apapun metoda atau properti yang digunakan kita harus memulainya dengan kata *Math*, contohnya adalah sebagai berikut :

`Math.cos(1);`

Berikut ini adalah tabel beberapa metoda standart dari obyek math :

Metoda	Keterangan	Contoh
abs()	mengembalikan nilai absolut dari satu bilangan, kalau bilangan positif dia akan mengirimkan kembali bilangan itu, sebaliknya kalau bilangan negatif, dia akan mengirimkan bentuk positifnya	<code>x = Math.abs(3.17); //hasilnya x = 3.17</code> <code>x = Math.abs(-3.17); //hasilnya x = 3.17</code>
ceil()	mengembalikan nilai integer terkecil yang lebih besar sama dengan nilai parameter yang diberikan	<code>x = Math.ceil(6.01); //hasilnya x = 7</code> <code>x = Math.ceil(3.99); //hasilnya x = 4</code>
floor()	mengembalikan nilai integer terbesar yang lebih kecil sama dengan nilai parameter yang diberikan	<code>x = Math.floor(6.01); //hasilnya x = 6</code> <code>x = Math.floor(3.99); //hasilnya x = 3</code>
round()	membulatkan bilangan di parameter ke integer yang terdekat, jika bilangan tersebut 0,5 maka akan dibulatkan keatas	<code>x = Math.round(6.01); //hasilnya x = 6</code> <code>x = Math.round(3.80); //hasilnya x = 4</code> <code>x = Math.round(3.50); //hasilnya x = 4</code>
max(bil1,bil2)	mengembalikan nilai terbesar diantara dua parameter yang dibandingkan	<code>var x = Math.max(6,7.25); //hasilnya x = 7.25</code> <code>var x = Math.max(-8.21,-3.65); //hasilnya x = -3.65</code>
min(bil1,bil2)	mengembalikan nilai terkecil diantara dua parameter yang di bandingkan	<code>x = Math.min(6,7.25); //hasilnya x = 6</code> <code>x = Math.min(-8.21,-3.65); //hasilnya x = -8.21</code>

pow(bil1,bil2)	mengembalikan nilai hasil parameter satu pangkat parameter dua	<pre>x = Math.pow(3,3); //hasilnya x = 27 x = Math.pow(9,0.5); //hasilnya x = 3</pre>
random()	Mengembalikan nilai acak antara 0 sampai 1, nilai di generate berdasarkan sistem jam dari komputer	<pre>x = Math.random(); //hasilnya x = 0.6489534931546957</pre>
sqrt(bil)	Mengembalikan akar dari bilangan yang dilewatkan sebagai parameter	<pre>x = Math.sqrt(9); //hasilnya x = 3</pre>

2.3.5 obyek string

obyek string adalah satu obyek yang berisi beberapa metoda dan properti untuk memanipulasi data jenis string. Obyek string sendiri hanya mempunyai satu properti yaitu properti *length* untuk memperoleh panjang dari variabel data string. Sintaks dari properti ini adalah sebagai berikut :

```
x = nama_variabel_string.length;  
x = ('sembarang teks').length;
```

metoda dari obyek string memungkinkan kita untuk memperoleh satu potongan/bagian dari data string dan juga memodifikasinya. Berikut ini adalah tabel beberapa metoda standard dari obyek string :

Metoda	Keterangan
string.big()	menaikkan ukuran huruf satu point
string.small()	menurunkan ukuran huruf satu point
string.blink()	mentransformasi teks menjadi teks berkedip kedip
string.bold()	memtransformasi huruf di teks teks menjadi huruf tebal (tag)
string.charAt(posisi)	mengembalikan karakter di posisi ke parameter posisi
string.charCodeAt(posisi)	mengembalikan kode <i>Unicode</i> di posisi ke parameter posisi
concat(teks1,teks2,...)	menyambungkan teks teks yang dimasukkan sebagai parameter dari ujung ke ujung
string.fontcolor(warna)	memodifikasi warna dari teks (parameter warna, bisa berupa teks ataupun bilangan heksadesimal)
string.fontSize(ukuran)	memodifikasi ukuran dari teks (parameter ukuran dalam bentuk integer)
string.indexOf(substring, posisi)	mengembalikan posisi dari substring (huruf atau kumpulan dari huruf huruf dari satu string), dengan pencarian dari arah kiri ke kanan, dimulai dari lokasi parameter posisi.
string.lastIndexOf(substring, posisi)	fungsinya sama dengan metoda indexOf akan tetapi pencarian dari arah kanan ke kiri
string.italics()	mentransformasi huruf di teks menjadi huruf miring (tag <i>)
string.link(URL)	mentransformasikan teks menjadi hiperteks (tag <a href>)

string.strike()	mentransformasi huruf di teks menjadi huruf dicoret teks (tag <strike>)
string.sub()	mentransformasikan huruf di teks menjadi huruf ^{teks} (tag <sub>)
string.sup()	mentransformasikan huruf di teks menjadi huruf ^{teks} (tag <sup>)
string.substr(posisi, panjang)	mengembalikan substring (beberapa huruf atau kata) yang dimulai dari parameter posisi ke berapa dan sepanjang parameter panjang unit
string.substring(posisi1, posisi2)	mengembalikan substring (beberapa huruf atau kata) yang terletak diantara parameter posisi1 dan parameter posisi2
string.toLowerCase()	mentransformasi semua huruf dalam teks menjadi huruf kecil
string.toUpperCase()	mentransformasi semua huruf dalam teks menjadi huruf besar

Berikut ini adalah beberapa contoh penggunaan metoda obyek string :

```
var test= 'ilmu komputer' ;
```

- var hasil = charAt(test,6); //hasilnya 'k'
- var hasil = ("ilmu komputer").charAt(7); //hasilnya 'o'
- var hasil = charAt(test,-1); //hasilnya "
- var hasil = test.substring(1,5); //hasilnya 'Imu k'
- var hasil = test.toUpperCase(); //hasilnya 'ILMU KOMPUTER'

3. Variable

3.1 Konsep Variabel

Variable adalah suatu obyek yang berisi data data, yang mana dapat di modifikasi selama pengekseskuan program. Di JavaScript kita bisa memberikan nama variabel sepanjang yang kita suka, akan tetapi harus memenuhi kriteria berikut ini .

- Nama variabel harus dimulai oleh satu huruf (huruf besar maupun huruf kecil) atau satu karakter "_".
- Nama variabel bisa terdiri dari huruf huruf, angka angka atau karakter _ dan & (spasi kosong tidak diperbolehkan).
- Nama variabel tidak boleh memakai nama nama berikut ini (reserved oleh program)
 - *abstract*
 - *boolean break byte*
 - *case catch char class const continue*
 - *debugger default delete do double*
 - *else export extends*
 - *false final finally float for function*
 - *goto*
 - *if, implements, import, in, infinity, instanceof, int, interface*
 - *label, long*

- o *native, new, null*
- o *package, private, protected, public*
- o *return*
- o *short, static, super, switch, synchronized*
- o *this, throw, throws, transient, true, try, typeof*
- o *var, void, volatile*
- o *while, with*

berikut ini adalah contoh pemberian nama variabel yang benar dan tidak benar :

Nama variabel yang benar	Nama variabel yang tidak benar	Alasan
Variabel	Nama Dari Variabel	Ada spasi kosong
Nama_Dari_Variabel	123Nama_Dari_Variabel	Dimulai dgn angka
nama_dari_variabel	andry@yahoo.com	Karakter @
nama_dari_variabel_123	Nama-Dari-Variabel	Karakter -
_707	transient	Reserved word



di JavaScript kita menggunakan sistem *case sensitive* yang artinya membedakan nama variabel dengan huruf besar dan huruf kecil, oleh karena itu biasanya memberikan nama variabel dengan aturan yang sama, huruf besar semua atau huruf kecil semua (disarankan menggunakan selalu huruf kecil)

3.2 Mendeklarasikan Variabel

Penulisan variabel JavaScript sangatlah fleksibel, dan tidaklah terlalu rumit dan ketat, sehingga kita tidaklah terlalu sering menerima pesan error pada saat menjalankan program. Sebagai contoh deklarasi variabel di JavaScript dapat kita lakukan dengan dua cara :

- **eksplisit** : dengan menuliskan kata kunci *var* kemudian diikuti dengan nama variabel dan nilai dari variabel :

```
var test = "halo"
```

- **implisit** : dengan menuliskan secara langsung nama dari variabel dan diikuti nilai dari variabel :

```
test = "halo"
```

navigator secara otomatis akan memperlakukan pernyataan itu sebagai deklarasi dari sebuah variabel. Pada navigator versi lama mungkin terjadi kasus di mana navigator tidak mengenali pendeklarasian variabel secara implisit, maka disarankan untuk menggunakan cara eksplisit dalam menulis program JavaScript.

