1. Sejarah Javascript

JavaScript pada awalnya bernama LiveScript dan diperkenalkan pertama kali di browser Netscape Navigator 2 pada awal tahun 1995. Pengembangan javascrip ditujujkan untuk mempermudah dalam mengakses Java Applet yang sedang popular di masa itu.

Untuk mempermudah programmer non-Java dalam mengakses Java Applet tersebut, maka Netscape mengembangkan Javascript. Ditujukan untuk desainer dan pengembang web, Javascript dirancang untuk dapat digunakan tanpa memerlukan pengetahuan yang terlalu mendalam tentang pemrograman berorientasi objek.

Javascript, seperti namanya, merupakan bahasa pemrograman scripting. Dan seperti bahasa scripting lainnya, Javascript umumnya digunakan hanya untuk program yang tidak terlalu besar: biasanya hanya beberapa ratus baris. Ingat kembali tujuan awal Javascript: hanya untuk mengontrol program Java. Jadi memang pada dasarnya Javascript tidak dirancang untuk digunakan dalam aplikasi skala besar.

Karena tujuan yang sederhana ini, ditambah dengan tekanan yang sangat besar untuk meluncurkan sebuah bahasa secepatnya, perancangan Javascript dilakukan secara instan. Javascript dikembangkan dalam waktu yang sangat singkat, hanya 10 hari saja. Walaupun merupakan pencapaian yang sangat hebat, singkatnya waktu pengembangan Javascript ini menyebabkan Javascript memiliki banyak kekurangan. Begitupun, kekurangan-kekurangan Javascript tidak mengurangi kesuksesannya.

Meskipun dibuat dengan tujuan awal untuk mengendalikan program Java, komunitas Javascript menggunakan bahasa ini untuk tujuan lain: memanipulasi gambar dan isi dari dokkumen HTML. Singkatnya, pada akhirnya Javascript digunakan untuk satu tujuan utama: "menghidupkan" dokumen HTML dengan mengubah konten statis menjadi dinamis dan interaktif. Bersamaan dengan perkembangan Internet dan dunia web yang pesat, Javascript akhirnya menjadi bahasa utama dan satu-satunya untuk membuat HTML menjadi interaktif di dalam browser.

Di masa sekarang, 2016 dan selanjutnya, Javascript telah menjadi bahasa pemrograman yang umum di dalam dunia pemrograman web dalam browser. Meskipun telah lahir bahasa-bahasa lain untuk pengembangan web, bahasa-bahasa tersebut biasanya dikompilasi ke Javascript. Javascript juga telah merambah ke dunia server (lihat: NodeJS), dan digunakan di luar konteks browser dan HTML. Javascript tak lagi terikat dengan HTML dan browser, dan tak lagi hanya digunakan untuk program-program kecil.

2. Mengapa Memilih Javascript ?

Setiap bahasa pemrograman memiliki pendekatan tersendiri untuk menyelesaikan masalah. Pendekatan atau cara pandang untuk penyelesaian masalah ini dikenal dengan istilah paradigma pemrograman. Meskipun ada sangat banyak paradigma pemrograman, hanya 5 yang populer dan banyak digunakan, yaitu imperatif, fungsional, orientasi objek, logis, dan simbolik. Java dan C# merupakan contoh bahasa pemrograman berorientasi objek. Haskell dan Scheme utamanya adalah bahasa pemrograman fungsional. C merupakan bahasa imperatif, sementara Prolog adalah bahasa Logis dan Datalog dikenal sebagai bahasa dengan paradigma simbolik. Tentu saja satu bahasa bisa mendukung lebih dari satu paradigma, tetapi biasanya ada satu paradigma yang paling menonjol dari bahasa tersebut. C# misalnya mendukung banyak fitur fungsional dan imperatif, tetapi utamanya adalah bahasa berorientasi objek.

