Javascript

Pengenalan

Bahasa script Javascript paling populer di Internet dan dapat bekerja pada semua browser utama seperti Internet Explorer, Firefox, Google Chrome, Safari, dan Opera.
Untuk mempelajari Javascript dibutuhkan penguasaan terhadap bahasa utama web yaitu HTML atau XHTML.

Sebelum mengenali lebih jauh, berikut beberapa hal yang perlu diketahui tentang javascript:

- Javascript dirancang untuk memberikan unsur interaktif kepada halaman HTML
- Javascript merupakan bahasa script. Bahasa script adalah bahasa pemrograman yang ringan, dalam arti aturan pemrogramannya tidak serumit bahasa pemrograman murni
- Javascript biasanya digunakan dengan cara menuliskannya secara langsung pada halaman HTML
- Javascript merupakan bahasa interpreter, yang berarti script Javascript tidak perlu dikompilasi sebelum dieksekusi.
- Javascript bersifat gratis

Javascript dapat digunakan untuk melakukan hal-hal berikut:

- Javascript dapat membuat teks dinamis dalam halaman HTML. Javascript dapat menulis teks atau isi suatu variabel pada halaman HTML.
- Javascript dapat menangani event. Kejadian-kejadian seperti tindakan mengklik mouse, atau ketika halaman HTML telah selesai dimuat ke browser dapat dikenali oleh javascript dan dapat mengatur tindakan tertentu ketika event itu terjadi.
- Javascript dapat membaca dan menulis elemen-elemen HTML, yang berarti dapat digunakan untuk mengubah konten suatu elemen HTML.
- Javascript dapat memvalidasi data, misalnya data form sebelum dikirim ke server sehingga dapat meringankan server.
- Javascript dapat mendeteksi browser pengunjung
- Javascript dapat membuat cookies yang akan menyimpan dan memanggil informasi kedalam komputer pengunjung.

Cara Menggunakan Javscript

Kode-kode Javascript disisipkan kedalam dokumen HTML menggunakan elemen SCRIPT. Elemen ini akan mengawali dan mengakhiri kode-kode javascript. Browser akan melihat elemen ini sebagai perintah kepadanya untuk menterjemahkan kode-kode yang berada didalam elemen SCRIPT sebagai script Javascript.
Contoh berikut akan memberikan gambaran bagaimana penulisan kode javascript kedalam dokumen HTML:

```
<html>
<body>
  <script type="text/javascript">
    document.write("Hello world!!");
  </script>
</body>
</html>
```

Dalam contoh diatas, elemen SCRIPT menggunakan atribut type untuk mendefinisikan bahasa script yang digunakan (bahasa script tidak hanya Javascript saja)
Jika akan menggunakan elemen HTML untuk memformat tampilan hasil kode Javascript, maka elemen HTML yang akan digunakan ditulis sebagai parameter dari perintah *document.write()*:

```
<html>
<body>
  <script type="text/javascript">
    document.write("<h1>Hello world!!</h1>");
  </script>
</body>
</html>
```

Perintah document.write() adalah perintah standar yang umum di Javascript untuk menampilkan keluaran ke halaman HTML.

Catatan : jika tag <*script*> tidak digunakan saat menuliskan kode Javascript kedalam dokumen HTML, maka browser akan menganggap kode tersebut sebagai teks murni dan akan menampilkan seluruh kode tersebut ke halaman HTML.

Browser Yang Tidak Mendukung Javascript

Suatu dokumen HTML yang terdapat kode javascript didalamnya dijalankan pada browser yang tidak mendukung Javascript, maka kode Javascript tidak akan dieksekusi malahan seluruh kodenya akan ditampilkan apa adanya ke halaman.
Saat ini, hampir seluruh browser mendukung Javascript. Meskipun demikian, untuk tindakan pencegahan, maka kode Javascript tersebut dapat diapit dengan elemen komentar HTML agar tidak ditampilkan kodenya oleh browser yang tidak mendukung Javascript. Berikut penulisannya:

```
<html>
<body>
  <script type="text/javascript">
    <!--
document.write(“Hello world!!”);
    //-->
  </script>
```

Dua slash pada tag penutup komentar merupakan simbol komentar Javascript yang akan mencegah Javascript untuk mengeksekusi tag ‘-->’.

Lokasi Penulisan Kode Javascript dalam Dokumen HTML

Kode Javascript yang terdapat dalam suatu dokumen HTML akan dieksekusi segera ketika halaman dimuat ke browser. Suatu saat, diinginkan agar kode Javascript dijalankan hanya ketika halaman selesai dimuat, atau ketika pengunjung mengklik tombol kiri mouse. Untuk melakukan ini, kode Javascript perlu ditulis sebagai sebuah fungsi.

Di Bagian HEAD

Penulisan kode Javascript di bagian HEAD dokumen HTML akan membuat kode tersebut tidak akan dieksekusi segera ketika halaman dimuat ke browser. Kode tersebut hanya akan dieksekusi ketika dipanggil atau ketika terjadi event tertentu.
Penulisan kode Javascript pada bagian HEAD ini merupakan cara yang baik untuk menggunakan Javascript sebab kode-kode Javascript terkumpul dalam satu lokasi penulisan dan tidak akan bercampur dengan kode konten halaman.

```
<html>
<head>
  <script text=”text/javascript”>
    function pesan()
    {
      alert(“Kotak pesan ini dipanggil dengan event onload”);
    }
  </script>
</head>
<body onload=”pesan()”>
</body>
</html>
```

Di Bagian BODY

Javascript yang diletakkan pada bagian body biasanya ditujukan untuk menampilkan konten saat halaman dimuat ke browser.

```
<html>
<head>
</head>
<body>
  <script text=”text/javascript”>
    document.write(“Pesan ini dibuat dengan Javascript”);
  </script>
</body>
</html>
```

Di Bagian HEAD dan BODY

Jumlah script yang dapat disisipkan pada dokumen HTML tidak terbatas. Script Javascript juga dapat ditulis pada bagian HEAD dan BODY berbarengan.

```
<html>
<head>
<script text=”text/javascript”>
      function pesan()
      {
         alert(“Kotak pesan ini dipanggil dengan event
onload”);
      }
</script>
</head>
<body onload=”pesan()”>
   <script text=”text/javascript”>
      document.write(“Pesan ini dibuat dengan Javascript”);
   </script>
</body>
</html>
```

Menggunakan File Javascript Eksternal

Selain menuliskan secara langsung pada dokumen HTML, penerapan Javascript juga dapat menggunakan file eksternal. Kode-kode Javascript dapat disimpan pada sebuah file dengan ekstension ‘.js’. Kemudian file eksternal ini dapat dipanggil dari dokumen HTML dengan menggunakan atribut ‘src’ dari tag <SCRIPT>. File eksternal ini dapat digunakan oleh banyak dokumen HTML.

