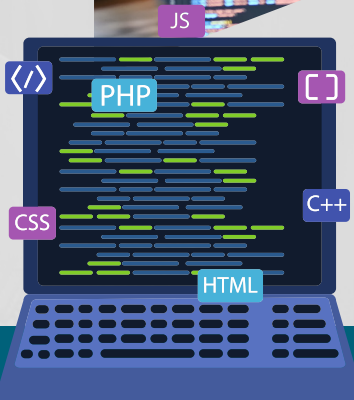
ISBN 978-623-196-189-9

PEMROGRAMAN WEB



Leo Willyanto Santoso
Sulistyowati
Wahyuddin S
Diana
Andy Rachman
Ronny Makhfuddin Akbar
Eferoni Ndruru
Mundzir

PEMROGRAMAN WEB

Leo Willyanto Santoso
Sulistyowati
Wahyuddin S
Diana
Andy Rachman
Ronny Makhfuddin Akbar
Eferoni Ndruru
Mundzir

PT GLOBAL EKSEKUTIF TEKNOLOGI

PEMROGRAMAN WEB

Penulis :
Leo Willyanto Santoso
Sulistyowati
Wahyuddin S
Diana
Andy Rachman
Ronny Makhfuddin Akbar
Eferoni Ndruru
Mundzir

ISBN : 978-623-198-189-9

Editor : Dina ediana, S. Kom., M. Kom.
Penyunting : Ari Yanto, M.Pd
Desain Sampul dan Tata Letak : Atyka Trianisa, S.Pd

Penerbit : PT GLOBAL EKSEKUTIF TEKNOLOGI
Anggota IKAPI No. 033/SBA/2022

Redaksi :
Jl. Pasir Sebelah No. 30 RT 002 RW 001
Kelurahan Pasie Nan Tigo Kecamatan Koto Tangah
Padang Sumatera Barat
Website : www.globaleksekutifteknologi.co.id
Email : globaleksekutifteknologi@gmail.com

Cetakan pertama, April 2023

KATA PENGANTAR

Segala Puji dan syukur atas kehadirat Allah SWT dalam segala kesempatan. Sholawat beriring salam dan doa kita sampaikan kepada Nabi Muhammad SAW. Alhamdulillah atas Rahmat dan Karunia-Nya penulis telah menyelesaikan Buku Pemrograman Web ini.

Buku Ini Membahas Javascript (JS), Variabel, Konversi type data dengan JS, Pencabangan IF, Operator dengan JS, Function dalam pemrograman web dengan JS, Operasi Aritmatika dengan PHP, Perulangan.

Proses penulisan buku ini berhasil diselesaikan atas kerjasama tim penulis. Demi kualitas yang lebih baik dan kepuasan para pembaca, saran dan masukan yang membangun dari pembaca sangat kami harapkan.

Penulis ucapkan terima kasih kepada semua pihak yang telah mendukung dalam penyelesaian buku ini. Terutama pihak yang telah membantu terbitnya buku ini dan telah mempercayakan mendorong, dan menginisiasi terbitnya buku ini. Semoga buku ini dapat bermanfaat bagi masyarakat Indonesia.

Padang, April 2023

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
JAVASCRIPT

Oleh Leo Willyanto Santoso

1.1 Sejarah Javascript

HTML versi pertama yang dirancang oleh Tim Berners-Lee dari tahun 1989 hingga 1991 tergolong statis, halaman web hanya menampilkan konten yang sudah fix. Pada tahun 1995, perusahaan browser yang dominan adalah Netscape, dan salah satu karyawannya, Brendan Eich, berpendapat bahwa menambahkan fungsionalitas dinamis ke halaman web akan berguna. Jadi dia merancang bahasa pemrograman JavaScript, yang menambahkan fungsionalitas dinamis ke halaman web saat digunakan bersamaan dengan HTML. Misalnya, JavaScript menyediakan kemampuan untuk memperbarui konten halaman web saat terjadi peristiwa/*event* tertentu, seperti saat pengguna mengklik tombol. Javascript juga menyediakan kemampuan untuk mengambil input pengguna dan memproses input tersebut.

Hanya butuh 10 hari bagi Eich di bulan Mei 1995 untuk mengimplementasikan bahasa pemrograman JavaScript—prestasi yang benar-benar luar biasa. Marc Andreessen, salah satu pendiri Netscape, awalnya menamai bahasa baru itu Mocha dan kemudian LiveScript. Namun untuk tujuan pemasaran, Andreessen memberi nama JavaScript. Pada saat itu, industri perangkat lunak sangat antusias dengan bahasa pemrograman baru yaitu Java. Andreessen membayangkan bahwa semua penggemar Java akan tertarik pada bahasa pemrograman browser baru mereka jika ada nama Java di dalamnya. Pada Desember 1995, Andreessen mendapatkan keinginannya ketika Netscape memperoleh lisensi merek dagang dari pembuat Java, Sun Microsystems, dan nama LiveScript diubah

menjadi JavaScript. Sayangnya, banyak, banyak orang selama bertahun-tahun membuat kesalahan dengan berasumsi bahwa JavaScript sama dengan Java atau sangat dekat dengannya. Jangan terkecoh dengan namanya—JavaScript tidak terlalu mirip dengan Java. Sebenarnya, C++ dan bahasa pemrograman populer lainnya lebih dekat ke Java daripada JavaScript.