Mengetahui dan mengerti paradigma pemrograman dari sebuah bahasa dapat disamakan dengan mengerti kegunaan utama dari sebuah alat kerajinan. Dengan mengerti *kenapa* sebuah alat dibuat, kita dapat menggunakan alat tersebut sesuai dengan tujuannya, dan bekerja dengan efektif. Bayangkan jika anda diminta untuk memasang paku dengan menggunakan pahat. Tentu saja anda bisa tetap memasang paku tersebut, misalnya dengan mengetuknya menggunakan gagang pahat tersebut. Tetapi hasilnya adalah, paku tidak masuk dengan lurus dan tangan anda sakit. Seorang pengrajin yang baik tentunya akan mencari palu ketika diminta untuk memasang paku. Dengan mengetahui dan mengerti paradigma pemrograman dari sebuah bahasa, kita dapat memanfaatkan paradigma tersebut sesuai dengan keperluan. Jika sebuah bahasa mendukung tiga paradigma, maka kita harus tahu kapan waktu untuk menggunakan masing-masing paradigma tersebut dengan baik. Kita dapat menghemat ratusan baris dan menghasilkan kode yang mudah dirawat dengan memanfaatkan pengetahuan akan hal ini.

Pertanyaan yang paling penting, tentunya, adalah apa saja paradigma pemrograman yang didukung oleh Javascript? Javascript mendukung tiga paradigma pemrograman utama, yaitu imperatif, fungsional, dan berorientasi objek.

Javascript merupakan bahasa pemrograman imperatif, seperti layaknya C. Hal ini berarti Javascript dapat menjalankan perintah program baris demi baris, dengan masing-masing baris berisi satu atau lebih perintah. Dapat dikatakan bahwa kita mendeskripsikan *bagaimana* menyelesaikan masalah kepada Javascript ketika menulis

program Javascript. Pada setiap langkah dalam program imparatif, umumnya kita akan mengubah keadaan (*state*) program ataupun data.

Javascript juga merupakan bahasa pemrograman fungsional, yang mana kita memandang struktur dan elemen-elemen dalam program sebagai fungsi matematis yang tidak memiliki keadaan (*state*) dan data yang dapat berubah (*mutable data*). Pemrograman fungsional, seperti namanya, berfokus pada fungsi. Fungsi merupakan nilai kelas pertama pada paradigma ini, yang berarti fungsi diperlakukan sama seperti nilai-nilai lain seperti variabel dan kelas. Fungsi dapat disimpan di dalam variabel, dikirimkan sebagai parameter dari fungsi lain, dan dapat mengembalikan fungsi baru. Pemrograman fungsional juga menekankan pada fungsi rekursif, yaitu fungsi yang memanggil dirinya sendiri, serta fungsi murni atau fungsi yang tidak memiliki efek samping. Efek samping di sini artinya fungsi mempengaruhi atau dipengaruhi oleh hal-hal di luar fungsi tersebut seperti nilai variabel di luar fungsi. Dengan tidak memiliki efek samping, sebuah fungsi akan selalu memberikan hasil yang sama jika dipanggil dengan argumen yang nilainya sama.

Selain imparatif dan fungsional, Javascript juga mendukung pemrograman berorientasi objek. Pada pemrograman berorientasi objek, kita mendeskripsikan segala sesuatu yang terlibat dalam program kita sebagai "objek". Sebuah objek memiliki data (atribut yang mendeskripsikan objek tersebut) serta operasi-operasi yang dapat dilakukan terhadap objek tersebut. Perlu diingat bahwa meskipun kebanyakan bahasa berorientasi objek mengadopsi pendekatan kelas (*class-based*), Javascript menggunakan pendekatan yang berbeda, yaitu pendekatan prototipe (*prototype-based*). Pembahasan mendetail mengenai hal ini akan dilakukan pada bagian selanjutnya dari buku ini.