Perlu dicatat, bahwa kode-kode Javascript yang ditulis pada file eksternal tidak boleh menggunakan elemen <SCRIPT>!

dokumen.html

```
<html>
<head>
   <script type=”” src=”file_eksternal.js”></script>
</head>
<body onload=”pesan()”>
</body>
</html>
```

file_eksternal.js

```
function pesan()
{
   alert(‘Kode kotak pesan ini disimpan dalam file
eksternal’);
}
```

Pada contoh diatas, dokumen HTML dan file eksternal Javascript diletakkan pada direktori yang sama. Jika tidak, maka pada atribut ‘src’ harus dilengkapi *path* lokasi penyimpanan file eksternal javascript relatif terhadap dokumen HTML. Misalnya, jika dokumen HTML diletakkan pada direktori *root* dan file eksternal Javascript pada direktori *javascript* maka nilai ‘src’ menjadi ‘javascript/file_eksternal.js’.

Statement Javascript

Case Sensitive dalam Statement

Javascript memiliki statement-statement. Rangkaian statement Javascript akan dieksekusi oleh browser. Berbeda dengan HTML, statement Javascript bersifat *case sensitive*. Oleh karena itu, penulisan statement perlu diperhatikan benar. Hal ini berlaku ketika membuat statement Javascript, membuat atau memanggil variabel, objek, dan fungsi.

Sebuah statement Javascript merupakan suatu perintah kepada browser yang bertujuan untuk memberitahu browser apa yang harus dilakukan. Misalnya, suatu statement Javascript yang mengatakan kepada browser untuk menampilkan kalimat 'Hello world' berupa seperti berikut:

```
document.write("Hellow world");
```

Tanda Akhir Suatu Statement

Setiap statement Javascript akan diakhiri dengan tanda *semicolon*. Sebagian orang berpendapat bahwa hal ini merupakan suatu praktek pemrograman yang baik. Tetapi aturan ini bersifat opsional dalam Javascript. Jika tidak menggunakannya, maka browser akan menganggap bahwa akhir dari sebuah baris adalah akhir dari satu statement. Maka jika akan menulis beberapa statement dalam satu baris, gunakanlah tanda *semicolon* ini.

Kode Javascript

Kode javascript (disingkat Javascript) adalah sebuah rangkaian statement-statement Javascript. Setiap statement dieksekusi oleh browser dalam urutan sesuai penulisannya.

Blok Javascript

Statement-statement Javascript dalam dikelompokkan dalam satu kelompok. Awal blok dimulai dengan tanda kurung kurawal, demikian juga akhir blok. Kegunaan dari blok adalah untuk membuat rangkaian statement dieksekusi bersama.

Menyisipkan Komentar

Suatu komentar biasanya disisipkan oleh programmer untuk menjelaskan kode program. Javascript menggunakan tanda *backslash* sebanyak dua untuk menandai komentar. Contoh:

```
<script type="text/javascript">
  // Buat Heading
  document.write("<h1>This is a heading</h1>");
  // Tulis 2 paragrap
  document.write("<p>This is a paragraph.</p>");
  document.write("<p>This is another paragraph.</p>");
</script>
```

Untuk menyisipkan komentar yang panjangnya melebihi satu baris, dapat menggunakan tanda '/*' untuk awalan dan sebagai akhiran '*/'. Contoh:

```
<script type="text/javascript">
  /*
  Kode dibawah ini akan membuat
  heading dan dua buah paragrap
  */
  document.write("<h1>This is a heading</h1>");
  document.write("<p>This is a paragraph.</p>");
  document.write("<p>This is another paragraph.</p>");
</script>
```

Komentar juga dapat ditulis di akhir statement, contoh:

```
<script type="text/javascript">
  document.write("Hello");       // mencetakWrite "Hello"
  document.write(" Nafilla!");   // mencetak " Nafilla!"
</script>
```

Variabel

Dalam pemrograman variabel dipakai untuk menampung nilai data. Nilai yang disimpan dalam variabel ini dapat berubah sepanjang eksekusi program.
Aturan dalam pembuatan variabel di Javascript cukup sederhana, yaitu:

- Nama variabel bersifat *case sensitive* (variabel Y berbeda dengan variabel y)
- Nama variabel harus dimulai dengan huruf atau garisbawah (*underscore*)

Mendeklarasikan variabel

Pembuatan variabel di Javascript sering disebut sebagai mendeklarasikan variabel. Keyword yang diperlukan untuk mendeklarasikan variabel adalah *var*. Contoh:

```
var x;
var nama;
```

Pendeklarasian diatas akan membuat variabel kosong. Variabel juga dapat dideklarasikan berikut nilainya seperti statement berikut:

```
var x=10;
var nama=”Javas”;
```

Kalau nilai yang akan dimasukkan kedalam variabel berupa string, maka tanda petik diperlukan untuk mengurung nilai, seperti contoh nilai string ‘Javas’ diatas (tanda petik tunggal dan ganda dapat digunakan secara bergantian).

Pendeklarasian kembali suatu variabel tidak akan menghilangkan nilai data yang telah tersimpan didalamnya, contoh:

```
var x=5;
var x;
```

nilai data ‘5’ dalam variabel x tidak akan hilang setelah statement baris dibawahnya dieksekusi.

Operator

Operator Aritmetika

Operator	Description	Example	Result	
+	Addition	x=y+2	x=7	y=5
-	Subtraction	x=y-2	x=3	y=5
*	Multiplication	x=y*2	x=10	y=5
/	Division	x=y/2	x=2.5	y=5
%	Modulus (division remainder)	x=y%2	x=1	y=5
++	Increment	x=++y	x=6	y=6
		x=y++	x=5	y=6
--	Decrement	x=--y	x=4	y=4
		x=y--	x=5	y=4

Operator Assignment

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

Operator + Untuk String

Operator + digunakan juga untuk menggabungkan 2 atau lebih string menjadi satu.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

Setelah diekseksusi variabel txt3 berisi “What a verynice day”.
Untuk menambahkan spasi diantara dua string dapat dilakukan dengan cara menambahkan pada salah satu string:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

atau tambahkan pada ekspresi:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+"" ""txt2;
```

Setelah dieksekusi pernyataan diatas, maka variabel txt3 berisi “What a very nice day”. Jika yang akan ditambahkan adalah string dan bilangan maka hasilnya berupa string.

Operator Pembandingan dan Logika

Operator logika dan pembandingan digunakan untuk menguji yang hasilnya berupa nilai true atau false.

Operator Pembandingan

Operator ini digunakan dalam pernyataan-pernyataan logik untuk menentukan persamaan atau perbedaan diantara variabel-variabel atau nilai-nilai. Tabel dibawah ini berdasarkan nilai x=5.

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

Operator pembandingan dalam penerapannya juga digunakan dalam pernyataan-pernyataan kondisional untuk membandingkan nilai dan aksi berikutnya tergantung pada hasil pembandingan. Contoh:

```
if (usia<17) document.write(“Swet seventeen”);
```

Operator Logika

Operator yang digunakan untuk menentukan nilai logika diantara dua variabel atau nilai. Tabel dibawah ini berdasarkan nilai x=6 dan y=3.

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Javascript juga mempunyai operator kondisional yang berguna untuk menempatkan suatu nilai ke suatu variabel berdasarkan suatu kondisi. Sintaks penggunaan operator ini adalah:

nmVar=(kondisi)?nilai1:nilai2;

Contoh:
salam=(waktu==”PAGI”)?”Selamat pagi ”:”Selamat datang ”;

Kondisi Percabangan

Pada saat membuat program, acapkali kita menginginkan kode program melakukan operasi yang berbeda berdasarkan keputusan yang berbeda. Disini kita dapat menggunakan kondisi percabangan untuk melakukannya.
Javascript menyediakan beberapa jenis kondisi percabangan:

- **Percabangan if**, dipakai untuk menjalankan kode tertentu hanya jika kondisi true terpenuhi.
  Sintaksnya adalah:

```
if (condition)
        {
        code to be executed if condition is true
        }
```

  Contoh:

```
<script type="text/javascript">
//Cetak salam "Good morning" jika waktu kurang dari pukul 10
var d=new Date();
var time=d.getHours();
if (time<10)
        {
        document.write("<b>Good morning</b>");
        }
</script>
```

- **Percabangan if...else**, dipakai jika terdapat dua kode yang akan dijalankan apabila kondisi true terpenuhi dan tidak terpenuhi.
  Sintaksnya:

```
if (condition)
        {
        code to be executed if condition is true
        }
else
        {
        code to be executed if condition is not true
        }
```

  Contoh:

```
<script type="text/javascript">
//Jika waktu kurang dari pukul 10, cetak salam "Good morning"
//Selain itu cetak salam "Good day"
```

```
var d = new Date();
var time = d.getHours();
if (time < 10)
        {
        document.write("Good morning!");
        }
else
        {
        document.write("Good day!");
        }
</script>
```

- **Percabangan if...else if..else**, dipakai jika terdapat banyak pilihan blok kode yang akan dijalankan
  Sintaks:

```
if (condition1)
        {
        code to be executed if condition1 is true
        }
else if (condition2)
        {
        code to be executed if condition2 is true
        }
else
        {
        code to be executed if condition1 and condition2 are not true
        }
```

Contoh:

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
        {
        document.write("<b>Good morning</b>");
        }
else if (time>10 && time<16)
        {
        document.write("<b>Good day</b>");
        }
else
        {
        document.write("<b>Hello World!</b>");
        }
```

```
</script>
```

- **Percabangan switch**, dipakai bila akan memilih salah satu dari sekian banyak blok kode untuk dijalankan
  Sintaks:

```
switch(n)
{
case 1:
        execute code block 1
        break;
case 2:
        execute code block 2
        break;
default:
        code to be executed if n is different from case 1 and 2
}
```

Switch bekerja sebagai berikut, ekspresi *n* yang biasanya berupa variabel, dievaluasi sekali. Kemudian nilai ekspresi tersebut akan dibandingkan dengan nilai dari setiap *case* pada struktur. Jika nilai salah satu *case* cocok dengan nilai ekspresi, maka kode dari *case* tersebut akan dieksekusi. Pernyataan break berfungsi untuk mencegah kode-kode pada *case* selanjutnya dieksekusi.

Contoh:

```
<script type="text/javascript">
//Cetak berbagai salam berdasarkan hari. Untuk hari Sunday=0, Monday=1,
//Tuesday=2, dst.
var d=new Date();
var theDay=d.getDay();
switch (theDay)
{
case 5:
        document.write("Finally Friday");
        break;
case 6:
        document.write("Super Saturday");
        break;
case 0:
        document.write("Sleepy Sunday");
        break;
default:
        document.write("I'm looking forward to this weekend!");
}
</script>
```

Popup Boxes

Javascript memiliki 3 jenis kotak dialog popup:

- **alert**, kotak dialog alert sering digunakan untuk menyampaikan informasi kepada user. Kotak dialog memiliki sebuah tombol, yaitu OK.
  Sintaks:

```
alert(“teks”);
```

  Contoh:

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
        alert("I am an alert box!");
}
</script>
</head>
<body>
<input type="button" onclick="show_alert()" value="Show alert box" />
</body>
</html>
```

- **confirm**, kotak dialog confirm biasanya digunakan jika kita menghendaki user memverifikasi atau menerima sesuatu. Kotak dialog memiliki 2 tombol, yaitu OK dan Cancel. Jika user mengklik tombol OK, maka kotak dialog akan mengembalikan nilai *true*, dan jika tombol *Cancel* yang diklik, maka nilai yang dikembalikan *false*.
  Sintaks:

```
confirm(“teks”);
```

  Contoh:

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
        {
                alert("You pressed OK!");
```

```
        }
        else
        {
                alert("You pressed Cancel!");
        }
}
</script>
</head>
<body>
<input type="button" onclick="show_confirm()" value="Show confirm box" />
</body>
</html>
```

- **prompt**, kotak dialog digunakan untuk meminta user memasukkan sebuah nilai sebelum masuk ke sebuah halaman. Kotak dialog memiliki 2 tombol, yaitu *OK* dan *Cancel*. Jika tombol *OK* diklik, maka nilai yang diinputkan akan dikembalikan oleh kotak dialog. Jika user mengklik tombol *Cancel*, maka niai *null* akan dikembalikan.
  Sintaks:

```
prompt("teks","nilai_default");
```

  Contoh:

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
    {
        document.write("Hello " + name + "! How are you today?");
    }
}
</script>
</head>
<body>
<input type="button" onclick="show_prompt()" value="Show prompt box" />
</body>
</html>
```

Fungsi

Kode javascript pada suatu halaman secara normal akan dieksekusi segera setelah halaman tersebut dimuat ke browser. Untuk mencegah hal ini, kode dapat dimasukkan kedalam sebuah fungsi. Fungsi adalah sekelompok kode yang dapat dieksekusi dengan pemanggilan fungsi atau even tertentu. Pemanggila fungsi dapat dilakukan darimana saja didalam halaman. Bahkan fungsi dapat dipanggil oleh halaman lain jika fungsi tersebut ditempatkan dalam suatu file eksternal *.js*.
Fungsi dapat diletakkan pada bagian *head* maupun *body*. Tetapi, untuk memastikan bahwa fungsi sudah siap atau sudah dimuat oleh browser saat akan dipanggil, sebaiknya fungsi diletakkan pada bagian *head*.
Untuk mendefinisikan sebuah fungsi, sintaks yang digunakan:

```
function functionname(var1,var2,...,varX)
{
        some code
}
```

Parameter *var1, var2,* adalah variabel atau nilai yang dilewatkan kedalam fungsi. Simbol kurung kurawal buka dan tutup menandakan awal dan akhir dari fungsi.

Penting!
Sebuah fungsi dengan tanpa parameter tetap harus menyertakan kurung buka dan tutup setelah nama fungsi. Dan seperti seharusnya, tulisan *function* menggunakan huruf kecil serta saat pemanggilan fungsi perhatikan penulisan nama fungsi harus sama dengan nama fungsi saat didefinisikan.

Contoh fungsi:

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
        alert("Hello World!");
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```

Pernyataan return

Sebuah fungsi jika dikehendaki dapat dibuat agar mengembalikan sebuah nilai. Untuk membuat fungsi seperti ini, maka pernyataan return dapat digunakan didalam tubuh fungsi. Sebagai contoh, fungsi dibawah ini akan mengembalikan nilai berupa perkalian nilai parameter-parameternya:

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
        return a*b;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3));
</script>
</body>
</html>
```

Variabel Lokal

Jika suatu variabel dideklarasikan dengan menggunakan pernyataan *var* didalam sebuah fungsi, variabel tersebut hanya dapat diakses didalam fungsi. Jika keluar dari fungsi, variabel tersebut akan dihapus. Variabel ini disebut variabel lokal. Variabel lokal dengan nama yang sama dapat dibuat pada dua fungsi yang berbeda, sebab variabel-variabel tersebut hanya dikenali didalam fungsi dimana mereka dideklarasikan.

Jika suatu variabel dideklarasikan diluar fungsi, semua fungsi pada halaman dapat mengaksesnya. Variabel ini akan ada mulai saat mereka dideklarasikan sampai dengan saat halaman ditutup.

Perulangan

Ketika sedang memprogram, suatu saat kita ingin suatu blok kode dieksekusi berkali-kali. Javascript menyediakan 2 jenis bentuk perulangan yang dapat digunakan untuk melakukan hal tersebut.

- **For,** perulangan for digunakan jika jumlah perulangan yang diinginkan diketahui. Sintaks:

```
for
(variable=startvalue;variable<=endvalue;variable=variable+increment)
{
        code to be executed
}
```

Contoh:

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
        document.write("The number is " + i);
        document.write("<br />");
}
</script>
</body>
</html>
```

Contoh diatas menentukan perulangan yang dimulai dengan nilai i=0. Perulangan akan terus dilakukan selama nilai i kurang dari atau sama dengan 5. Setiap satu iterasi selesai dilakukan, nilai i akan ditambah 1.

- **While,** perulangan while akan mengulang blok kode selama kondisi perulangan terpenuhi. Sintaks:

```
while (variable<=endvalue)
{
code to be executed
}
```

Contoh:

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
        {
        document.write("The number is " + i);
        document.write("<br />");
        i++;
        }
</script>
</body>
</html>
```

Program diatas akan melakukan perulangan selama nilai variabel *i* kurang dari sama dengan 5. Perulangan diawali dengan pemberian nilai 0 ke variabel *i*. Nilai *i* akan bertambah 1 setiap satu perulangan berjalan.

- **Do...while**, merupakan varian dari perulangan *while*. Keunikan perulangan ini yaitu perulangan pasti akan melakukan perulangan minimal 1 kali apapun kondisinya kemudian perulangan berikutnya berlangsung hingga kondisi terpenuhi. Sintaks:

```
do
{
code to be executed
}
while (variable<=endvalue);
```

Contoh:

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
        {
        document.write("The number is " + i);
        document.write("<br />");
        i++;
        }
while (i<=5);
</script>
</body>
</html>
```

Pernyataan Break dan Continue

Break

Suatu perulangan dapat dihentikan sebelum selesai melakukan seluruh perulangannya dengan suatu pernyataan break. Jika pada saat dieksekusi program menemui pernyataan break ini, maka perulangan pada saat itu akan dihentikan dan program akan mengeksekusi baris-baris kode setelah blok perulangan jika ada. Pernyataan ini akan menyebabkan kode-kode perulangan yang berada dibawah pernyataan break tidak akan dieksekusi. Contoh:

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
        {
        if (i==3)
                {
                break;
                }
        document.write("The number is " + i);
        document.write("<br />");
        }
</script>
</body>
</html>
```

Continue

Hampir sama seperti break, pernyataan continue akan menghentikan perulangan, tetapi bedanya setelah itu program akan mengeksekusi perulangan berikutnya. Seperti break, baris kode setelah continue dalam blok perulangan juga tidak akan dieksekusi. Contoh:

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
        {
        if (i==3)
                {
                continue;
                }
        document.write("The number is " + i);
        document.write("<br />");
```

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

Pernyataan For...In

Perulangan ini digunakan untuk melakukan suatu perulangan berdasarkan elemen-elemen suatu array atau properti-properti suatu objek. Sintaks:

```
for (variable in object)
{
code to be executed
}
```

Kode dalam tubuh perulangan akan dieksekusi untuk setiap elemen atau properti. Argumen *variable* dapat berupa sebuah variabel, elemen array, atau properti objek. Contoh:

```
<html>
<body>
<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";
for (x in mycars)
        {
        document.write(mycars[x] + "<br />");
        }
</script>
</body>
</html>
```

Even

Pengertian even ini adalah suatu aksi yang dapat dideteksi oleh Javascript. Setiap elemen dalam sebuah halaman web mempunyai even tertentu yang dapat memicu Javascript. Misalnya, dengan even onClick, kita dapat menetapkan suatu fungsi supaya dieksekusi jika suatu tombol diklik oleh user. Even didefinisikan di dalam tag HTML. Even-even tersebut contohnya:

- Klik mouse
- Loading halaman web atau gambar.
- Pergerakan mouse melewati sebuah hotspot pada halaman web
- Pemilihan sebuah input di dalam sebuah form HTML.
- Submit sebuah form HTML
- Penekanan sebuah tombol keyboard.

Secara normal penggunaan even dikombinasikan dengan sebuah fungsi yang tidak akan dieksekusi sebelum even terjadi.

onLoad dan onUnload

Even onLoad akan terpicu jika terjadi user masuk ke sebuah halaman, sedangkan onUnload akan terjadi jika user meninggalkan sebuah halaman.
Even onLoad biasanya digunakan untuk mendeteksi jenis dan versi browser pengunjung dan memuat versi halaman web tepat berdasarkan informasi tersebut.
Baik even onLoad maupun onUnload juga kerap digunakan untuk menyimpan suatu cookie yang diset ketika pengunjung masuk atau meninggalkan halaman. Ketika pertama kali berkunjung, seorang pengunjung akan ditampilkan sebuah popup yang menanyakan namanya. Kemudian nama tersebut disimpan dalam sebuah cookie. Kemudian ketika suatu saat pengunjung kembali, maka suatu popup dapat ditampilkan yang menampilkan ucapan selamat datang misalnya.

onFocus, onBlur dan onChange

Even-even ini biasanya digunakan dalam suatu kombinasi untuk memvalidasi suatu inputan form. Sebagai contoh, di bawah ini penggunaan even onChange pada suatu inputan form yang akan mengeksekusi suatu fungsi *checkmal()* jika isi inputan mengalami perubahan.

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```

onSubmit

Even ini biasanya digunakan untuk memvalidasi semua inputan form saat akan disubmit.
Contoh berikut ini tentang bagaimana penggunaan even ini. Ketika user mengklik tombol submit form, maka suatu fungsi yang bernama *checkForm()* akan dipanggil. Fungsi ini akan memeriksa isi inputan form. Fungsi ini akan mengembalikan nilai *false* jika isian tidak diterima dan nilai *true* jika isian diterima. Jika nilai true yang dikembalikan, form akan disubmit, sebaliknya form akan dibatalkan.

```
<form method="post" action="xxx.htm" onsubmit="return checkForm()">
```

onMouseOver dan onMouseOut

Even ini biasanya digunakan untuk membuat suatu tombol animasi.
Contoh dibawah ini akan menampilkan kotak *alert* jika sebuah even onMouseOver terdeteksi.

```
<a href="http://www.w3schools.com" onmouseover="alert('An onMouseOver event');return false"><img src="w3s.gif" alt="W3Schools" /></a>
```

Try dan Catch

Try dan catch digunakan untuk menguji suatu blok kode terhadap suatu kesalahan. Ketika Anda browsing halaman-halaman web di Internet, suatu saat mungkin Anda akan menjumpai suatu halaman web yang menampilkan suatu kotak peringatan Javascript yang menyatakan bahwa ada suatu kesalahan runtime dan menanyakan "Do you wish to debug?". Pesan kesalahan semacam ini mungkin berguna bagi para pengembang website, tetapi tidak untuk pengunjung. Ketika pengunjung melihat kesalahan ini ketika mengunjungi salah satu halaman web Anda, mereka biasanya akan meninggalkan website Anda. Hal ini tentunya tidak diharapkan oleh kita.
Pernyataan try dan catch memungkinkan kita untuk menguji blok kode terhadap kesalahan. Blok try berisi kode yang akan dijalankan, dan blok catch berisi kode yang akan dijalankan jika suatu kesalahan terjadi. Sintaks:

```
try
{
//Run some code here
}
catch(err)
{
//Handle errors here
}
```

Perhatikan bahwa penulisan pernyataan try dan catch dalam huruf kecil. Penulisan dengan huruf besar akan menyebabkan kesalahan.

Beberapa Contoh Penggunaan

Contoh dibawah ini akan menampilkan "Welcome Guest!" ketika tombol diklik. Disini sengaja diberikan suatu kesalahan dalam fungsi message(), yaitu penulisan pernyataan *alert* yang tertulis *adddalert*. Maka ketika terjadi kesalahan ini, blok catch akan menangkap kesalahan dan mengeksekusi kode yang sudah disiapkan. Kode akan menampilkan pesan kesalahan yang sudah disiapkan yang menyampaikan kepada user apa yang terjadi.

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
        {
        adddlert("Welcome guest!");
        }
catch(err)
```

```
        {
        txt="There was an error on this page.\n\n";
        txt+="Error description: " + err.description + "\n\n";
        txt+="Click OK to continue.\n\n";
        alert(txt);
        }
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

Contoh berikutnya menggunakan kotak *confirm* yang memberitahukan kepada user dapat mengklik tombol *OK* untuk melanjutkan melihat halaman atau *Cancel* yang akan membawa user menuju halaman homepage. Jika metode confirm mengembalikan nilai false, yang berarti user mengklik tombol *Cancel*, maka user akan di-*redirect.* Jika metode confirm mengembalikan nilai true, kode tidak akan melakukan apa-apa.

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
        {
        adddlert("Welcome guest!");
        }
catch(err)
        {
        txt="There was an error on this page.\n\n";
        txt+="Click OK to continue viewing this page,\n";
        txt+="or Cancel to return to the home page.\n\n";
        if(!confirm(txt))
                {
                document.location.href="http://www.w3schools.com/";
                }
        }
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
```

```
</html>
```

Throw

Pada struktur try...catch, ketika terjadi suatu kesalahan maka blok kode dalam catch akan dieksekusi. Kode untuk menangani kesalahan didalam blok catch dapat dibuat terdiri dari beberapa kelompok kode yang akan menangani kesalahan yang berbeda-beda. Disini kesalahan yang mungkin terjadi harus sudah diketahui terlebih dahulu. Untuk lebih jelasnya lihat contoh dibawah ini:

```
<!DOCTYPE html>
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 5 and 10:","");
try
  {
  if(x>10)
    {
    throw "Err1";
    }
  else if(x<5)
    {
    throw "Err2";
    }
  else if(isNaN(x))
    {
    throw "Err3";
    }
  }
catch(err)
  {
  if(err=="Err1")
    {
    document.write("Error! The value is too high.");
    }
  if(err=="Err2")
    {
    document.write("Error! The value is too low.");
    }
  if(err=="Err3")
    {
    document.write("Error! The value is not a number.");
    }
  }
</script>
```

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

Pada contoh, terdapat suatu variabel *x* berisi nilai yang dimasukkan oleh user. Nilai tersebut harus lebih besar dari sama dengan 5, kurang dari sama dengan 10, dan harus berupa angka. Jika tidak sesuai dengan ketentuan tersebut maka akan tampil pesan kesalahan yang tergantung dari kesalahan yang terjadi.
Untuk menampilkan pesan kesalahan yang tepat sesuai dengan kesalahan yang terjadi disini menggunakan pernyataan *throw.* Sintaks pernyataan ini:

```
throw exception
```

exception dapat berupa sebuah string, integer, boolean, atau objek. Pada contoh diatas *exception* berupa string (*err1, err2, err3*)

Karakter Khusus

Didalam javascript suatu teks string dapat dimasukkan karakter khusus menggunakan tanda *backslash*. Backslash ini dapat digunakan untuk menyisipkan karakter *apostrophes, new lines, quotes,* dan karakter khusus lainnya. Supaya lebih jelas lihat contoh berikut:

```
var txt="We are the so-called "Vikings" from the north.";
document.write(txt);
```

Suatu string dalam javascript diapit oleh tanda petik tunggal atau ganda. Maka kode diatas akan menghasilkan: *We are so-called*. Disini terdapat karakter khusus dalam string, yaitu *double quotes* atau tanda petik ganda. Agar seluruh teks dapat dicetak perlu menggunakan backslash didepan setiap tanda petik ganda sehingga kode diatas akan menjadi:

```
var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);
```

Tabel dibawah ini berisi daftar karakter khusus yang dapat disisipkan kedalam string menggunakan backslash.