Berikut ini adalah contoh pendeklarasian variabel dengan kedua cara tersebut.

```
<SCRIPT language="Javascript">  
<!--
```

```
var VariabelKu;  
var VariabelKu2 = 3;  
VariabelKu = 2;  
  
document.write(VariabelKu*VariabelKu2);  
// -->  
</SCRIPT>
```

3.3 Peletakan variabel (global atau lokal)

berdasarkan tempat dimana kita mendeklarasikan suatu variabel, variabel bisa diakses dari seluruh bagian program atau hanya di dalam bagian tertentu dari program. Pada saat suatu variabel di deklarasikan tanpa menggunakan kata kunci *var*, atau bisa kita sebut dengan cara **implisit**, maka variabel itu bisa di akses dari seluruh bagian program(semua fungsi di dalam program dapat memanggil dan memakai variabel ini), dan kita sebut variabel ini sebagai **variabel global**.

Sebaliknya jika kita mendeklarasikan dengan cara **eksplisit** suatu variabel JavaScript (pendeclarasian variabel dengan menggunakan kata kunci *var*), maka kemungkinan pengaksesan variabel tersebut bergantung lokasi dimana dia dideklarasikan :

- Jika dia dideklarasikan dibagian awal dari skrip program, yang artinya sebelum pendeklarasian semua fungsi, maka semua fungsi di dalam program bisa mengakses variabel ini, dan variabel ini menjadi **variabel global**.
- Jika dia deklarasikan dengan menggunakan kata kunci *var* di dalam suatu fungsi tertentu, maka variabel itu hanya bisa di akses dari dalam fungsi tersebut, dan artinya variabel ini tidak berguna bagi fungsi fungsi yang lain, dan kita sebut variabel ini menjadi **variabel lokal**

Mari kita lihat contoh berikut ini :

```
<SCRIPT language="Javascript">  
<!--  
  
var a = 12;  
var b = 4;  
function PerkalianDengan2(b) {  
    var a = b * 2;  
    return a;  
}  
  
document.write("Dua kali dari ",b," adalah ",PerkalianDengan2(b));  
document.write("Nilai dari a adalah",a);  
  
// -->  
</SCRIPT>
```

Dari contoh diatas, variabel a dideklarasikan secara eksplisit di awal dari skrip program dan juga di deklarasikan di dalam fungsi . berikut ini hasil dari program diatas.

```
Dua kali dari 4 adalah 8  
Nilai dari a adalah 12
```

Berikut ini adalah contoh lain dimana variabel di deklarasikan secara implisit di dalam suatu fungsi :

```
<SCRIPT language="Javascript">
<!--

var a = 12;
var b = 4;

function PerkalianDengan2(b) {
    a = b * 2;
    return a;
}

document.write("Dua kali dari ",b," adalah ",PerkalianDengan2(b));
document.write("Nilai dari a adalah",a);

// -->
</SCRIPT>
```

Berikut ini hasil dari program diatas.

Dua kali dari 4 adalah 8
Nilai dari a adalah 8



Dari contoh diatas bisa kita lihat pentingnya kita membiasakan diri untuk menggunakan kata *var* pada saat membuat variabel baru.

3.4 Jenis jenis data dari variabel

Di JavaScript, kita tidak perlu mendeklarasikan jenis variabel yang akan kita gunakan, sebaliknya di bahasa pemrograman yang lain (yang lebih advanced) seperti bahasa C atau Java kita harus mendeklarasikan secara detail apakah variabel yang digunakan tersebut adalah merupakan suatu bilangan bulat (*int*), bilangan desimal (*float*), karakter (*char*), dan lainnya ...

Sebenarnya di JavaScript sendiri, kita hanya bisa memanipulasi 4 jenis data yaitu :

- Bilangan : bulat atau desimal, yang kita sebut sebagai *integer* atau *float*
- Kata (kumpulan huruf) : kita sebut *string*
- *Boolean* : suatu variabel yang mempunyai dua nilai dan berfungsi untuk memeriksa suatu kondisi :
 - *true* : jika kondisinya benar
 - *false* : jika kondisinya salah
- variabel dengan jenis *null* : satu kata khusus (termasuk keyword juga) untuk menjelaskan bahwa tidak ada data didalamnya.

3.4.1 Integer(bilangan bulat)

bilangan bulat dapat ditampilkan dalam beberapa basis berikut ini :

- *basis desimal* : integer di tuliskan dalam urutan unit bilangan (dari 0 sampai dengan 9), permulaan bilangan tidak boleh dimulai oleh angka 0
- *basis heksadesimal* : dituliskan dalam urutan unit bilangan dari 0 sampai dengan 9 atau urutan huruf dari A sampai dengan F (atau a sampai dengan f), permulaan bilangan dimulai oleh 0x atau 0X
- *basis oktal* : dituliskan dalam urutan unit angka dari 0 sampai dengan 7, permulaan bilangan dimulai dengan angka 0

3.4.2 Float (bilangan desimal)

bilangan desimal bisa kita sebut juga sebagai bilangan pecahan atau bilangan yang bisa kita tuliskan dalam bentuk menggunakan tanda koma. Bilangan ini juga bisa di tuliskan dengan beberapa cara berikut

- *bilangan bulat desimal* : 895
- *bilangan dengan tanda koma* : 895,12
- *bilangan pembagian* : 27/11
- *bilangan eksponensial* : bilangan dengan tanda koma , kemudian diikuti oleh huruf e(atau E), kemudian diikuti oleh bilangan bulat yang artinya pangkat dari bilangan 10 (+ atau -, pangkat positif atau negatif), contoh :

```
var a = 2.75e-2;  
var b = 35.8E+10;  
var c = .25e-2;
```

3.4.3 String

String adalah kumpulan dari karakter, kita deklarasikan variabel string menggunakan tanda (') atau ("), kedua tanda tersebut harus digunakan secara berpasangan dan tidak bisa digunakan secara sendiri sendiri atau bercampur. Berikut ini adalah beberapa cara untuk mendeklarasikan variabel string :

```
var a = "Hallo";  
var b = 'Sampai Ketemu Lagi !';
```

Ada beberapa karakter spesial yang bisa kita gunakan untuk mensimulasikan bagian dari karakter yang tidak terlihat (non visual) dan juga untuk menghindari kemungkinan navigator "mengalami kebingungan" dalam membedakan antara string dan skripnya sendiri, karakter spesial ini menggunakan simbol antislash (\), beberapa contoh karakter spesial tersebut

- \n : kembali ke baris awal
- \r : menekan tombol ENTER
- \t : tab
- \" : tanda petik ganda
- \' : tanda petik tunggal
- \\ : karakter antislash

satu contoh lagi, misalnya kita ingin menyimpan variabel judul (string) berikut ini :

Ada apa di dalam "c:\windows\"

Kita harus menuliskannya dalam bentuk berikut ini di dalam JavaScript :

```
Judul = "Ada apa di dalam \"c:\\windows\\\"";
```

Atau bisa juga dengan cara berikut ini (menggunakan tanda petik tunggal) :

```
Judul = 'Ada apa di dalam "c:\\windows\\"';
```

Untuk memanipulasi variabel String, JavaScript mempunyai satu obyek yang bernama obyek String (lihat subbab 2.2.5), yang terdiri dari beberapa metode untuk membuat variabel string dan memanipulasinya.