Pada tahun 1996, Netscape mengirimkan JavaScript ke organisasi standar Ecma International, mempromosikan JavaScript di semua browser (bukan hanya browser Netscape). Ecma International menggunakan JavaScript sebagai dasar untuk membuat standar ECMAScript. Seperti yang diharapkan, ECMAScript sekarang berfungsi sebagai standar untuk bahasa pemrograman interaktif yang disematkan di semua browser populer saat ini. Pada saat pencetakan buku ini, versi ECMAScript terbaru adalah versi 12, diterbitkan pada tahun 2021.

Pada tahun 1998, Netscape membentuk komunitas perangkat lunak bebas Mozilla, yang akhirnya mengimplementasikan Firefox, salah satu browser terkemuka saat ini. Brendan Eich pindah ke Mozilla, di mana dia dan yang lainnya terus memperbarui JavaScript selama bertahun-tahun, mengikuti standar ECMAScript sebagaimana ditetapkan oleh Ecma International.

Produsen browser lain mendukung versi JavaScript mereka sendiri. Untuk browser Edge, Microsoft menggunakan JScript. Untuk browser Chrome, Google menggunakan V8 JavaScript Engine. Untungnya, semua produsen browser berusaha mengikuti standar ECMAScript, jadi untuk sebagian besar, pemrogram dapat menulis satu versi kode mereka dan itu akan berfungsi untuk semua browser yang berbeda. Dalam buku ini, kami tetap menggunakan kode ECMAScript standar yang berfungsi sama di semua browser.

1.2 Menampilkan Hello World

Saat mempelajari bahasa pemrograman baru, program pertama Anda seharusnya hanya mencetak "Hello, world!". Karena tugas program sangat sederhana, kodenya pendek dan memberikan kesempatan kepada pembelajar untuk fokus pada dasar-dasar sintaksis dan tidak terjebak dengan terlalu banyak detail. Gambar 1.1, menampilkan "Hello, world!" ketika pengguna mengklik tombol.

Gambar 1.1 : Program Hello World

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="author" content="Leo">
<title>Hello</title>
<script>
 function tampilHello() {
 var msg;
 msg = document.getElementById("message");
 msg.outerHTML = "<h1>Hello, world!</h1>";
 }
</script>
</head>
<body>
<h3 id="message">
 Program Pertama Javascript
</h3>
<input type="button" value="Click Me!"
onclick="tampilHello();">
</body>
</html>
```

1.3 Button

Ada berbagai jenis button, masing-masing dengan sintaksnya sendiri. Untuk menjaga hal-hal sederhana, kita akan mulai hanya dengan satu jenis button, dan inilah sintaksnya:

```
<input type="button"
 value="button-label"
 onclick="click-event-handler">
```

Perhatikan bahwa kode tersebut adalah elemen kosong yang menggunakan tag input. Seperti namanya, tag input mengimplementasikan elemen yang menangani input pengguna. Anda mungkin tidak menganggap tombol sebagai input pengguna, tetapi memang demikian—pengguna memilih untuk melakukan sesuatu dengan mengklik tombol. Nanti, kami akan

memperkenalkan elemen input pengguna lain (mis., kontrol teks dan kotak centang) yang juga menggunakan tag input.

Atribut nilai elemen input menentukan label tombol. Jika Anda tidak memberikan atribut nilai, tombol tidak akan memiliki label. Jika Anda menginginkan tombol tanpa label, bukan hanya menghilangkan atribut value, kami sarankan Anda menentukan value="". Itu adalah bentuk dokumentasi, dan itu membuat kode Anda lebih mudah dipahami.

Atribut onclick elemen input menentukan instruksi JavaScript yang dieksekusi mesin JavaScript saat pengguna mengklik tombol. Apa itu mesin JavaScript? Mesin JavaScript adalah bagian dari perangkat lunak browser yang menjalankan JavaScript halaman web. Di halaman web Hello, nilai atribut onclick adalah kode JavaScript yang "menangani" apa yang seharusnya terjadi saat pengguna mengklik tombol. Mengklik tombol dianggap sebagai event, sehingga kode JavaScript atribut onclick dikenal sebagai event handler. Selain onclick, ada atribut lain yang memanggil event handler, seperti onfocus dan onload, tetapi atribut tersebut tidak banyak digunakan untuk tombol, jadi atribut tersebut tidak akan diperkenalkan.

Dalam *source code* halaman web Hello, perhatikan bahwa nilai atribut onclick hanyalah tampilHello();. Fungsi memanggil function tampilHello, yang ditentukan dalam blok skrip halaman web. Kita akan membahas pemanggilan fungsi dan definisi fungsi di bagian berikutnya.

1.4 *Function*/Fungsi

Fungsi dalam JavaScript mirip dengan fungsi matematika. Fungsi matematika menerima argumen, melakukan perhitungan, dan mengembalikan jawaban. Misalnya, fungsi matematika sin(x) menerima argumen x, menghitung sinus dari sudut x yang diberikan, dan mengembalikan sinus yang dihitung dari x. Demikian pula, fungsi JavaScript menerima argumen, melakukan

perhitungan, dan mengembalikan jawaban. Berikut sintaks untuk memanggil suatu fungsi:
nama-*function(arguments*);

Seperti disebutkan sebelumnya, tombol halaman web Hello memiliki atribut onclick dengan nilai tampilHello();. Itu adalah panggilan fungsi JavaScript. Perhatikan bahwa tanda kurung kosong karena tidak perlu meneruskan nilai argumen apa pun ke fungsi tampilHello. Jika ada argumen, mereka perlu dipisahkan dengan koma.
Berikut sintaks untuk definisi fungsi:

```
function nama-function (parameters) {
 statement-1;
 statement-2;
 …
 last-statement;
}
```

1.5 Variable

Mari kita terus bekerja melalui kode di fungsi halaman web Hello. Inilah pernyataan pertama fungsi:

```
var msg;
```

Hal pesan adalah variabel. Anda seharusnya sudah familiar dengan variabel dalam aljabar. Anda dapat menganggap variabel sebagai kotak yang menyimpan nilai. Dalam hal ini, variabel msg akan menyimpan string yang membentuk pesan.

Sebelum Anda menggunakan variabel dalam JavaScript, Anda harus menggunakan var untuk mendeklarasikan variabel dalam pernyataan deklarasi. Sebagai contoh:

```
var nama;
var tinggi;
```

Kata-kata yang merupakan bagian dari bahasa JavaScript dikenal sebagai kata kunci. Kata var adalah kata kunci. Di sisi lain, nama dan tinggi bukanlah kata kunci karena spesifik untuk kode

halaman web tertentu dan bukan untuk bahasa JavaScript secara umum.

Di bagian sebelumnya, kami menunjukkan fungsi tampilHello. Dalam fungsi itu, selain var, dapatkah Anda mengidentifikasi kata kunci lain? Judul fungsi menggunakan kata function dan function adalah kata kunci. Dengan sebagian besar kata kunci JavaScript, adalah ilegal bagi Anda sebagai programmer untuk mendefinisikan ulang kata kunci tersebut agar memiliki arti lain. Jadi Anda tidak bisa menggunakan "function" sebagai nama variabel. Kata kunci tersebut "di-*reserved*" untuk bahasa JavaScript, dan dikenal sebagai *reserved word*. Ada beberapa kata kunci yang dapat didefinisikan ulang oleh seorang programmer, tetapi penjelasannya berada di luar cakupan buku ini. Banyak pemrogram menggunakan istilah "kata kunci" dan "kata cadangan" secara bergantian, tetapi ada sedikit perbedaan, karena tidak semua kata kunci merupakan kata kunci.

Di sebagian besar bahasa pemrograman, saat Anda mendeklarasikan variabel, Anda menentukan jenis nilai yang diizinkan untuk disimpan oleh variabel—angka, string, dan seterusnya. Namun, dengan JavaScript, Anda tidak menentukan tipe variabel sebagai bagian dari deklarasi. Jenis variabel ditentukan secara dinamis oleh jenis nilai yang diberikan ke dalam variabel. Sebagai contoh:

```
nama = "Leo Willyanto";
tinggi = 174;
```

Apa jenis nilai "Leo Willyanto”? Sebuah string, karena sebuah string terdiri dari nol atau lebih karakter yang diapit oleh sepasang tanda kutip ganda ("") atau sepasang tanda kutip tunggal ('). Apa jenis nilai 174? Angka, karena angka terdiri dari angka dengan titik desimal bersifat opsional. JavaScript dikenal sebagai bahasa yang dituliskan secara longgar, atau bahasa yang dituliskan

secara dinamis, yang berarti bahwa Anda tidak mendeklarasikan tipe data variabel secara eksplisit, dan Anda dapat menetapkan berbagai jenis nilai ke dalam variabel pada waktu yang berbeda selama eksekusi sebuah program. Misalnya, adalah legal untuk menetapkan string ke nama dan kemudian menetapkan nomor ke nama. Tetapi konvensi pengkodean yang tepat menentukan bahwa Anda tidak melakukan itu, karena dapat menyebabkan kode yang sulit dipahami.

1.6 Identifier

Identifier adalah istilah teknis untuk nama komponen program—nama fungsi, nama variabel, dan nama komponen program lain yang akan kita bahas nanti. Di halaman web Hello, tampilHello adalah identifier untuk nama fungsi, dan msg adalah identifier untuk variabel. Dalam menamai variabel dan fungsi Anda, mesin JavaScript mengharuskan Anda mengikuti aturan tertentu. *Identifier* harus seluruhnya terdiri dari huruf, angka, tanda dolar ($), dan/atau garis bawah (_). Karakter pertama tidak boleh berupa angka. Jika Anda tidak mengikuti aturan ini, kode JavaScript Anda tidak akan berfungsi.

Aturan konvensi pengkodean lebih sempit dari aturan sebelumnya. Konvensi pengkodean menyarankan agar Anda menggunakan huruf dan angka saja, bukan tanda dolar atau garis bawah. Mereka juga menyarankan *identifier* semua harus huruf kecil kecuali huruf pertama pada kata kedua, kata ketiga, dan seterusnya. Hal ini disebut sebagai *camel case*, dan berikut adalah beberapa contohnya: firstName, message, daysInMonth.

Perhatikan bahwa penamaan *identifier* bersifat deskriptif. Konvensi pengkodean menyarankan agar Anda menggunakan kata-kata deskriptif untuk *identifier* Anda. Pemrogram pemula cenderung menggunakan nama seperti x, y, dan num. Biasanya, itu adalah nama variabel yang buruk. Namun, jika Anda memiliki situasi di mana Anda seharusnya membaca dalam angka dan angka

tersebut tidak mewakili sesuatu yang istimewa, maka x atau angka tidak apa-apa.

1.7 Assignment Statement dan Object

Sekarang, kita akan melihat bagaimana halaman web dapat menggantikan pesan awal dengan "Hello, world!" ketika pengguna mengklik tombol. Untuk memahami cara kerjanya, kita perlu berbicara tentang *assignment* dan *object statement*.

Sekali lagi, inilah fungsi displayHello halaman web Hello:

```
function tampilHello() {
 var msg;
 msg = document.getElementById("message");
 msg.outerHTML = "<h1>Hello, world!</h1>";
}
```

Di badan fungsi, pernyataan pertama adalah deklarasi variabel untuk variabel msg. Setelah Anda mendeklarasikan variabel, Anda akan ingin menggunakannya, dan langkah pertama dalam menggunakan variabel adalah memasukkan nilai ke dalamnya. *Assignment statement* menempatkan/memberikan nilai ke dalam variabel. Seperti yang Anda lihat pada contoh sebelumnya, pernyataan kedua dan ketiga badan fungsi adalah *assignment statement*.

Operator *assignment* (=) memberikan nilai di sebelah kanan ke dalam variabel di sebelah kiri. Jadi dalam pernyataan penugasan pertama, objek document.getElementById("message") ditugaskan ke variabel msg. Dalam *assignment statement* kedua, "<h1>Hello, world!</h1>" ditugaskan ke variabel msg.outerHTML.

Kedua *assignment statement* itu cukup membingungkan. Untuk memahami sintaks diperlukan pemahaman tentang objek. Objek adalah entitas perangkat lunak yang mewakili sesuatu yang nyata.

Di belakang layar, semua elemen di halaman web direpresentasikan sebagai objek. Saat browser memuat halaman

web Hello, perangkat lunak browser menghasilkan objek untuk elemen head, elemen body, elemen h3, dan seterusnya. Ada juga objek yang terkait dengan seluruh halaman web, dan nama objek itu adalah document. Setiap objek memiliki satu set properti terkait, ditambah satu set *behavior*. Properti adalah atribut dari suatu objek. *Behavior* adalah tugas yang dapat dilakukan objek. Objek document berisi properti seperti tipe halaman web. Sebagian besar halaman web saat ini (dan semua halaman web dalam buku ini) memiliki nilai HTML5 untuk properti tipe objek document. Namun dimungkinkan untuk memiliki tipe lain, seperti HTML 4.01 atau XHTML 1.0 Strict. Nilai properti type berasal dari instruksi doctype, yang akan muncul di bagian atas setiap halaman web. Inilah instruksi doctype halaman web Hello:

```
<!DOCTYPE html>
```

Nilai html menunjukkan bahwa tipe objek document adalah HTML5. Untuk mengakses properti objek, tentukan nama objek, titik, lalu nama properti. Jadi untuk mengakses dokumen halaman web, gunakan document untuk nama objek, . untuk dot, dan doctype untuk properti. Ini kode JavaScriptnya:

```
document.doctype
```

Ingatlah bahwa objek bukan hanya sekumpulan properti, tetapi juga sekumpulan *behavior*. Salah satu *behavior* objek document adalah kemampuannya untuk mengambil elemen menggunakan nilai id elemen. Dalam JavaScript (dan juga banyak bahasa pemrograman lainnya), *behavior* objek disebut sebagai metode. Untuk mengambil elemen, objek document menggunakan metode getElememementById. Untuk memanggil metode objek, Anda menentukan nama objek, titik, nama metode, dan kemudian tanda kurung di sekitar argumen yang ingin diteruskan ke metode. Misalnya, inilah pemanggilan metode getElememementById dari fungsi tampilHello pada halaman web Hello:

```
dokumen.getElementById("message")
```

Lihat bagaimana pemanggilan metode menyertakan "message" untuk argumennya? Dalam mengeksekusi pemanggilan metode, mesin JavaScript mencari elemen dengan id="message". Ada elemen seperti itu di halaman web Hello, dan ini dia:

```
<h3 id="message">
 Program Pertama Javascript
</h3>
```

Jadi panggilan metode getElementById mengambil elemen h3 itu.

Standar HTML5 mengatakan bahwa nilai atribut id harus unik untuk halaman web tertentu. Anda mungkin ingat bagaimana kami menggunakan atribut id untuk mengidentifikasi target tautan di dalam halaman web. Penggunaan atribut id diperlukan dalam situasi itu karena kita memerlukan target tautan agar unik. Demikian pula, kami menggunakan atribut id untuk mengambil elemen (dengan getElementById) sehingga tidak akan ada kebingungan dalam hal elemen mana yang akan diambil.

DAFTAR PUSTAKA

Dean, J. 2019. Web Programming with HTML 5, CSS and Javascript. Jones & Bartlett Learning.

Flanagan, D. 2020. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language. 7th Ed. O'Reilly Media.

https://www.w3schools.com/js/default.asp.

Lars Svekis, L., van Putten, M., dan Percival, R. 2021. JavaScript from Beginner to Professional: Learn JavaScript quickly by building fun, interactive, and dynamic web apps, games, and pages. Packt Publishing.

Minnick, C. dan Holland, E. 2015. Coding with JavaScript For Dummies. 1st Edition. For Dummies.

BAB 2
VARIABEL

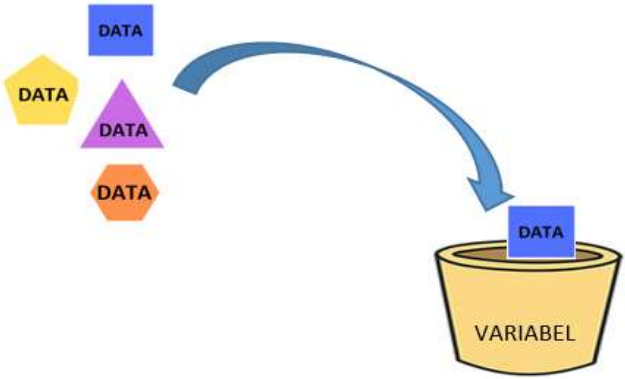
Oleh Sulistyowati

2.1 Pendahuluan

Variabel adalah komponen yang sangat penting dalam sebuah pemrograman. Pada pemrograman, ada beberapa aktivitas yang terjadi, yaitu menerima inputan, melakukan pemrosesan, dan selanjutnya menghasilkan output. Saat melakukan inputan, berarti yang dilakukan adalah memasukkan nilai (data) dari piranti input. Nilai tersebut bisa berupa nilai numerik maupun nilai string atau karakter. Pada tahap ini dibutuhkan bantuan variabel untuk menampung atau menyimpan nilai tersebut (Ullman, 2012). Selanjutnya tahap pemrosesan, yang dilakukan adalah pengolahan nilai atau data yang sudah tersimpan dalam variabel. Dan Sebagai akhir proses pemrograman adalah output, yaitu menampilkan hasil pemrograman ke piranti output.

2.2 Definisi Variabel

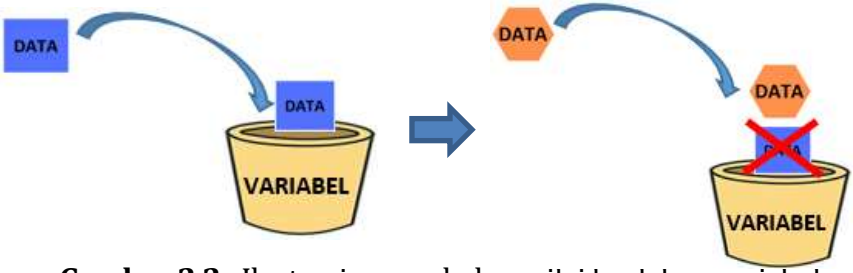
Variabel adalah sebuah tempat untuk menyimpan nilai atau data pada program. Nilai tersebut dapat berubah-ubah selama program tersebut dieksekusi. Variabel merupakan tempat penyimpanan sementara data yang sedang diolah. Kata variabel berarti 'dapat berubah' dan digunakan karena variabel dapat menyimpan berbagai jenis nilai dan dapat mengubah nilainya berkali-kali (Ogden, 2015). Ilustrasi variabel dapat dilihat pada Gambar 2.1 :



Gambar 2.1 : Ilustrasi variabel
(Sumber : Pribadi)

Varibel bersifat sementara (tidak permanen), artinya setiap nilai atau data yang tersimpan dalam variabel akan dapat diakses hanya pada saat program berjalan dan akan hilang pada saat program tersebut berhenti atau dimatikan. Perlu diingat bahwa sebuah variabel hanya dapat menyimpan satu nilai atau data.

Bagaimana jika kita memasukkan lebih dari satu nilai atau data ke dalam sebuah variabel? Jawabnnya adalah nilai atau data yang akan tersimpan dalam variabel tersebut adalah nilai atau data yang paling akhir dimasukkan. Saat nilai atau data dalam sebuah variabel diubah, maka nilai atau data lama akan dihapus dari variabel (Perhatikan Gambar 2.2 berikut :



Gambar 2.2 : Ilustrasi penambahan nilai ke dalam variabel
(Sumber : pribadi)

Gambar 2.2 menunjukkan bahwa mula-mula variabel menyimpan data berwarna biru. Karena dimasukkan lagi data berwarna orange ke variabel tersebut, maka data sekarang yang tersimpan dalam variabel adalah data orange, bukan data biru.

2.3 Penamaan Variabel

Setiap variabel dalam permograman harus memiliki identifier atau nama yang unik. Artinya, jika dalam pemrograman menggunakan banyak variabel maka setiap variabel tersebut harus memiliki identifier atau nama yang berbeda-beda.

Aturan penamaan variabel pada JavaScript :

1. Karakter pertama harus berupa huruf (tidak boleh numerik), garis bawah (_), atau dollar ($) (Flanagan, 2011)
   Contoh : Nama1
   _Nama
   $nama
   Nama
   namaPasien1
2. Jika nama variabel terdiri dari dua kata atau lebih, maka menggunakan format *snake case*. Yaitu penulisannya tidak boleh dipisah dengan spasi, harus menggunakan garis bawah
   (_).
   Contoh : nama_pasien
   Gaji_Pokok
3. Hanya boleh terdiri dari alfanumerik (huruf A – z dan angka 0 – 9) dan garis bawah.
4. Tidak boleh menggunakan kata-kata kunci (*keyword*) dalam JavaScript.
   Kata kunci dalam JavaScript dapat dilihat pada Tabel 2.1 :

Tabel 2.1: Kata kunci dalam Javascript

abstract	arguments	await	boolean
break	byte	case	catch
char	class	const	continue
debugger	default	delete	do
double	else	enum	eval
export	extends	false	final
finally	float	fir	function
goto	if	impelements	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

(Sumber : w3schools.com)

Karena Javascript juga dapat digunakan sebagai bahasa pemrograman di banyak aplikasi, maka hindari juga penggunaan nama objek dan properti HMTL dan Window, seperti pada Tabel 2.2.

Tabel 2.2 : Kata kunci objek dan properti HTML dan Window

allert	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

(Sumber : w3schools.com)

Biasanya untuk pemberian nama variabel pada JavaScript adalah dengan format *camel case*. Yaitu cara penulisan variabel yang terdiri dari dua kata atau lebih, dimana huruf pertama di setiap kata ditulis dengan huruf kapital, kecuali kata pertama.

Contoh : nipPegawai
totalBiaya
namaAnggotaKeluarga

Ingat, itu hanyalah kebiasaan saja dan bukan suatu kewajiban. Progam JavaScript tidak akan menolak jika programmer tetap ingin membuat nama variabel dengan menggunakan huruf besar semua, atau setiap kata diawali huruf besar, atau setiap kata dipisah dengan garis bawah.

2.4 Pendeklarasian Variabel

Mendeklarasikan variabel berarti mengumumkan keberadaan variabel tersebut secara formal. Dalam beberapa bahasa pemrograman, seperti C, setiap variabel yang digunakan dideklarasikan sebelum mereferensikannya. Namun dalam JavaScript tidak mengharuskan untuk mendeklarasikan variabel terlebih dahulu dan dapat langsung mereferensikannya, seperti pada contoh berikut :

Terdapat 3 cara untuk mendeklarasikan variabel, yaitu:

1. **Menggunakan kata kunci var**

   Sintaks :

   var nama_variabel;

   Berikut contoh deklarasi variabel dengan kata kunci var :