Kode	Keluaran
\’	tanda petik tunggal
\”	tanda petik ganda
\\	garis miring
\n	baris baru
\r	*carriage return*
\t	*tab*
\b	*backspace*
\f	*form feed*

Lebih Lanjut Tentang Event

Event-event javascript berhubungan dengan DOM HTML. Sebenarnya, event-event pada javascript adalah atribut-atribut yang dideskripsikan pada DOM HTML. Event-event ini dikombinasikan dengan javascript yang berfungsi menyediakan kode-kode untuk dieksekusi bila suatu event terjadi.
Selanjutnya akan dibahas satu persatu even-even tersebut secara lebih detail.

Event click

Event ini akan tejadi bila tombol mouse kiri diklik pada sebuah elemen. Disini ada dua sintaks. Sintaks pertama jika digunakan dalam suatu elemen HTML, yang kedua jika digunakan pada script Javascript.

Dalam HTML:
<elemen onclick=”kodejavascript”>
Dalam Javascript:
object.onclick=”kodejavascript”;

kodejavascript adalah kode javascript yang akan dieksekusi manakala event terjadi.

onclick dapat digunakan pada tag-tag HTML dibawah ini:

<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <object>, <ol>, <p>, <pre>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Dan object javascript yang dapat menggunakan event ini:

Document, Window

Contoh berikut ini akan menampilkan data tanggal pada elemen paragrap bernama *demo* jika tombol *Display Date* diklik.

```
<html>
<head>
<script type="text/javascript">
function displayDate()
        {
        document.getElementById("demo").innerHTML=Date();
        }
</script>
</head>
<body>
```

```
<h1>My First Web Page</h1>
<p id="demo">This is a paragraph.</p>
<button type="button" onclick="displayDate()">Display Date</button>
</body>
</html>
```

Event onmousedown dan onmouseup

Event onmousedown terjadi jika tombol mouse diklik pada suatu elemen sedangkan event onmouseup terjadi jika tombol mouse dilepas setelah diklik sebelumnya.

Sintaks keduanya:
Pada elemen HTML:

```
<elemen onmousedown=”kodejavascript”>
<elemen onmouseup=”kodejavascript”>
```

Pada script javascript:

```
<object.onmousedown=”kodejavascript”;>
<object onmouseup=”kodejavascript”;>
```

Sementara, *kodejavascript* adalah kode javascript yang akan dieksekusi bila event terjadi.

onmousedown dan onmouseup support pada tag-tag HTML:

<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <ol>, <p>, <pre>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

object javascript yang support:

Document, Window

Contoh:

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function mouseDown()
    {
    document.getElementById("p1").style.color="red";
    }

function mouseUp()
    {
    document.getElementById("p1").style.color="green";
    }
```

```
</script>
</head>
<body>

<p id="p1" onmousedown="mouseDown()" onmouseup="mouseUp()">Click the text! The mouseDown() function is triggered when the mouse button is pressed down over this paragraph. The function sets the color of the text to red.
The mouseUp() function is triggered when the mouse button is released. That function sets the color of the text to green.</p>

</body>
</html>
```

Contoh diatas akan mengeksekusi fungsi *mouseDown()* dan mengubah warna teks paragrap menjadi merah bila tombol mouse ditekan. Dan warna teks paragrap akan dirubah oleh fungsi *mouseUp()* menjadi hijau bila tombol mouse dilepas setelah ditekan.

Objek Javascript

Javascript adalah bahasa pemrograman berbasis objek. Suatu bahasa pemrograman berbasis objek mengijinkan kita mendefinisikan objek dan tipe data milik kita sendiri. Perlu dicatat, objek hanyalah suatu jenis khusus dari data.

Javascript memiliki objek-objek yang siap untuk digunakan, disebut built-in objek. Berikut ini referensi lengkap objek yang dimaksud.

Objek String

Objek string berguna untuk memanipulasi teks, dibuat dengan menggunakan perintah:

```
new String()
```

Contoh:

```
var txt=new String("string");
```

atau dapat juga langsung seperti ini:

```
var txt="string";
```

Properti

Properti	Penjelasan
constructor	Mengembalikan fungsi yang membuat prototype dari objek string
length	Mengembalikan panjang string
prototype	Memperbolehkan kita menambahkan properti dan method ke sebuah objek

Method

Properti	Penjalasan
charAt()	Mengembalikan karakter pada indeks spesifik
charCodeAt()	Mengembalikan Unicode dari karakter pada indeks spesifik
concat()	Menggabungkan dua atau lebih string dan mengembalikan salinan dari string yang telah digabung
fromCharCode()	Mengkonversi nilai Unicode ke karakter
indexOf()	Mengembalikan posisi dari nilai tertentu yang ditemukan pertama kali pada sebuah string
lastIndexOf()	Mengembalikan posisi dari nilai tertentu yang ditemukan terakhir kali pada sebuah string
match()	Mencari sebuah kesamaan diantara sebuah ekspresi reguler dan sebuah string, dan mengembalikan nilai

	kecocokannya
replace()	Mencari kesamaan antara sebuah substring (atau ekspresi reguler) dan sebuah string, dan menggantikan substring tersebut dengan substring yang baru
search()	Mencari kesamaan antara sebuah ekspresi reguler dan sebuah string, dan mengembalikan nilai posisi kesamaannya
slice()	Mengekstrak suatu bagian dari sebuah string dan mengembalikan sebuah string baru
split()	Membagi sebuah string kedalam sebuah array dari subtring
substr()	Mengekstrak karakter dari sebuah string, dimulai dari posisi start tertentu sejumlah karakter tertentu
substring()	Mengekstrak karakter dari sebuah string antara dua indeks tertentu
toLowerCase()	Mengkonversi sebuah string ke huruf kecil
toUpperCase()	Mengkonversi sebuah string ke huruf besar
valueOf()	Mengembalikan nilai primitif dari sebuah objek string

PROPERTY

Properti Constructor

Sintaks:

```
string.constructor
```

Contoh:

```
<script type="text/javascript">
var txt = "Hello World!";
document.write(txt.constructor);
</script>
```

Output:

```
function String() { [native code] }
```

Properti Length

Sintaks:

```
string.length
```

Contoh:

```
<script type="text/javascript">
var txt = "Hello World!";
```

```
document.write(txt.length);
</script>
```

Output:

```
12
```

Properti Prototype

Merupakan properti. global

Sintaks:

object.prototype.name=value

Contoh:

```
<script type="text/javascript">
function employee(name,jobtitle,born)
{
this.name=name;
this.jobtitle=jobtitle;
this.born=born;
}

var fred=new employee("Fred Flintstone","Caveman",1970);
employee.prototype.salary=null;
fred.salary=20000;

document.write(fred.salary);
</script>
```

Output:

```
20000
```

METHOD

Method charAt()

Indeks dari karakter pertama adalah 0 dan indeks karakter terakhir dari string bernama “txt” adalah txt.length-1

Sintaks:

```
string.charAt(index)
```

Parameter indeks diperlukan yang bernilai integer 0 sampai dengan *string*.length-1

Contoh:

```
<script type="text/javascript">
var str = "Hello world!";
document.write("First character: " + str.charAt(0) + "<br />");
document.write("Last character: " + str.charAt(str.length-1));
</script>
```

Output:

```
First character: H
Last character: !
```

Method charCodeAt()

Sintaks:

string.charCodeAt(index)

Contoh:

```
<script type="text/javascript">
var str = "Hello world!";
document.write("First character: " + str.charCodeAt(0) + "<br />");
document.write("Last character: " + str.charCodeAt(str.length-1));
</script>
```

Output:

```
First character: 72
Last character: 33
```

Method concat()

Sintaks:

```
string.concat(string2, string3, ..., stringX)
```

Contoh:

```
<script type="text/javascript">
var str1="Hello ";
var str2="world!";
var str3=" Have a nice day!";
document.write(str1.concat(str2,str3));
</script>
```

Output:

Hello world! Have a nice day!

Method fromCharCode()

Sintaks:

String.fromCharCode(n1, n2, ..., nX)

Contoh:

```
<script type="text/javascript">
document.write(String.fromCharCode(72,69,76,76,79));
</script>
```

Output:

Hello

Method indexOf()

Mengembalikan nilai -1 jika nilai yang dicari tidak ada

Sintaks:

string.indexOf(searchstring, [start])

Contoh:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.indexOf("d") + "<br />");
document.write(str.indexOf("WORLD") + "<br />");
document.write(str.indexOf("world"));
</script>
```

Output:

10
-1
6

Perbedaan antara indexOf dan lastIndexOf:

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

```
var str="I love W3Schools!"


document.write("Index of first o: " + str.indexOf("o"))
document.write("<br />")
document.write("Index of last o: " + str.lastIndexOf("o"))
document.write("<br />")
document.write("Index of first 'love': " + str.indexOf("love"))
document.write("<br />")
document.write("Index of last 'love': " + str.lastIndexOf("love"))

</script>
```

Output:

```
Index of first o: 3
Index of last o: 13
Index of first 'love': 2
Index of last 'love': 2
```

Method lastIndexOf()

String dicari mulai dari belakang, tetapi indeks yang dikembalikan adalah posisi karakter dari kiri ke kanan (mulai dari 0).

Sintaks:

```
string.lastIndexOf(searchstring, start)
```

Contoh:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.lastIndexOf("d") + "<br />");
document.write(str.lastIndexOf("WORLD") + "<br />");
document.write(str.lastIndexOf("world"));
</script>
```

Output:

```
10
-1
6
```

Method match()

Mengembalikan nilai dalam bentuk array, null jika tidak ada yang cocok.

Sintaks:

string.match(regexp)

Contoh:

```
<script type="text/javascript">
var str="The rain in SPAIN stays mainly in the plain";
var patt1=/ain/gi;
document.write(str.match(patt1));
</script>
```

Output:

ain,AIN,ain,ain

Method replace()

Sintaks: *string*.replace(*regexp/substr,newstring*)

Contoh:

Pencarian case sensitive:

```
<script type="text/javascript">
var str="Visit Microsoft!";
document.write(str.replace("Microsoft", "W3Schools"));
</script>
```

Keluaran:

```
Visit W3Schools!
```

Search()

Akan mencari kecocokan antara ekspresi reguler dengan string. Mengembalikan nilai berupa posisi kecocokan ditemukan. Jika tidak ada kecocokan akan mengembalikan nilai -1

Sintaks: *string*.search(regexp)

Contoh 1:

Pencarian case sensitive:

```
<script type="text/javascript">
var str="Visit W3Schools!";
document.write(str.search("W3SCHOOLS"));
</script>
```

Keluaran:

```
-1
```

Contoh 2:

Pencarian case-insensitive:

```
<script type="text/javascript">
var str="Visit W3Schools!";
document.write(str.search(/w3schools/i));
</script>
```

Keluaran:

```
6
```

Slice()

Akan mengekstrak bagian dari suatu string dan mengembalikan hasil ekstrak tersebut sebagai sebuah string baru.

Sintaks: *string*.slice(begin,end)

Contoh:

```
<script type="text/javascript">
var str="Hello happy world!";

// extract all characters, start at position 0:
document.write(str.slice(0)+"<br />");

// extract all characters, start at position 6:
document.write(str.slice(6)+"<br />");

// extract from the end of the string, and to position -6:
document.write(str.slice(-6)+"<br />");

// extract only the first character:
document.write(str.slice(0,1)+"<br />");

// extract the characters from position 6 to position 11:
document.write(str.slice(6,11)+"<br />");
</script>
```

Keluaran:

```
Hello happy world!
happy world!
world!
H
happy
```

Split()

Akan membagi sebuah string menjadi array dari sub-substring.

Sintaks: *string*.split(separator, limit)

Contoh:

```
<script type="text/javascript">
var str="How are you doing today?";

document.write(str.split() + "<br />");
document.write(str.split(" ") + "<br />");
document.write(str.split("") + "<br />");
document.write(str.split(" ",3));
</script>
```

Keluaran:

```
How are you doing today?
How,are,you,doing,today?
H,o,w, ,a,r,e, ,y,o,u, ,d,o,i,n,g, ,t,o,d,a,y,?
How,are,you
```

Substr()

Akan mengambil karakter-karakter dari sebuah string dimulai dari “start” sampai sejumlah tertentu karakter dan mengembalikan nilai berupa sub string baru.

Sintaks: *string*.substr(start,length)

Contoh:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.substr(3)+"<br />");
document.write(str.substr(3,4));
</script>
```

Keluaran:

```
lo world!
lo w
```

Substring()

Akan mengambil karakter-karakter dari sebuah string diantara dua indeks yang ditentukan dan mengembalikan sub string baru. Metode ini akan mengambil karakter diantara “from” dan “to” tidak termasuk “to” itu sendiri.