3.4.4 Booleans

boolean adalah satu variabel khusus yang berguna untuk mengevaluasi suatu kondisi tertentu, oleh karenanya boolean mempunyai dua nilai :

- *True* : diwakili oleh nilai 1
- *False* : diwakili oleh nilai 0

3.5 Konversi jenis variabel

Meskipun JavaScript memungkinkan pengaturan perubahan jenis variabel secara transparan, kadang kadang kita perlu juga untuk melakukan konversi jenis variabel secara paksa. Ada 2 fungsi dasar yang memungkinkan merubah jenis variabel yang dilewatkan dengan parameter tertentu :

- `parseInt()`

Fungsi ini mungkinkan merubah satu variabel yang dilewatkan dengan parameter tertentu (bisa dalam bentuk string ataupun dalam bentuk bilangan dalam basis yang disebutkan di parameter kedua) menjadi bilangan bulat. Sintaksnya adalah sebagai berikut :

```
parseInt(string[, basis]);
```

Agar supaya fungsi `parseInt()` mengembalikan nilai bilangan bulat, maka parameter yang dilewatkan harus dimulai dengan karakter bilangan [0-9], prefiks hexadesimal `0x`, dan/atau karakter `+`, `-`, `e`, dan `E`. Selain daripada itu maka fungsi `parseInt()` akan mengembalikan nilai *NaN* (*Not a Number*). Jika karakter berikutnya tidak valid, maka dia akan diabaikan oleh fungsi `parseInt()`, dan akan ditampilkan terpotong jika di bagian depan karakter valid dan bagian belakang karakter tidak valid.

Berikut ini salah satu contoh penggunaan fungsi `parseInt()` :

```
var a = "123";  
var b = "456";
```

```
document.write(a+b,"<BR>"); // hasil 123456  
document.write(parseInt(a)+parseInt(b),"<BR>"); // hasil 579
```

tabel berikut ini memberikan sedikit gambaran penggunaan dari fungsi `parseInt()` :

Contoh	Hasil
<code>parseInt("128.34");</code>	128
<code>parseInt("12.3E-6");</code>	12

<code>parseInt("12E+6");</code>	12
<code>parseInt("Halo");</code>	NaN
<code>parseInt("24Halo38");</code>	24
<code>parseInt("Halo3824");</code>	NaN
<code>parseInt("AF8BEF");</code>	NaN
<code>parseInt("0284");</code>	284
<code>parseInt("0284",8);</code>	2
<code>parseInt("AF8BEF",16);</code>	11504623
<code>parseInt("AB882F",16);</code>	11241519
<code>parseInt("0xAB882F");</code>	11241519
<code>parseInt("0xAB882F",16);</code>	11241519
<code>parseInt("00100110");</code>	100110
<code>parseInt("00100110",2);</code>	38
<code>parseInt("00100110",8);</code>	32840
<code>parseInt("00100110",10);</code>	100110
<code>parseInt("00100110",16);</code>	1048848

- `parseFloat()`

Adalah satu fungsi inti dari JavaScript yang memungkinkan merubah variabel yang dilewatkan dengan parameter tertentu menjadi bilangan desimal, Sintaks dari fungsi `parseFloat` adalah sebagai berikut :

`parseFloat(string);`

tabel contoh tentang penggunaan fungsi `parseFloat()`

Contoh	Hasil
<code>parseFloat("128.34");</code>	128.34
<code>parseFloat("128,34");</code>	128
<code>parseFloat("12.3E-6");</code>	0.0000123
<code>parseFloat("Halo");</code>	NaN
<code>parseFloat("24.568Halo38");</code>	24.568
<code>parseFloat("Halo38.24");</code>	NaN
<code>parseFloat("AF8BEF");</code>	NaN
<code>parseFloat("0284");</code>	284
<code>parseFloat("0xAB882F");</code>	11241519

4. Tabel

4.1 Pengenalan Tabel

Variabel variabel di JavaScript hanya bisa menyimpan satu data pada suatu saat. Adanya kecenderungan yang sangat besar untuk memanipulasi banyak data dalam satu variabel membuat konsep variabel tersebut menjadi tidak cukup memenuhi kebutuhan itu. Untuk mengatasi hal tersebut, JavaScript menghadirkan solusi struktur data yang memungkinkan menyimpan himpunan/group data dalam satu variabel khusus yang disebut : *Tabel*.

Tabel, di dalam javascript adalah satu variabel yang dapat berisi banyak data yang independent, dengan indeks berdasarkan nomer urut, dengan indeks ini pula memungkinkan kita untuk mengakses data yang disimpan di lokasi tertentu.

4.2 Tabel multidimensi

Tabel yang hanya berisi variabel variabel kita sebut tabel monodimensi atau unidimensi, berikut ini ilustrasi tabel unidimensi :

Indeks	0	1	2	3
Data	Data1	Data2	Data3	Data4



Perhatikan bahwa element pertama dari tabel selalu mempunyai indeks 0. Dalam suatu tabel dengan n element, maka element ke n akan mempunyai indeks n-1.

Jika suatu tabel berisi tabel yang lain, maka kita sebut tabel itu adalah tabel multidimensi, berikut ini ilustrasi tabel multidimensi :

0	1	2	3			
Data 1 (variabel)	Data 2 (tabel)			Data 3 (variabel)	Data 4 (tabel)	
	0	1	2		0	1
	Data 1	Data 2	Data 3		Data 1	Data 2

4.3 Tabel asosiatif

Kita juga bisa menggunakan indeks yang sudah kita modifikasi, untuk memudahkan menandai indeks dari satu nilai data. Tabel yang menggunakan indeks seperti ini kita sebut tabel asosiatif. JavaScript sendiri memungkinkan kita untuk menggunakan string ataupun bilangan yang sudah di definisikan sebagai indeks data dari suatu tabel, berikut ini adalah ilustrasi dari tabel asosiatif

Indeks	"Andry"	"Indra"	"Santi"	"Sandra"
Data	10	24	77	47

4.4 Pembuatan tabel

Ada beberapa cara untuk membuat tabel di dalam JavaScript :

```
var TabelKu = ["data 1", "data 2", "data 3", "data 4"];
```

```
var TabelKu = new Array("data 1", "data 2", "data 3", "data 4");
```

pada cara diatas tersebut tabel diinisialisasikan dengan nilai data pada saat pembuatannya. Untuk lebih jelasnya sebaiknya kita deklarasikan tabel sebelum kita isikan dengan data, contoh deklarasinya adalah sebagai berikut :

```
var TabelKu = new Array();
```

4.5 Pengaksesan data di dalam tabel

Pengaksesan data atau elemen di dalam tabel dapat dilakukan dengan menuliskan nama tabel diikuti tanda kurung yang berisi indeks dari elemen.

```
var TabelKu = ["Buaya", "Harimau", "Gajah", "Singa", "Jerapah", "Zebra"];
```

```
document.write("elemen ke 4 dari tabel adalah "+TabelKu[3]);
```

```
//hasil "elemen ke 4 dari tabel adalah Singa"
```

4.6 Memasukan data pada tabel

Untuk membuat tabel asosiatif, cukup dengan mendeklarasikan variabel yang akan kita masukkan, kemudian menuliskan nama tabel, di ikuti dengan nama indeks dalam kurung, dan memasukkan nilai datanya :

```
TabelKu[0] = "Hallo";
```

```
TabelKu["Andry"] = 10;
```

```
TabelKu["Sandra"] = 47;
```

4.7 Memanipulasi tabel

Untuk memanipulasi tabel, JavaScript mempunyai satu obyek yang bernama obyek Array (lihat subbab 2.2.1) , yang memungkinkan kita memanipulasi tabel, seperti insert (memasukkan), delete (menghapus) atau mengambil elemen tertentu (ekstrak) ke/dari suatu tabel.