Dari ketiga paradigma yang didukung oleh Javascript, paradigma imparatif merupakan paradigma yang paling sederhana dan mudah dimengerti. Hampir semua bahasa pemrograman yang ada, dari Pascal sampai ke C# mendukung paradigma ini. Jika anda pernah menulis program yang menjalankan baris demi baris kode secara berurutan, anda telah menulis program dengan paradigma imparatif. Pembahasan tentang paradigma imparatif tidak akan dilakukan, dan kita akan berfokus kepada dua aspek dari Javascript yang jarang dibahas: paradigma fungsional dan imparatif. Empat bab selanjutnya dari buku ini akan membahas kedua hal tersebut secara mendalam.

3. Apa yang bisa dilakukan JavaScript ?

4. Memulai JavaScript

Code Javascript dapat disisipkan dalam bagian <head> </head> maupun dalam <body>... </body>. Untuk menyisipkan Suatu script dalam dokurnen HTML, caranya hampir sama dengan menyisipkan bagian HTML yang lain, yaitu dengan menggunakan tag untuk menandai awal dan akhir. Tag untuk menandai awal script adalah <script> dan djakhiri dengan </script>. Tag <script> memiliki bebeIapa atribut, tetapi yang terpenting adalah atribut **language** dan **type**. Karena Javascript bukan satu—satunya bahasa scripting maka sangatlah perlu untuk memberitahukan kepada browser bahwa bahasa script yang digunakan adalah JavaScript dan selanjutnya browser akan menjalankan modul pendukung Iavascript untuk memrosesnya sehingga untuk Javascript, pada tag <script> perlulah "ditambahkan atribut berikut.

- <script language="JavaScript" type = "text/javascript" >

Sekarang kita akan membuat contoh Javascript. Pada contoh ini, Javascript digunakan untuk memberi warna merah pada background halaman. Kita dapat membuat script ini melalui:

Background.html

```
1.  <html>
2.  <HEAD>
3.      <title>Background</title>
4.  </HEAD>
5.  <body bgcolor="White">
6.      <p>Mengatur Warna Latar dengan JavaScript</p>
7.
8.      <script language="JavaScript" type="text/javascript">
9.          document.bgColor ="red";
10.     </script>
11.
12. </body>
13. </html>
```

hasil dari file tersebut adalah:

Jika kita perhatikan pada bagian Javascript, maka pada akhir statement di akhiri dengan tanda semicolon atau titik coma. Untuk mengubah wama halaman web sendiri, kita menggunakan perintah document.bgColor= “red";.

Apa yang kita sebut dengan ***halaman*** adalah ***document*** sehingga terdapat perintah document pada perintah di atas. Selanjutnya ‘document’ memiliki banyak properti, termasuk di dalamnya adalah ‘bgcolor’ yang digunakan Untuk mengatur wama background halaman. Sekarang, kita lanjutkan contoh kedua.

Contoh kedua kita akan mencoba fungsi alert..

Coba buat dokumen seperti berikut

```
<html>
<head>
    <title>Fungsi Alert</title>
</head>
    <body bgcolor="white">
<p>Paragraph 1</p>
<script language="JavaScript" type="text/javascript">
alert("Halo Bro.... ");
</script>
</body>
</html>
```

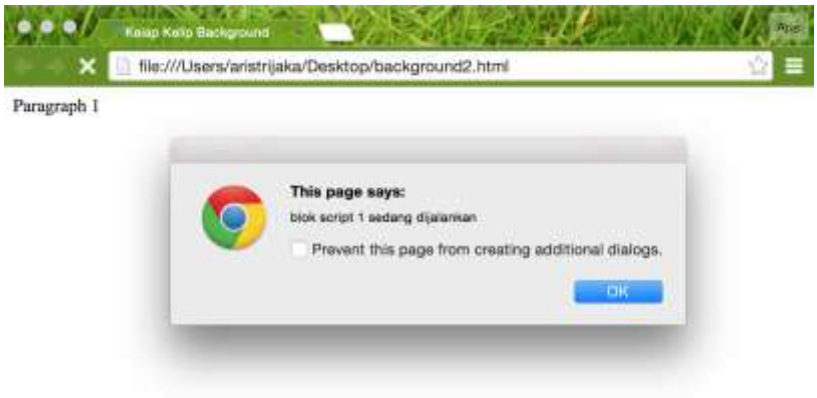
```
<html>
<head>
    <title>Fungsi Alert</title>
</head>
    <body bgcolor="white">
<p>Paragraph 1</p>
<script language="JavaScript" type="text/javascript">
alert("Halo Bro.... ");
alert("Apakabar ??");
</script>
</body>
</html>
```