Sintaks: *string*.substring(from, to)

Contoh:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.substring(3)+"<br />");
document.write(str.substring(3,7));
</script>
```

Keluaran:

```
lo world!
lo w
```

ToLowerCase()

Sintaks: *string*.toLowerCase()

Contoh:

```
<script type="text/javascript">
var str="Hello World!";
document.write(str.toLowerCase());
</script>
```

ToUpperCase()

Sintaks: *string*.toUpperCase()

Contoh:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.toUpperCase());
</script>
```

ValueOf()

Method ini biasanya akan dipanggil secara otomatis oleh Javascript, tidak secara explisit dalam kode.

Sintaks: *string*.valueOf()

Contoh:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.valueOf());
</script>
```

Keluaran:

```
Hello world!
```

Objek Date

Objek Date digunakan untuk bekerja dengan tanggal dan waktu. Objek Date dibuat dengan perintah: new Date().

Ada 4 cara untuk menggunakan perintah diatas :

```
var d = new Date();
var d = new Date(milliseconds);
var d = new Date(dateString);
var d = new Date(year, month, day, hours, minutes, seconds, milliseconds);
```

Sebagian besar parameter diatas bersifat opsional, jika tidak ditentukan maka nilai 0 yang akan dilewatkan. Sistem waktu yang digunakan bisa berupa waktu lokal(local time) atau UTC(Universal, GMT). Tanggal dihitung dalam milidetik dari 1 Januari 1970 00:00:00 Universal Time(UTC), sehingga satu hari berisi 86.400.000 milidetik.

Beberapa contoh deklarasi objek tanggal:

```
var today = new Date()
var d1 = new Date("October 13, 1975 11:13:00")
var d2 = new Date(79,5,24)
var d3 = new Date(79,5,24,11,33,0)
```

Objek tanggal dapat dimanipulasi dengan mudah menggunakan method yang tersedia untuk objek tanggal. Sebagai contoh, dibawah ini akan diset objek tanggal ke tanggal 14 Januari 2012:

```
var myDate=new Date();
yDate.setFullYear(2012,0,14);
```

sedangkan dibawah ini objek tanggal diset sebagai tanggal 5 hari kedepan:

```
var myDate=new Date();
myDate.setDate(myDate.getDate()+5);
```

PROPERTI

Properti	Penjelasan
constructor	Mengembalikan fungsi yang membuat prototype objek tanggal
prototype	Dapat digunakan untuk menambah method dan properti ke sebuah objek

constructor()

Sintaks: `object.prototype.name=value`

Contoh:

```
<script type="text/javascript">
function employee(name,jobtitle,born)
{
this.name=name;
this.jobtitle=jobtitle;
this.born=born;
}

var fred=new employee("Fred Flintstone","Caveman",1970);
employee.prototype.salary=null;
fred.salary=20000;

document.write(fred.salary);
</script>
```

Keluaran:

```
20000
```

prototype()

Sintaks: `object.prototype.name=value`

Contoh:

```
<script type="text/javascript">
function employee(name,jobtitle,born)
{
this.name=name;
this.jobtitle=jobtitle;
this.born=born;
}

var fred=new employee("Fred Flintstone","Caveman",1970);
employee.prototype.salary=null;
fred.salary=20000;

document.write(fred.salary);
</script>
```

Keluaran:

```
20000
```

METHOD

getDate()

Mengembalikan nilai berupa tanggal dalam satu bulan (1 s/d 31), sesuai dengan waktu lokal.

Sintaks: *Date*.getDate()

Contoh 1:

Mengembalikan nilai tanggal:

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

var d = new Date();
document.write(d.getDate());

</script>
```

Keluaran:

```
31
```

Contoh 2:

Mengembalikan tanggal dari tanggal tertentu:

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

var d = new Date("July 21, 1983 01:15:00");
document.write(d.getDate());

</script>
```

Keluaran:

```
21
```

getDay()

Mengembalikan nilai tanggal dalam seminggu(0 s/d 6) sesuai waktu lokal. Hari Minggu dihitung sebagai 0, Senin 1, dan seterusnya.

Sintaks: *Date*.getDay()

Contoh 1:

Mengembalikan nilai hari dalam seminggu:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getDay());
</script>
```

Keluaran:

```
2
```

Contoh 2:

Mengembalikan nama hari dalam seminggu:

```
<script type="text/javascript">
var d=new Date();
var weekday=new Array(7);
weekday[0]="Sunday";
weekday[1]="Monday";
weekday[2]="Tuesday";
weekday[3]="Wednesday";
weekday[4]="Thursday";
weekday[5]="Friday";
weekday[6]="Saturday";

document.write("Today is " + weekday[d.getDay()]);
</script>
```

Keluaran:

```
Today is Tuesday
```

getFullYear()

Akan mengembalikan nilai tahun sepanjang 4 digit berdasarkan waktu lokal.

Sintaks: *Date*.getFullYear()

Contoh 1:

Mengembalikan 4 digit tahun:

```
<script type="text/javascript">
var d = new Date();
```

```
document.write(d.getFullYear());
</script>
```

Keluaran:

```
2012
```

Contoh 2:

Mengembalikan 4 digit tahun dari tanggal tertentu:

```
<script type="text/javascript">
var d = new Date("July 21, 1983 01:15:00");
document.write("I was born in " + d.getFullYear());
</script>
```

Keluaran:

```
I was born in 1983
```

getHours()

Mengembalikan nilai jam (0 s/d 23) berdasarkan waktu lokal.

Sintaks: *Date*.getHours()

Contoh 1:

Mengembalikan jam:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getHours());
</script>
```

Keluaran:

```
16
```

Contoh 2:

Mengembalikan jam dari tanggal tertentu:

```
<script type="text/javascript">
var d = new Date("July 21, 1983 01:15:00");
document.write(d.getHours());
</script>
```

Keluaran:

```
1
```

getMilliseconds()

Mengembalikan nilai milidetik (0 s/d 999) dari objek Date berdasarkan waktu lokal.

Sintaks: *Date*.getMilliseconds()

Contoh 1:

Mengembalikan nilai milidetik:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getMilliseconds());
</script>
```

Keluaran:

```
263
```

Contoh 2:

Mengembalikan nilai milidetik dari objek Date:

```
<script type="text/javascript">
var d = new Date("July 21, 1983 01:15:00");
document.write(d.getMilliseconds());
</script>
```

The output of the code above will be:

```
0
```