5. Event

Event adalah aksi dari user yang bisa menghasilkan interaktifitas, pada kenyataannya event di dalam JavaScript adalah klik dari tombol mouse (merupakan satu satunya aksi yang dapat diatur oleh HTML). JavaScript memungkinkan mengasosiasikan event dengan beberapa fungsi dari metode seperti lewatnya mouse pointer diatas zona tertentu, perubahan nilai, dan lain sebagainya.

Event administrator adalah yang memperbolehkan kita untuk mengasosiasikan satu aksi ke dalam sebuah event. Sintaks dari event administrator tersebut adalah sebagai berikut :

```
OnEvenement = "Aksi_Javascript_atau_Fungsi();"
```

Untuk penggunaan link hiperteks, maka sintaksnya adalah sebagai berikut :

```
<a href ="onEvenement ='Aksi_Javascript_atau_Fungsi();">Link</a>
```

5.1 Daftar event

Berikut ini beberapa kode yang harus dimasukkan ke dalam tag event administrator untuk menghasilkan aksi tertentu.

Event	Keterangan
Abort (onAbort)	terjadi pada saat user mengagalkan proses download image
Blur (onBlur)	terjadi ketika elemen kehilangan fokus, artinya user melakukan klik diluar elemen itu,
Change (onChange)	terjadi pada saat user memodifikasi isi dari data dalam satu field data
Click (onClick)	terjadi pada saat user melakukan klik mouse terhadap satu elemen yang berhubungan dengan event
Dblclick (onDblclick)	terjadi pada saat user melakukan klik dua kali pada satu elemen yang berhubungan event, elemen bisa berupa satu hiperlink atau elemen dari satu form. Event ini hanya mensupport JavaScript ver 1.2 keatas
Dragdrop (onDragdrop)	terjadi pada saat user melakukan drag dan drop elemen di dalam navigator. Hanya mensupport JavaScriptver 1.2 keatas
Error (onError)	muncul ketika error pada saat loading halaman. Hanya support JavaScript ver 1.1 keatas
Focus (onFocus)	terjadi pada saat user memberikan focus kepada satu elemen
Keydown (onKeydown)	terjadi pada saat user menekan satu tombol pada keyboardnya. Hanya mensupport JavaScript ver 1.2 keatas
Keypress (onKeypress)	terjadi pada saat user menekan dan manahan tombol di keyboardnya tetap ditekan. Hanya mensupport JavaScript ver 1.2 keatas
KeyUp (onKeyUp)	terjadi pada saat user melepaskan tombol pada keyboardnya. Hanya mensupport JavaScript versi 1.2 keatas
Load (onLoad)	terjadi pada saat navigator user meload/memanggil suatu halaman
Mouseover (onMouseover)	terjadi pada saat user meletakkan kursor mouse diatas suatu elemen
Mouseout (onMouseout)	terjadi pada saat kursor mouse meninggalkan posisinya dari atas suatu elemen
Reset (onReset)	terjadi pada saat user menghapus data pada suatu form dengan bantuan satu tombol reset
Resize (onResize)	terjadi pada saat user merubah dimensi ukuran dari jendela navigator

Select (onSelect)	terjadi pada saat user melakukan proses select terhadap suatu teks (sebagian atau semuanya) di dalam satu field bertjenis teks atau textarea
Submit (onSubmit)	terjadi pada saat user melakukan klik terhadap tombol pengiriman suatu form
Unload (onUnload)	terjadi pada saat navigator user meninggalkan halaman yang sedang diproses (atau di load)

5.2 Hubungan event dengan obyek

Tidak semua elemen bisa berhubungan atau berasosiasi dengan sembarang obyek. Sangat jelas sekali, sebagai contoh event *onChange* tidak akan bisa diaplikasikan ke suatu hiperteks. Berikut ini adalah tabel rekapitulatif obyek obyek yang mana bisa berasosiasi dengan suatu event.

Event	Obyek yang bisa berasosiasi
Abort	Image
Blur	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, Window
Change	FileUpload, Select, Submit, Text, TextArea
Click	Button, document, Checkbox, Link, Radio, Reset, Select, Submit
Dblclick	Document, Link
dragdrop	Window
Error	Image, Window
Focus	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, Window
Keydown	Document, Image, Link, TextArea
KeyPress	Document, Image, Link, TextArea
KeyUp	Document, Image, Link, TextArea
Load	Image, Layer, Window
Mousedown	Button, Document, Link
Mousemove	Not spesific
Mouseout	Layer, Link
Mouseover	Area, Layer, Link
Mouseup	Button, Document, Link
Move	Window
Reset	Form
Resize	Window
Select	Text, TextArea
Submit	Form
Unload	Window

5.3 Contoh penggunaan event

cara terbaik untuk mengerti tentang penggunaan event adalah dengan cara berlatih menulis program kecil kecilan. Sebagai inspirasi anda, mungkin anda bisa mencoba melihat source code dari beberapa halaman web, akan tetapi marilah kita tidak langsung melakukan copy & paste

terhadap skrip itu dan membiasakan diri untuk menghormati hasil kerja orang lain dengan meminta ijin terlebih dahulu kepada si pembuat skrip tersebut, kecuali memang skrip tersebut tersedia secara free, seperti banyak tersedia di beberapa site yang menyediakan source code dari JavaScript.

Berikut ini kita akan mencoba satu contoh sederhana cara menampilkan kotak dialog yang berisi tulisan **'teks anda'** (dengan tanda satu petik di depan dan di belakang kata) sesudah anda melakukan klik terhadap satu link yang mentrigger event untuk mengaktifkan kotak dialog tersebut.

```
<html>
<head>
<title>Membuka kotak dialog pada saat melalukan klik mouse di satu link</title>
</head>
<body>

<a href="javascript:;"
onClick="window.alert('\`teks anda\`');">
klik disini !</a>

</body>
</html>
```

analisa skrip diatas :

- Event administrator *onClick* dimasukkan didalam tag links dari hiperteks
- Tujuan dari skrip ini adalah menampilkan satu kotak dialog, jadi kita tidak ingin kalau link tersebut akan membawa kita ke halaman yang lain, oleh karena itu kita masukkan kode "javascript:;" didalam atribut *href* untuk memberi tahu navigator kalau kita ingin tetap berada di halaman yang sedang di proses tersebut.
- Perhatikan penggunaan \` di dalam kalimat **'teks anda'** . tanda antislash (\) di depan tanda petik untuk memperingatkan navigator kalo dianggap sebagai karakter biasa dan bukan di intepretasi sebagai pembatas string.

Selanjutnya contoh kedua adalah contoh penggunaan event *onMouseover* untuk membuat menu interaktif yang akan berubah tampilan imagenya pada saat kursor mouse lewat diatasnya, kita bahkan bisa menambahkan event *onMouseout* untuk mengembalikan image ke asalnya pada saat kursor sudah tidak diatas image lagi. Karena di sini kita tidak bisa mencoba secara live, anda bisa mencoba sendiri di rumah dengan mengcopy paste skrip ini dan mengganti file image1.gif dan image2.gif dengan sembarang file yang anda punya .

```
<html>
<head>
<title>Memodifikasi image pada saat kursor mouse bergerak diatas image tersebut</title>
</head>
<body>

<a href="javascript:;"
onMouseOver="document.img_1.src='image2.gif';"
onMouseOut="document.img_1.src='image1.gif';">
 </a>

</body>
</html>
```


analisa skrip diatas :

- Untuk bisa mengasosiasikan image dengan event tersebut, maka kita harus membuat image itu sebagai suatu link, sehingga kita memakai tag `<img.>` diantara tag `<a ...>` dan ``
- Event `onMouseover` dan `onMouseout` terbatas penggunaannya untuk JavaScript versi 1.1 keatas

6. Operator

Operator adalah simbol yang memungkinkan kita untuk memanipulasi variabel, dengan kata lain melakukan operasi operasi , mengevaluasi, dan lain lainnya. Ada beberapa jenis operator :

6.1 Operator Penghitungan (Calculation)

Operator penghitungan memungkinkan kita untuk memodifikasi nilai dari variabel secara matematika.