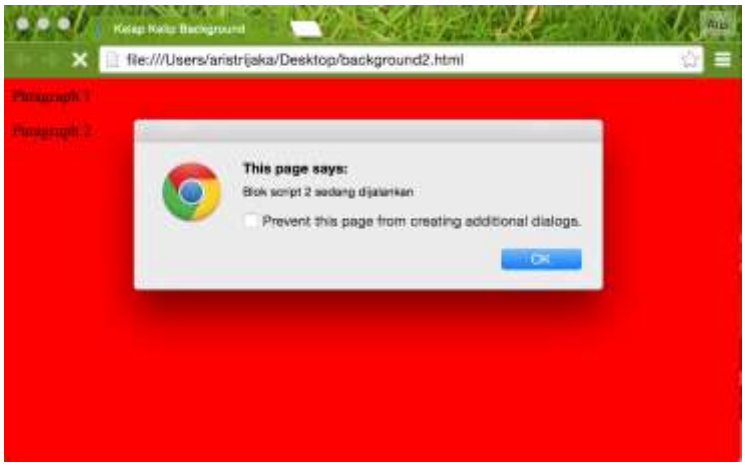
Pada contoh ketiga ini kita akan melihat bagaimana flow proses dari Javascript terkait dengan adanya beberapa blok yang digunakan.

```
<html>
<head>
    <title>Kelap Kelip Background</title>
</head>
    <body bgcolor="white">

<p>Paragraph 1</p>
<script language="JavaScript" type="text/javascript">
alert("blok script 1 sedang dijalankan");
</script>

<p>Paragraph 2</p>
<script language="JavaScript" type="text/javascript">
document.bgColor = "RED";
alert("Blok script 2 sedang dijalankan");
</script>

<p>paragraph 3</p>
</body>
</html>
```

Jika script di atas dijalankan, berikut ini beberapa screen shot tampilannya. Tampilan gambar pertama, adalah ketika script blok 1 dijalankan. Gambar kedua ketika blok script 2 dijalankan. Sedangkan gambar terakhir adalah tampilan setelah semua script blok

dijalankan. Dari contoh ini dapat disimpulkan bahwa jika dalam suatu halaman web terdiri dari beberapa blok Javascript, jalannya script dimulai dari blok paling atas, kemudian berurutan ke blok berikutnya.





5. *Aturan dasar penulisan kode JS*

a. Perbedaan Penulisan Huruf (Case Sensitivity)

Di dalam **JavaScript**, penulisan huruf besar dan huruf kecil dibedakan, atau dalam istilah pemograman bersifat **Case Sensitif**. Hal ini berarti penulisan *variabel* , *keyword*, maupun nama*fungsi* di dalam JavaScript harus konsisten. Variabel **nama**, **Nama**, dan **NAMA** merupakan 3 variabel berbeda. Sedangkan untuk penulisan *keyword* **while**, harus ditulis dengan '**while'**, bukan '**While'** atau '**WHILE'**.

Namun karena **HTML** sendiri tidak bersifat *case-sensitif*, kadang hal ini bisa mendatangkan permasalahan. Contohnya, jika menggunakan **event handler**, di dalam HTML kadang bisa ditulis:**onclick**, **onClick**, atau **OnClick**. Ketiga penulisan ini dibolehkan di dalam **HTML**. Akan tetapi untuk menghindari permasalahan, sebaiknya kita membuat kesepakatan untuk menggunakan huruf kecil untuk semua penulisan *keyword* dan *variabel* di dalam JavaScript.

b. Penggunaan Karakter Spasi, Enter, dan Tab (Whitespace)

Karakter-karakter spasi, enter, tab dan karakter lain yang 'tidak kelihatan' (sering dikenal dengan istilah whitespace) akan diabaikan pada saat pemrosesan JavaScript. Karakter-karakter ini bisa digunakan untuk 'menjorokkan' (indent) kode program agar lebih mudah dibaca.

c. Cara Penulisan Komentar dalam JavaScript

JavaScript mendukung 2 jenis cara penulisan komentar, yakni menggunakan karakater // untuk komentar dalam 1 baris, dan karakter pembuka komentar /* dan penutup */ untuk komentar yang mencakup beberapa baris.