```
var gaji_pokok;
var nama, umur;
```

   Pada pendeklarasian variabel tersebut, variabel akan bernilai *undifined* (belum ditentukan). Hal ini karena variabel tersebut belum diisi nilai atau data.

Kita bisa mengisikan nilai atau data ke dalam variabel tersebut seperti contoh berikut :

```
var gaji_pokok;
var nama, umur;
gaji_pokok = 2000000;
nama =“Azzahra Aulia”;
umur = 18;
```

Pemberian atau pengisian nilai atau data ke dalam variabel tersebut disebut sebagai inisialisasi atau pemberian nilai awal.

Namun juga bisa dilakukan deklarasi variabel sekaligus menginisialisasi variabel tersebut, seperti pada contoh berikut :

```
var gaji_pokok = 2000000;
var nama =“Azzahra Aulia”;
var umur = 18;
```

Nilai dari variabel yang sudah dideklarasikan dengan menggunakan kata kunci **var** dapat diubah. Nilai yang diinputkan boleh sama atau berbeda tipe datanya. Perhatikan contoh berikut :

```
var nama =“Azzahra Aulia”;
nama = “Naufal Rafif”;
console.log(nama);
```

Pada contoh diatas, variabel nama semula bernilai “Azzahra Aulia”. Namun pada baris dibawahnya, isi dari variabel nama diubah menjadi “Naufal Rafif”, dimana tipe data kedua nilai

tersebut adalah sama, yaitu string. Dan apabila nilai dari variabel nama dicetak atau ditampilkan, maka akan menghasilkan output : Naufal Rafif.
Perhatikan juga contoh berikut :

```
var gaji_pokok = 2000000;
gaji_pokok =“Dua Juta”;
cosole.log(gaji_pokok);
```

Pada contoh diatas, variabel gaji_pokok semula bernilai 200000, dimana tipe datanya adalah number. Namun kemudian isi dari variabel gaji_pokok diubah menjadi “Dua Juta” yang bertipe string. Sehingga jika nilai dari variabel gaji_pokok dicetak atau ditampilkan, maka akan menghasilkan output : Dua Juta.

2. Menggunakan kata kunci let

Sintaks :

let nama_variabel;

Berikut contoh deklarasi variabel dengan kata kunci **let** :

```
let harga = 14700;
let alamat =“Jl. Sedayu No. 34”;
```

3. Menggunakan kata kunci const

Sintaks :

const nama_variabel;

Berikut contoh deklarasi variabel dengan kata kunci **const** :