Operator	Keterangan	Fungsi	Contoh	Hasil (dgn x=7)
+	penjumlahan	menjumlahkan dua nilai	$x+3$	10
-	pengurangan	mengurangkan satu nilai dari nilai yang lain	$x-3$	4
*	perkalian	mengalikan dua nilai	$x*3$	21
/	pembagian	membagi satu nilai dengan nilai yang lain	$x/3$	2,3333
=	afektasi nilai	memberikan satu nilai terhadap satu variabel	$x=3$	Memberikan nilai 3 terhadap variabel X
%	modulo	mengembalikan nilai sisa pembagian dari nilai sebelah kiri dibagi dengan nilai sebelah kanan	$x\%2$	1

6.2 Operator Afektasi

Operator ini memungkinkan kita untuk menyederhanakan operasi penambahan nilai dalam satu variabel dan menyimpan hasilnya di dalam variabel itu sendiri. Operasi ini biasanya ditulis dengan cara berikut : $x=x+2$, dengan menggunakan operator afektasi operasi tersebut bisa dituliskan menjadi $x+=2$, dan jika nilai awal $x=7$ maka nilai akhir x menjadi 9.

Jenis jenis operator seperti ini adalah sebagai berikut :

Operator	Fungsi
$+=$	menambahkan nilai sebelah kiri dengan nilai sebelah kanan, kemudian menyimpan hasilnya variabel sebelah kiri
$-=$	mengurangi nilai sebelah kiri dengan nilai sebelah kanan, kemudian

	menyimpan hasilnya variabel sebelah kiri
*=	mengalikan nilai sebelah kiri dengan nilai sebelah kanan, kemudian menyimpan hasilnya variabel sebelah kiri
/=	membagi nilai sebelah kiri dengan nilai sebelah kanan, kemudian menyimpan hasilnya variabel sebelah kiri
%=	menghitung sisa pembagian nilai sebelah kiri oleh nilai sebelah kanan, kemudian menyimpannya variabel sebelah kiri

6.3 Operator Inkrementasi

Operator ini memungkinkan kita untuk menambahkan ataupun mengurangi per unit dari satu variabel. Operator ini sangat berguna dalam struktur pemrograman sistem Loop, yang membutuhkan penghitung (variabel yang nilainya naik/turun satu persatu) .

Operator dengan model x++ bisa menggantikan notasi x=x+1 atau x+=1

Operator	Fungsi	Sintaks	Hasil (dgn x=7)
++	menambahkan nilai satu unit dari variabel	x++	8
--	mengurangi nilai satu unit dari variabel	x--	6

6.4 Operator Perbandingan

Operator	Fungsi	Sintaks	Hasil (dgn x=7)
== (jgn bingung dengan operator =) !	membandingkan dua nilai dan memverifikasi kesamaannya	x==3	mengembalikan nilai <i>True</i> , kalau x sama dengan 3, sebaliknya nilai <i>False</i>
===	memverifikasi apakah identitas dari dua nilai dan jenis dari kedua nilai tersebut sama	a===b	mengembalikan nilai <i>True</i> jika a sama dengan b, dan mempunyai jenis yang sama, sebaliknya nilai <i>False</i>
!=	memverifikasi apakah satu variabel berbeda dengan satu nilai	x!=3	mengembalikan nilai 1, jika x tidak sama (berbeda) dengan 3, sebaliknya nilai 0
!==	memverifikasi apakah dua nilai tidak bernilai sama atau juga tidak berjenis sama	a!==b	mengembalikan nilai <i>True</i> , jika a berbeda dengan b atau berbeda jenis, sebaliknya nilai <i>False</i>
<	memverifikasi apakah satu nilai lebih kecil dari nilai yang lain	x<3	mengembalikan nilai <i>True</i> , jika x lebih kecil dari 3, sebaliknya nilai <i>False</i>

<=	memverifikasi apakah satu nilai lebih kecil atau sama dengan nilai yang lain	x<=3	mengembalikan nilai <i>True</i> , jika x lebih kecil atau sama dengan 3, sebaliknya nilai <i>False</i>
>	memverifikasi apakah satu nilai lebih besar dari nilai yang lain	x>3	mengembalikan nilai <i>True</i> , jika x lebih besar dari 3, sebaliknya nilai <i>False</i>
>=	memverifikasi apakah satu nilai lebih besar atau sama dengan nilai yang lain	x>=3	mengembalikan nilai <i>True</i> , jika x lebih besar atau sama dengan 3, sebaliknya nilai <i>False</i>

6.5 Operator Logika (Booleans)

Operator jenis ini memungkinkan kita untuk memverifikasi apakah beberapa kondisi sudah benar

Operator	Keterangan	Fungsi	Sintaks
	logika OR (atau)	memverifikasi apakah syarat satu kondisi sudah terpenuhi ?	((kondisi1) ((kondisi2))
&&	logika AND (dan)	memverifikasi apakah semua kondisi sudah memenuhi syarat ?	((kondisi1))&&((kondisi2))
!	logika NON	Membalikkan kondisi nilai (invers) dari variabel boolean (mengembalikan nilai 1, jika variabel bernilai 0, dan mengembalikan nilai 0, jika variabel bernilai 1)	(!kondisi)

6.6 Operator untuk memanipulasi data string

Operator + yang digunakan untuk variabel jenis String memungkinkan kita untuk melakukan penggabungan (concatenate) dua variabel String. Perlu dicatat juga bahwa *var='a'+b'* adalah sama dengan *var='ab'*

```
var1='a'
var=var1+'b' -> var='ab'
```

6.7 Prioritas

Pada saat kita melibatkan banyak operator dalam satu operasi, navigator harus tahu dengan urutan mana operasi dilakukan berdasarkan prioritas dari operator.

Berikut ini adalah tabel tingkat prioritas dari seluruh operator.

Prioritas Operator												
1	()	[]										
2	--	++	!	~	-							
3	*	/	%									
4	+	-										
5	<	<=	>=	>								
6	==	!=										
7	^											
8												
9	&&											
10	?	:										
11	=	+=	-=	*=	/=	%=	<<=	>>=	>>>=	&=	^=	=
12	,											

7. Struktur Kondisional

Struktur kondisional adalah instruksi instruksi yang memungkinkan kita untuk melakukan test apakah satu kondisi adalah benar atau tidak, dan memungkinkan juga terjadinya proses interaksi di dalam skrip yang kita buat

7.1 Instruksi if

Instruksi paling dasar, kita bisa temukan instruksi ini hampir dalam semua bahasa pemrograman (mungkin dengan sedikit perbedaan sintaks). Instruksi ini memungkinkan kita untuk mengeksekusi satu blok instruksi jika kondisi syaratnya terpenuhi.

Sintaks dari instruksi ini adalah sebagai berikut :

```
if (kondisi syarat terpenuhi) {
    daftar instruksi atau blok instruksi
}
```

beberapa catatan penting tentang instruksi ini

- kondisi harus terletak diantara dua tanda kurung
- kita bisa melatakkkan beberapa kondisi dengan menggunakan operator AND atau OR (&& atau ||)

contoh :

```
if ((kondisi1)&&(kondisi2))
if ((kondisi1) || (kondisi2))
```



Jika hanya ada satu instruksi di dalam blok instruksi, kita tidak perlu menggunakan tanda kurung, seperti contoh berikut ini :

```
if (x==2) document.write("Nilai x adalah 2");
```

7.2 Instruksi if..else

Instruksi If, adalah instruksi dasar yang hanya memungkinkan kita untuk melakukan pemeriksaan apakah satu kondisi terpenuhi atau tidak. Pada kenyataannya kita menginginkan lebih dari satu kondisi syarat untuk menjalankan program, untuk kebutuhan itu, kita bisa menggunakan instruksi If ...Else....