Berikut adalah contoh penulisan komentar di dalam JavaScript:

```
<script>
   // ini adalah komentar dalam 1 baris
   /* Baris ini juga merupakan komentar */ //ini juga komentar

   /* Komentar ini
     mencakup beberapa
     baris
   */

   /*
   * Beberapa programmer
   * menambahkan tanda bintang
   * agar penulisan komentar
   * lebih rapi
   */
</script>
```

d. Aturan Penulisan Identifier (Variabel dan Nama Fungsi) JavaScript

Di dalam JavaScript, identifier adalah sebutan untuk nama. Nama ini bisa terdiri dari nama variabel, atau nama dari fungsi. Aturan penulisan identifier dalam JavaScript adalah :

- Karakter pertama harus diawali dengan huruf, underscore (_) atau tanda dollar ($)
- Karakter kedua dan seterusnya bisa ditambahkan dengan huruf, angka, underscore (_) atau tanda dollar ($).

Dari aturan tersebut dapat dilihat bahwa kita tidak bisa menggunakan angka sebagai karakter pertama dari sebuah variabel atau nama fungsi.

Berikut adalah contoh penulisan nama variabel yang dibolehkan:

```
<script>
   duniaInformatika
   $informatk
   v12
   _karakter
   b3l4j4r
</script>
```

Namun karakter berikut tidak bisa digunakan sebagai identifier:

```
<script>
   %duniailkom   // terdapat karakter %
   dunia ilkom   // terdapat karakter spasi
   4ngka         // diawali dengan angka
   suka#suka     // terdapat karakter #
</script>
```

e. Kata Kunci (Reserved Keyword) JavaScript

Seperti bahasa pemograman lain, JavaScript juga memiliki beberapa kata kunci atau keyword yang tidak bisa digunakan sebagai nama variabel atau nama dari sebuah fungsi. Istilah ini sering disebut sebagai reserved keyword.

Reserved keyword merupakan kata kunci yang digunakan JavaScript dalam menjalankan fungsinya.

Keyword di dalam JavaScript adalah sebagai berikut:

abstract	arguments	boolean	break	byte
case	catch	char	class*	const
continue	debugger	default	delete	do
double	else	enum*	eval	export*
extends*	false	final	finally	float
for	function	goto	if	implements
import*	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super*	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	volatile
while	with	yield		

object properties method:

Array	Date	eval	function	hasOwnProperty
Infinity	isFinite	isNaN	isPrototypeOf	length
Math	NaN	name	Number	Object
prototype	String	toString	undefined	valueOf

HTML dan windows object dan properties

alert	all	anchor	anchors	area
assign	blur	button	checkbox	clearInterval
clearTimeout	clientInformation	close	closed	confirm
constructor	crypto	decodeURI	decodeURIComponent	defaultStatus
document	element	elements	embed	embeds
encodeURI	encodeURIComponent	escape	event	fileUpload
focus	form	forms	frame	innerHeight
innerWidth	layer	layers	link	location
mimeTypes	navigate	navigator	frames	frameRate
hidden	history	image	images	offscreenBuffering
open	opener	option	outerHeight	outerWidth
packages	pageXOffset	pageYOffset	parent	parseFloat
parseInt	password	pkcs11	plugin	prompt
propertyIsEnum	radio	reset	screenX	screenY
scroll	secure	select	self	setInterval
setTimeout	status	submit	taint	text
textarea	top	unescape	untaint	window