```
const harga = 14700;
const tanggal = “14 Januari 2023”;
```

2.5 Perbedaan var, let, dan const

Untuk membedakan pendeklarasian variabel dengan menggunakan **var**, **let**, dan **const**, dapat dilihat dari *scope*-nya, penggunaan-nya, dan *hoisting*-nya. Berikut penjelasannya :

1. **Sistem *scope***

   Scope adalah cakupan variabel tersebut akan digunakan. Ada dua jenis scope, yaitu *local scope* dan *global scope*. Jika sebuah variabel berada dalam *global scope*, berarti variabel tersebut dideklarasikan di luar fungsi atau blok program, sehingga dapat diakses oleh fungsi atau blok lain di dalam program.

   Sedangkan variabel yang berada dalam *local scope*, berarti variabel tersebut dideklarasikan dalam sebuah fungsi atau blok. Sehingga variabel tersebut hanya dapat diakses dalam fungsi atau blok dimana variabel dideklarasikan.

   Local scope dibagi menjadi dua, yaitu *function scope* dan *block scope. Function scope* berarti variabel yang didefinisikan dalam suatu fungsi tidak dapat diakses dari luar fungsi. *Block scope* berarti bahwa variabel yang didefinisikan di dalam blok tidak akan dapat diakses dari luar blok. Sebuah blok dapat berada di dalam sebuah fungsi, dan variabel yang berada dalam *block scope* tidak akan tersedia di luar blok bahkan jika blok tersebut berada di dalam sebuah fungsi. Blok yang dimaksud adalah kurung kurawal ({}).

   Misal variabel dideklarasikan dalam sebuah blok perulangan, maka variabel tersebut hanya dapat diakses dalam blok perulangan tersebut. Jika kita coba mengakses variabel tersebut di luar blok perulangan, maka akan mengakibatkan kesalahan. Demikian juga variabel yang dideklarasikan dalam fungsi, hanya dapat diakses di dalam fungsi tersebut.

Berikut adalah contoh pendeklarasian variabel dalam blok dengan mengunakan kata kunci **let** :

```
for (let x=5; x>=1; x--) {
    console.log(x);
}
console.log(x);
```

Ketika program ini dijalankan, maka akan menampilkan output pesan error yaitu 'x is not defined', yang berarti variabel x belum dibuat.s

Namun jika pendeklarasiannya menggunakan kata kunci **var**, seperti contoh berikut :

```
for (var x=5; i>1; i--) {
    console.log(x);
}
console.log(x);
```

Ketika program dijalankan, maka akan menampilkan output nilai 5 sampai 2, dan output berikutnya berupa nilai 1. Dari kedua contoh program diatas dapat disimpulkan bahwa deklarasi variabel dalam *block scope* dan mengaksesnya di luar blok, dengan kata kunci **var** tidak menyebabkan kesalahan atau error. Namun jika menggunakan kata kunci **let** akan menyebabkan kesalahan atau error, karena **let** menganut sistem *block scope.*

Sekarang perhatikan contoh pendeklarasian varibel dalam fungsi berikut :

```
function coba() {
    var x=100;
    return x;
}
console.log(x);
```

Ketika program tersebut dijalankan, maka akan menghasilkan output pesan error 'x is not defined'. Mengapa? Karena **var** menganut sistem *function scope*, dimana variabel tidak bisa diakses di luar fungsi. Tetapi masih bisa diakses di luar blok lain, seperti perulangan maupun percabangan. **let** dan **const** tidak bisa diakses di luar fungsi ataupun di luar blok.

2. Ketatapan data

Perbedaaan pendeklarasian variabel dengan kata kunci **var**, **let**, dan **const** juga dapat dilihat dari ketetapan datanya. Apakah data yang didefinisikan dapat diubah nilainya atau tidak. Perhatikan contoh deklarasi berikut :

```
let x=100;
x=200;
console.log(x);
```

Ketika dijalankan, program tersebut akan menghasilkan output : 200. Ini berarti bahwa dengan menggunakan **let** data bisa diubah.

Sekarang perhatikan contoh penggunaan **var** berikut :

```
var x=100;
x=200;
console.log(x);
```

Ketika dijalankan, program tersebut akan menghasilkan output : 200. Ini berarti bahwa dengan menggunakan **var** data bisa diubah.

Bagaimana jika menggunakan **const** ?

```
csonst x=100;
x=200;
console.log(x);
```

Ketika program dijalankan, akan menghasilkan output : pesan error 'Assignment to constant variable'.

Ini berarti bahwa dengan menggunakan **const** data tidak bisa diubah-ubah.

3. Hoisting

Hoisting merupakan sebuah mekanisme atau konsep dalam JavaScript, dimana semua deklarasi variabel dan fungsi akan dinaikkan ke bagian atas kode program ketika program dieksekusi. Namun dalam hal ini, **JavaScript hanya memindahkan deklarasinya saja dan bukan inisiasinya**. Dengan konsep hoisting ini, variabel maupun fungsi akan dapat digunakan sebelum mendeklarasikannya. Sebelum dieksekusi, JavaScript akan mencari bagian deklarasi variabel maupun fungsi terlebih dahulu, setelah itu menginisialisasinya dengan nilai undifined.

Hoisting pada **var**, dapat dilihat pada contoh kode program berikut :