Sintaks lengkap dari instruksi ini adalah :

```
if (kondisi syarat1 terpenuhi) {  
    daftar instruksi atau blok instruksi  
}  
else {  
    daftar instruksi/blok instruksi yang lain  
}
```

7.3. Instruksi for ...

Sebelum berbicara tentang instruksi, kita bicarakan dulu tentang *Loop*. Loop adalah struktur instruksi instruksi yang dapat di eksekusi berulang umang selama kondisi syaratnya belum terpenuhi. Cara yang paling umum dalam melakukan Loop adalah dengan menambahkan variabel penghitung/counter (variabel yang bertambah satu unit nilai selama satu kali Loop instruksi dijalankan (increment)), Loop akan berhenti jika variabel penghitung sudah melewati batas nilai tertentu yang dijadikan syarat.

For adalah salah satu Instruksi yang menggunakan fasilitas *Loop*. Dalam sintaksnya kita hanya perlu memasukkan nama variabel sebagai penghitung (dan juga nilai awalnya, serta kondisi dimana loop akan berhenti (pada dasarnya, kondisi dimana nilai penghitung melewati angka tertetu)), dan yang terakhir instruksi modifikasi penghitung, increment (naik per unit) atau decrement (turun per unit)

Sintaks lengkap dari instruksi ini adalah :

```
for (penghitung; kondisi loop berhenti; modfikasi penghitung) {  
    daftar instruksi instruksi atau blok instruksi  
}
```

Sebagai contoh :

```
for (i=1; i<6; i++) {  
    Alert(i)  
}
```

Loop ini akan 5 kali menampilkan nilai dari i, mulai dari i=1,2,3,4,5. Loop dimulai dari i=1 dan akan selalu melakukan cek dan verifikasi apakah nilai i kurang dari 6. Sampai pada i=6, dimana kondisi syaratnya sudah tidak terpenuhi maka loop akan berhenti.

7.4 Instruksi while

Instruksi while merupakan salah satu cara alternatif untuk menjalankan sekumpulan instruksi, seperti juga instruksi For..

Sintaks dari instruksi ini adalah sebagai berikut :

```
while (kondisi syarat terpenuhi) {  
  
daftar instruksi instruksi atau blok instruksi  
  
}
```



Karena instruksi ini menjalankan program selama kondisi syarat terpenuhi, maka perlu diperhatikan baik baik syarat yang kita berikan, agar supaya instruksi tidak menjadi loop tanpa henti (infinity) dan membuat error navigator kita.

7.5 Instruksi continue

Ada hal yang patut di perhatikan juga, ada kalanya kita perlu melakukan lompatan (jump) terhadap satu atau beberapa nilai tertentu di dalam loop tanpa menghentikan loop itu sendiri. Sintaks yang digunakan disini adalah continue , dan di letakkan di dalam loop itu sendiri, pada umumnya kita tambahkan juga struktur kondisional sebagai syarat supaya sintaks tersebut berjalan lancar.

Contoh : kita akan mencetak hasil dari persamaan $1/(x-7)$ untuk nilai $x = 1$ sampai $x = 10$, cukup jelas untuk nilai $x = 7$ maka akan menghasilkan error (pembagian dengan 0), berkat instruksi continue kita bisa memperlakukan secara terpisah nilai $x = 7$, dan meneruskan loop dari program tersebut.

```
x=1;  
while (x<=10) {  
    if (x == 7) {  
        Alert('pembagian oleh 0');  
        X++;  
        continue;  
    }  
    a = 1/(x-7);  
    Alert(a);  
    x++;  
}
```

7.6. Instruksi break

Sebaliknya kita juga bisa memaksa loop berhenti sebelum waktunya dengan alasan yang dikemukakan di bagian awal dari loop. Instruksi Break memungkinkan menghentikan suatu loop (baik untuk for ataupun while). Pemakaiannya sendiri seperti instruksi continue, yaitu penambahan struktur kondisional agar supaya loop berhenti dan tidak berulang ulang looping.

Contoh :

```
for (x=1; x<=10; x++) {  
    a = x-7;  
    if (a == 0) {  
        Alert('pembagian oleh 0');  
        break;  
    }  
    Alert(1/a);  
}
```

7.7 Instruksi switch case

Instruksi ini memungkinkan kita untuk melakukan test berbagai nilai dari variabel yang sama. Dengan cara ini kita bisa melakukan testing terhadap berbagai nilai variabel lebih sederhana daripada memakai instruksi if.

Sintaksnya adalah sebagai berikut :

```
switch (Variabel) {  
  
    case Nilai1:  
        blok instruksi;  
        break;  
  
    case Nilai2:  
        blok instruksi;  
        break;  
  
    case NilaiX:  
        blok instruksi;  
        break;  
  
    default:  
        blok instruksi;  
        break;  
  
}
```

Kata di dalam tanda kurung sesudah kata switch menunjukkan nama variabel yang akan di test pada kasus nilai yang berbeda. Pada saat nilai variabel yang akan di test sama dengan nilai kasusnya (case) maka blok instruksi di bawahnya akan dieksekusi. Kata break berarti berhentinya atau keluar dari struktur kondisi switch. Kata default berarti blok instruksi dibawahnya akan dieksekusi bila nilai variabel yang masuk tidak sama dengan semua kasus (case) yang ada.

8. Fungsi

Fungsi adalah subprogram yang memungkinkan kita untuk menjalankan sekelompok instruksi dengan satu pemanggilan sederhana nama fungsi tersebut dari satu atau beberapa bagian di dalam badan suatu program. Bentuk subprogram yang kita sebut fungsi ini sangat umum di pakai di banyak bahasa pemrograman (tentu saja dengan cara yang sedikit berbeda antara satu dengan lainnya). Di lain pihak suatu fungsi, juga bisa memanggil dirinya sendiri, ini kita sebut dengan fungsi rekursif (akan tetapi jangan lupa untuk meletakkan kondisi khusus supaya fungsi bisa berhenti, kalau tidak bisa membahayakan kelangsungan program secara global).

JavaScript sendiri mempunyai fungsi native (predefined) yang dapat diaplikasikan untuk satu atau banyak jenis obyek spesifik, kita sebut fungsi ini sebagai *metoda* (lihat bab 9)

8.1 Deklarasi fungsi

Sebelum digunakan, suatu fungsi harus di definisikan terlebih dahulu karena untuk memanggil fungsi tersebut dari dalam program, navigator harus mengenalinya terlebih dahulu, dalam artian mengenali namanya, argumennya, dan instruksi di dalamnya. Pendefinisian satu fungsi dinamakan *deklarasi*. Sintaks pendeklarasian suatu fungsi adalah sebagai berikut :

```
function Nama_dari_Fungsi(argumen1, argumen2, ...) {  
    daftar instruksi atau blok instruksi  
}
```

Catatan :

- Kata kunci *function* diikuti nama dari fungsi tersebut.
- Penamaan dari fungsi mempunyai aturan yang sama dengan penamaan dari variabel
 - Nama harus dimulai oleh huruf.
 - Nama dari fungsi bisa berisi huruf, angka atau karakter `_` dan `&`, karakter kosong atau spasi tidak diperbolehkan.
 - Nama fungsi seperti juga nama variabel adalah *case sensitive*, jadi memperhatikan huruf besar dan huruf kecilnya.
- Argumen adalah optional, boleh ada dan boleh tidak ada, jika argumen tidak ada maka tanda kurung harus tetap ditampilkan.

8.2 Pemanggilan fungsi

Untuk mengeksekusi satu fungsi, kita cukup memanggil nama dari fungsi tersebut yang diikuti dengan kurung buka, argumen(jika ada) dan di tutup dengan kurung tutup.

```
Nama_dari_Fungsi() ;
```

Catatan :

- Titik koma memberikan tanda kepada navigator tentang akhir dari blok instruksi yang berbeda.
- Jika anda mendefinisikan suatu argumen untuk suatu fungsi, maka pada saat pemanggilannya anda harus memanggil fungsi tersebut lengkap dengan argumentnya.