HTML event handler

onblur	onclick	onerror	onfocus
onkeydown	onkeypress	onkeyup	onmouseover
onload	onmouseup	onmousedown	onsubmit

Sebagian dari nama keyword diatas bisa digunakan dalam situasi khusus (seperti nama dari method untuk objek), namun sedapat mungkin kita tidak menggunakannya agar tidak menimbulkan masalah di kemudian hari.

f. Aturan Penulisan Tanda Semicolon pada Akhir Baris

Berbeda dari kebanyakan bahasa pemograman, di dalam **JavaScript** karakter titik-koma (bahasa inggris: **semicolon**) sifatnya **opsional** untuk digunakan sebagai penanda akhir dari baris program, dan boleh tidak ditulis.

JavaScript ‘mendeteksi’ baris baru (karakter ‘*break’*) sebagai penanda akhir baris program. Kode perintah berikut berjalan sebagaimana mestinya di dalam JavaScript:

```
!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Belajar JavaScript</title>

<script>
   a=13
   b=a+2
   console.log(b)
   alert(b)
</script>

</head>
<body>
<h1>Belajar JavaScript</h1>
<p> Saya sedang belajar JavaScript di Informatika Upgris</p>
</div>
</body>
</html>
```

Perhatikan bahwa di dalam tag <script> saya tidak menggunakan tanda semicolon untuk menutup baris perintah JavaScript, dan kode tersebut berjalan sukses tanpa menghasilkan error. Namun javascript tidak akan 'mendeteksi' setiap baris baru sebagai penanda akhir baris program, perhatikan kode berikut ini:

```
<script>
   var a
   a
   =
   15
   console.log(a)
</script>
```

Kode tersebut akan diproses oleh JavaScript menjadi:

```
<script>
   var a; a = 15; console.log(a)
</script>
```

Akan tetapi, penulisan berikut ini:

```
<script>
   a = 5
   b = 3 + a
   (b + 10).toString()
</script>
```

Akan diproses menjadi:

```
<script>
    a = 5 ; b = 3 + a(b + 10).toString();
</script>
```

Hal ini terjadi karena JavaScript akan menganggap baris baru sebagai tanda semicolon (pemisah baris program) apabila pada saat memproses baris berikutnya sudah tidak mungkin 'bersambung'. Penggunaan karakter pembuka kurung "(" di awal baris baru, akan dianggap sebagai bagian sambungan dari baris sebelumnya.

Penggunaan tanda semicolon ";" walau bersifat opsional, namun sangat dianjurkan digunakan. Tanda semicolon akan membuat program lebih mudah dibaca dan tidak membuat ambigu seperti contoh kita diatas. Di dalam tutorial JavaScript di duniailkom ini, saya juga akan menggunakan tanda ";" pada setiap akhir baris.

6. Mengenal Variabel JS

a. Pengertian dan Sifat Variabel di Dalam JavaScript

Dalam bahasa pemograman, **variabel** adalah '*penampung'* sebuah nilai. Tergantung dengan '*nilai'*dari **variabel** tersebut, sebuah *variabel* di dalam JavaScript dapat bertipe Angka (N**umber**), **String**,**Boolean**, atau yang lainnya. Tidak seperti bahasa pemograman desktop seperti **C++** dan **Visual Basic**, di dalam **JavaScript** kita tidak perlu mendeklarasikan jenis tipe data. Seluruh **variabel** di dalam JavaScript dapat berisi nilai apapun (tipe data apapun), dan dapat diubah menjadi tipe lain sepanjang program. Jenis pemograman seperti ini dikenal juga dengan **Typeless Programming Language**. Salah satu bahasa pemograman yang juga berjenis **Typeless Programming Language** adalah**PHP.**

b. Aturan Penamaan Variabel JavaScript

Aturan penamaan variabel pernah kita bahas yakni sama dengan aturan pembuatan identifier:

- Karakter pertama harus diawali dengan huruf, underscore (_) atau tanda dollar ($)
- Karakter kedua dan seterusnya bisa ditambahkan dengan huruf, angka, underscore (_) atau tanda dollar ($).

c. Cara Membuat Variabel JavaScript

Walaupun kita tidak perlu menyebutkan jenis tipe data dari suatu variabel, namun kita tetap harus mendeklarasikan variabel di dalam JavaScript. Cara membuat variabel di dalam JavaScript di bedakan menjadi 2, yakni dengan menggunakan keyword var, dan tanpa var.