Perlu diperhatikan untuk selalu mendeklarasikan suatu fungsi sebelum kita panggil, karena navigator selalu membaca HTML dan skrip dari atas ke bawah, oleh karena itu pada umumnya fungsi di dalam tag **SCRIPT** terlatak di bagian kepala dari dokumen HTML (diantara tag **<HEAD>** dan **</HEAD>**)

Berkat event administrator yang bernama **onLoad** (di letakkan di dalam tag **BODY**), kita bisa memanggil fungsi pada saat loading halaman dokumen HTML, sebagai contoh inialisasi nilai awal dari skrip anda atau melakukan test apakah navigator dapat menjalankan fungsi dari skrip atau tidak. Contohnya adalah sebagai berikut :

```
<HTML>
<HEAD>
<SCRIPT language="Javascript">

<!--
function Pemanggilan() {
blok instruksi
}
//-->

</SCRIPT>
</HEAD>

<BODY onLoad="Pemanggilan();" >

.....

</BODY>
</HTML>
```

8.3 Parameter dari fungsi

Kita bisa melewati parameter di dalam suatu fungsi, dalam artian kita berikan nilai atau nama variabel supaya fungsi itu bisa di eksekusi berdasarkan parameter tersebut. Pada saat kita melewati beberapa parameter ke dalam fungsi, parameter parameter tersebut dipisahkan oleh tanda koma, baik pada saat deklarasi ataupun pada saat pemanggilan. Kalau anda melihat bab bab sebelumnya parameter ini kita sebut argumen fungsi secara umum.

Kita lihat contoh di bawah ini : kita akan membuat program JavaScript yang menampilkan kotak dialog :

```
<HTML>
<HEAD>

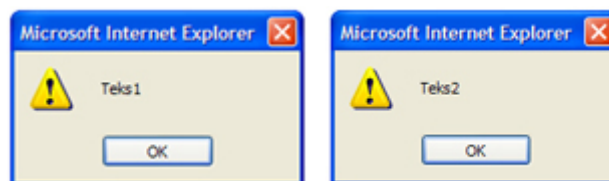
<SCRIPT language="Javascript">
<!--
function Tampilkan1() {
    alert('Teks 1');
}
-->
```

```
function Tampilkan2() {  
    alert('Teks 2');  
}  
//-->  
</SCRIPT>  
  
</HEAD>  
<BODY>  
  
<A href="javascript:;" onClick="Tampilkan1();">Teks1</A>  
<A href="javascript:;" onClick="Tampilkan2();">Teks2</A>  
  
</BODY>  
</HTML>
```

atau cara kedua berikut akan memberikan hasil yang sama :

```
<HTML>  
<HEAD>  
  
<SCRIPT language="Javascript">  
<!--  
function Tampilkan(Teks) {  
    alert(Teks);  
}  
//-->  
</SCRIPT>  
  
</HEAD>  
<BODY>  
  
<A href="javascript:;" onClick="Tampilkan('Teks1');">Teks1</A>  
<A href="javascript:;" onClick="Tampilkan('Teks2');">Teks2</A>  
  
</BODY>  
</HTML>
```

Hasil akhir dari kedua program itu sama saja, akan tetapi program kedua lebih fleksibel karena kita cuman punya satu fungsi yang bisa menampilkan sembarang teks.



8.4 Bekerja dengan variabel di dalam fungsi

Secara logika pada saat kita selesai memanipulasi variabel di dalam fungsi, dan kemudian kita keluar dari fungsi tersebut, maka nilai variabel itu akan kembali ke nilai asalnya, meskipun kita sudah merubahnya di dalam fungsi tersebut.

Akan tetapi ini semua bergantung dari jenis variabel itu sendiri, apakah dia variabel lokal atau variabel global :

- Variabel yang dideklarasikan secara implisit di dalam fungsi (tanpa kata kunci var) akan menjadi global, yang artinya variabel masih bisa di akses sesudah eksekusi dari fungsi.
- Variabel yang dideklarasikan secara eksplisit di dalam fungsi (menggunakan kata kunci var) akan menjadi lokal, yang artinya hanya dapat diakses dari dalam fungsi, semua referensi yang memakai variabel ini dari luar fungsi akan menyebabkan pesan error (variabel tidak dikenal).

Pada saat kita memanggil satu fungsi dari satu obyek, sebagai contoh misalkan suatu form, maka sebaiknya kita menggunakan kata kunci **this** untuk membuat satu referensi dengan obyek yang sedang berjalan/sedang di kerjakan. Dengan kata kunci ini kita juga menghindarkan dari menulis format obyek secara bertele tele seperti *windows.objet1.objet2....* dan juga pada saat ingin melewati obyek yang sedang di proses ke satu fungsi , kita tinggal menulis *Nama_dari_Fungsi(this)* . untuk bisa memanipulasi obyek tersebut dari dalam fungsi. Sedangkan untuk memanipulasi kondisi dari obyek itu sendiri, kita hanya tinggal mengetikkan *this.property*

8.5 Mendefinisikan obyek dengan fungsi

Kita bisa membuat suatu obyek dengan properti yang kita definisikan. Sebagai contoh kita bayangkan kita akan membuat satu obyek pohon dengan properti sebagai berikut :

- Jenis
- Tinggi
- Umur

Kita bisa membuat obyek pohon dengan menggunakan satu fungsi yang kita beri nama fungsi pohon, dimana properti dari obyek pohon didefinisikan di fungsi tersebut. Kita juga menggunakan kata kunci **this** dalam contoh yang detailnya dapat anda lihat berikut ini :

Umur

```
function Pohon(Jenis, Tinggi, Umur) {  
  
this.Jenis = Jenis;  
this.Tinggi = Tinggi;  
this.Umur = Umur;  
  
}
```

Selanjutnya kita gunakan kata kunci **new** kita bisa membuat instan dari obyek pohon (maksudnya satu obyek yang lain dengan properti sama dengan obyek pohon) :

```
Pohon1 = new Pohon("cemara", 14, 20)
```

Kita bisa membuat sebanyak mungkin instan yang kita inginkan. Selain itu di obyek yang sudah ada kita bisa menambahkan properti dari obyek yang lain, mari kita lihat contoh berikut ini :

```
function Pemilik>Nama, Jenis_Kelamin) {  
  
this>Nama =>Nama;  
this.Jenis_Kelamin = Jenis_Kelamin;  
  
}
```

seterusnya kita definisikan obyek pohon sebagai berikut :

```
function Pohon(Jenis, Tinggi, Umur, Pemilik) {  
  
    this.Jenis = Jenis;  
    this.Tinggi = Tinggi;  
    this.Umur = Umur;  
    this.Pemilik = Pemilik;  
  
}
```

selanjutnya kita bisa tulis nama pemilik dari pohon tersebut sebagai berikut :

```
Pohon.Pemilik>Nama
```

dan juga merubah propertinya seperti berikut ini :

```
Pohon.Pemilik>Nama = Andry;  
Pohon.Pemilik.Jenis_Kelamin = Pria;
```

9. Metoda

Metoda adalah suatu fungsi yang diasosiasikan dengan satu obyek , satu aksi yang bisa kita eksekusi pada satu obyek. Metoda pada obyek dari navigator adalah fungsi fungsi yang sudah di definisikan sebelumnya (predefined) berdasarkan aturan aturan HTML dan kita tidak bisa memodifikasinya. Akan tetapi bisa membuat metoda yang personalisasikan sendiri untuk setiap obyek yang kita buat. Mari kita lihat contoh tentang dokumen HTML, yang terdiri dari obyek yang bernama **document**, obyek ini mempunyai metoda yang bernama **write()** yang berguna untuk memodifikasi isi dari dokumen HTML dengan menampilkan teks tertentu. Maka metoda itu akan di panggil dengan dengan cara berikut :

```
window.document.write()
```

9.1 Metoda *write*

Metoda write() dari satu obyek dokumen memungkinkan kita untuk memodifikasi secara dinamik isi dokumen HTML. Berikut ini adalah sintaks secara umum dari metoda write :

```
window.document.write(ekspresi1, ekspresi2, ...)
```

secara praktisnya kita bisa tulis metoda write() sebagai berikut :

- Menuliskan secara langsung teks ke dalam parameter :
`document.write("hallo");`
yang akan mempunyai hasil menambahkan (concatenate) string bertuliskan hallo ke tempat dimana kita meletakkan skrip tersebut.
- Bisa juga melewati teks tersebut melalui suatu variabel :
`test='hallo';`

```
document.write(test);
```

mempunyai efek yang sama dengan cara sebelumnya

- Atau dengan menggabungkan kedua cara diatas :

```
test='hallo';  
document.write('dia berkata' + test);
```

yang akan mempunyai hasil menambahkan kata *hallo* di belakang kata *dia berkata* di dalam dokumen HTML
- Atau bisa juga dengan menyisipkan langsung satu ekspresi(ekspression) , yang akan di evaluasi dan di jalankan segera mungkin, dan hasilnya langsung ditampilkan

```
test='akar dari bilangan 2 adalah : '  
document.write(test+sqrt(2));
```

kita juga bisa memasukkan tag HTML kedalam metoda write :

```
document.write('<font color="#FF0000">Hallo</font>');
```

9.2 Metoda *writeln*

Metoda `writeln()` berfungsi sama persis seperti metoda `write` dengan penambahan pemindahan ke baris baru setiap kali usai menuliskan metodanya. Akan tetapi dokumen HTML tidak mengenal adanya penambahan baris baru dengan cara tradisional, penambahan baris baru hanya bisa dilakukan dengan menggunakan tag `
`, oleh karena itu metoda ini tidak terlalu berguna di dalam HTML, kecuali diantara tag `<PRE>` dan `</PRE>` dimana kita memperlakukan teks didalamnya seperti file teks biasa.

9.3 Mendefinisikan satu metoda untuk sebuah obyek

Seperti disebutkan diatas kita bisa membuat satu metoda untuk satu jenis obyek yang kita buat sendiri, dengan menggunakan fungsi sebagai properti dari satu obyek, mari kita lihat kembali contoh tentang Pohon :

Pertama kita definisikan satu properti :

```
function Pemilik>Nama, Jenis_Kelamin) {  
  
    Pemilik>Nama = Nama;  
    Pemilik.Jenis_Kelamin = Jenis_Kelamin;  
  
}
```

Kemudian kita buat satu fungsi yang akan menampilkan nama dan jenis kelamin dari pemilik pohon beserta karakteristik lainnya :

```
function TampilkanInfo() {  
  
    alert(this.Pemilik>Nama+ 'berjenis kelamin' + this.Pemilik.Jenis_Kelamin + 'mempunyai sebuah pohon' + this.Jenis);  
  
}
```

Sekarang kita tinggal mendefinisikan obyek Pohon sebagai berikut

```
function Pohon(Jenis, Tinggi, Umur, Pemilik, TampilkanInfo) {  
  
    this.Jenis = Jenis;  
    this.Tinggi = Tinggi;  
    this.Umur = Umur;  
    this.Pemilik = Pemilik;  
    this.TampilkanInfo = TampilkanInfo;  
  
}
```

Secara singkat kita lihat hasilnya, misalkan kita ketik *Pohon1.TampilkanInfo()* maka kita akan mendapatkan hasil sebagai berikut (sekedar contoh) :

Andry berjenis kelamin laki laki mempunyai sebuah pohon cemara

Kita lihat contoh tersebut metode *TampilkanInfo()* diaplikasikan ke obyek *Pohon1*

10. Kotak Dialog

Kotak dialog adalah satu jendela yang tampil di bagian depan (layer paling atas) menyusul satu event yang di jalankan, dan memungkinkan kita untuk :

- Memberikan peringatan kepada user
- Memberikan pilihan yang harus dipilih oleh user
- Meminta user untuk mengisi atau melengkapi isian pada suatu form field.

JavaScript menawarkan 3 metoda penggunaan kotak dialog berdasarkan ke tiga fungsi yang sudah disebut diatas. Metoda metoda itu adalah metoda dari obyek *window*.

10.1 Metoda *alert()*

Metoda *alert()* memungkinkan navigator untuk menampilkan satu kotak dialog yang berisi satu tombol *OK* dan teks keterangan (sebagai satu satunya parameter dari metoda). Pada saat kotak ini muncul, user tidak punya pilihan lain selain menekan tombol *OK*.

Sintaksnya adalah sebagai berikut :

```
alert(nama_dari_variabel);  
alert('teks');  
alert('teks' + nama_dari_variabel);
```

dan tampilan hasil dari metoda *alert()* adalah sebagai berikut (contoh menggunakan navigator Internet Explorer)



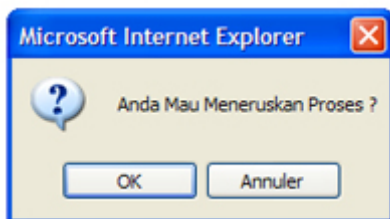
dalam hal ini sintaksnya adalah :

```
alert('Perhatian');
```

10.2 Metoda *confirm()*

Metoda *confirm()* sama seperti metoda *alert* , dengan tambahan kita bisa melakukan pilihan *OK* atau *Cancel*. Pada saat kita pilih *OK* maka metoda ini akan mengirimkan nilai *true* dan mengirimkan nilai *false* jika kita memilih *Cancel*

Tampilan kotak dialog di navigator internet explorer

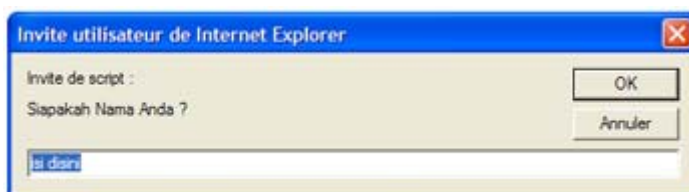


sedangkan sintaksnya sendiri sama seperti metoda *alert*, dalam contoh ini sintaksnya adalah :

```
confirm('Anda Mau Meneruskan Proses ?');
```

10.3 Metoda *prompt()*

Metoda *prompt()* agak sedikit lebih canggih dibandingkan kedua metoda sebelumnya karena dia dilengkapi dengan satu cara untuk mendapatkan informasi yang diberikan oleh user. Metoda *prompt()* terdiri dari 2 komponen, yang pertama teks untuk pertanyaan dan yang kedua adalah teks default dari field yang akan diisi informasi oleh user. Berikut ini contoh dari metoda *prompt()* menggunakan navigator Internet Explorer versi Prancis :D.



Sintaksnya adalah sebagai berikut :

```
prompt('Siapakah Nama Anda ?', 'isi disini');
```

metoda ini akan mengembalikan nilai teks yang diisi oleh user, dan akan mengirimkan nilai *null* jika user tidak mengisi field itu.

REFERENSI :

1. Buku : JavaScript Pocket Reference, 2nd Edition, Pengarang David Flanagan, Penerbit O'Reilly
2. Buku : JavaScript le poche, Pengarang Cedric Nilly, Jean Christophe Gigniac, Penerbit Eyrolles
3. Site Internet : <http://www.javascript.com>
4. Site Internet : <http://javascript.internet.com>

BIOGRAFI PENULIS



Andry Alamsyah. Lahir di Malang, Menamatkan SMU di SMUN 3 Malang. Setelah itu melanjutkan pendidikannya di Jurusan Matematika ITB Bandung dan lulus S1 pada tahun 1996. Setelah lulus bekerja di salah satu anak perusahaan Astra di bidang IT di Jakarta selama 5 tahun sebagai System Analyst. Pada Tahun 2001 meneruskan pendidikan S2 di Bidang Sistem Informasi Multimedia di Universitas Picardie, Amiens, Prancis. Saat ini sambil meneruskan kuliah, juga bekerja sebagai konsultan dan programmer spesialis aplikasi web di Bengkel Pengembangan Multimedia yang bergerak di bidang konsultasi dan pengembangan sistem belajar jarak jauh untuk beberapa universitas di Prancis, Hungaria, Aljazair, Tunisia dan Maroko

Informasi lebih lanjut tentang penulis ini bisa didapat melalui:

Email: alamsyah@tiscali.fr

URL: <http://www.alamsyah.net>