Jika menggunakan kata kunci var, berikut adalah contoh penulisannya:

```
<script>
  var a;
  var b, c, d;
  var e; var f;
  var g=12;
  var h="Saya Sedang Belajar JavaScript di Informatika Upgris";
</script>
```

Pada 2 baris terakhir saya membuat *variabel*, sekaligus memberikan nilai ke dalam variabel tersebut.

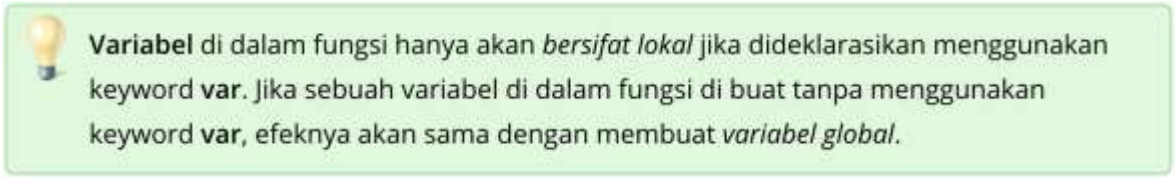
Cara kedua untuk membuat variabel adalah tanpa menggunakan keyword **var**, seperti berikut ini:

```
<script>
  a;
  b=12;
  c=" Saya Sedang Belajar JavaScript di Informatika Upgris";
</script>
```

Pembuatan variabel tanpa menggunakan keyword var memang lebih cepat, akan tetapi tidak disarankan. Walaupun variabel yang dideklarasikan tanpa keyword var akan tetap berfungsi sebagaimana mestinya seperti variabel dengan var, namun JavaScript ‘menyimpan’ variabel tersebut dengan cara yang berbeda. Salah satu perbedaannya adalah tentang variabel scope (yang akan kita bahas setelah ini). Sedapat mungkin kita selalu membuat variabel menggunakan keyword var.

d. Jangkauan Variabel (Variabel Scope) dalam JavaScript

Jangkauan Variabel (atau Variabel Scope) adalah konsep tentang pembatasan akses dari sebuah variabel. Yaitu pada bagian mana sebuah variabel masih bisa diakses. Sebuah variabel jika dideklarasikan (baik dengan keyword var ataupun tanpa var), akan bersifat global, atau dikenal dengan istilah global variable. Sebuah variabel akan menjadi global variabel sepanjang variabel tersebut di deklarasikan di luar fungsi. Jika sebuah variabel di deklarasikan di dalam fungsi, maka variabel tersebut hanya akan bisa diakses di dalam fungsi tersebut, atau bersifat lokal (dikenal juga dengan local variable). Apabila kita membuat 2 variabel dengan nama yang sama sebagai global variabel, dan juga local variable di dalam sebuah fungsi, maka local variable akan memiliki prioritas yang lebih tinggi dibandingkan global variabel.




```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Belajar JavaScript</title>

<script>
var nilai = "global";
function test() {
   var nilai = "lokal";
   var nilai_lokal = "duniailkom";
   tanpa_var = "no_scope"; //akan menjadi global variabel!!
   console.log(nilai);
   }

test(); // print: lokal
console.log(nilai); // print: global
console.log(tanpa_var); //print: no_scope (bisa diakses)
console.log(nilai_lokal); //error, karena nilai_lokal adalah lokal variabel
</script>

</head>
<body>
<h1>Belajar JavaScript</h1>
<p> Saya sedang belajar JavaScript di Informatika Upgris </p>
</body>
</html>
```

Belajar JavaScript

Saya sedang belajar JavaScript di Informatika Upgris