```
console.log(x);
var x=250;
```

Hoisting pada contoh diatas dapat diartikan seperti ini :

```
var x;
console.log(x); //x is undifined
x=250;
```

Jadi deklarasi variabel dengan **var** akan dinaikkan ke atas scope-nya dan variabel diinisialisasi dengan nilai undifined sebelum dieksekusi/diakses. Program diatas akan menghasilkan output : 250.

Bagimana jika hoisting pada **let** dan **const**? Perhatikan contoh berikut ini :

```
console.log(x);
let x=250;
// atau const x=250;
```

Ketika dijalankan, program tersebut akan menghasilkan output pesan error 'Cannot access 'x' before initialization'.

Seperti halnya **var,** variabel yang dideklarasikan dengan **let** atau **const** juga dinaikkan ke atas *scope*-nya, tetapi **TANPA INISIALISASI** default/undifined. Jadi, jika menggunakan **let** dan **const**, variabel tidak dapat diakses sebelum inisialisasi.

DAFTAR PUSTAKA

Flanagan, David . 2011. *JavaScript: The Definitive Guide, Sixth Edition*. United States of America : O'Reilly Media, Inc

Ogden, Maxwell. 2015. *JavaScript For Cats : An Introduction for New Programmers*.

Ullman, Larry. 2012. Modern JavaScript : Develop and Design. United States of America : Peachpit Press.

https://www.w3schools.com/js/js_reserved.asp

BAB 3
KONVERSI TYPE DATA DENGAN JS

Oleh Wahyuddin S

3.1 Konversi Tipe Data JavaScript

Konversi tipe (*type conversion/typecasting*) merupakan operasi mengonversikan data dari satu tipe data ke tipe data lainnya seperti, dari *number* ke *string*, *number* ke *Boolean*, dan lain sebagainya. Proses konversi dapat dilakukan secara otomatis (*implisit*) maupun secara manual (*eksplisit*) (Id.javascript.info, 2022; kodealgo.com, 2022).

3.1.1Konversi *String* ke *Number*

Cara untuk konversi *string* ke *number* pada JavaScript dengan menggunakan fungsi konstruktor bawaan, seperti pada gambar berikut:

```
const str = "15"
console.log(str); // "15"
console.log(typeof str); // "string"

const number = Number(str);
console.log(number); // 15
console.log(typeof number); // "number"
```

Gambar 3.1 : Konversi *String* ke *Number*
Sumber: https://www.sitepoint.com/convert-string-to-number-javascript/

Pada gambar 3.1 dapat dilihat pada saat mencatat nilai *string* nilai 15 sebagai *string*, setelah dilakukan konversi maka hasilnya nilai 15 sebagai *number* (sitepoint.com, 2022).

3.1.2 Konversi String ke Number Menggunakan parseInt() dan parseFloat()

Pendekatan lain adalah menggunakan *parseInt()* dan *parseFloat*. *parseInt()* mengurai *string* ke bilangan bulat, sedangkan *parseFloat()* mengurai *string* ke angka dengan titik desimal.

```
const str = "15.1"
console.log(str); // "15.1"
console.log(typeof str); // "string"

const intNumber = parseInt(str);
console.log(intNumber); // 15
console.log(typeof intNumber); // "number"

const floatNumber = parseFloat(str);
console.log(floatNumber); // 15.1
console.log(typeof floatNumber); // "number"
```

Gambar 3.2 : Konversi *String* ke *Number* Menggunakan *Parseint()* dan *parseFloat()*
Sumber: https://www.sitepoint.com/convert-string-to-number-javascript/

Seperti pada pendekatan sebelumnya, ketika mencatat nilai *string* dan jenis konsol maka, hasilnya adalah 15.1 sebagai *string*. Ketika mengurai *string* menggunakan *parseInt*, maka nilai *intNumber* menjadi 15 dalam bentuk *number.* Di sisi lain, ketika mengurai *string* menggunakan *parseFloat*, nilai *floatNumber* menjadi 15.1 sebagai angka atau *number.*

3.1.3 Konversi *String* ke *Number* Menggunakan *Unary Plus*

Cara ketiga untuk konversi *string* menjadi *number* adalah dengan menggunakan *unary plus* (+). Jika *unary plus* digunakan sebelum *operand* (entitas tempat operator

bertindak) maka diubah menjadi *number*. Sehingga, jika *operand* adalah *string* yang berisi *number*, *unary plus* akan mengubahnya menjadi *number.*

```
const str = "15";
console.log(typeof str); // "string"
console.log(+str); // 15
console.log(typeof +str); // "number"
```

Gambar 3.3 : Konversi *String* ke *Number* Menggunakan *Unary Plus*
Sumber: https://www.sitepoint.com/convert-string-to-number-javascript/

Berikut ini, beberapa metode yang dapat digunakan untuk mengkonversikan *Number* untuk *String*

Tabel 3.1 : Konversi Bilangan Untuk String

Metode	**Deskripsi**
toExponential()	Mengembalikan *string,* dengan sejumlah bilangan bulat dan ditulis menggunakan notasi eksponensial.
toFixed()	Mengembalikan *string,* dengan sejumlah bilangan bulat dan ditulis dengan angka desimal yang ditentukan.
toPrecison()	Mengembalikan *string,* ditulis dengan panjang tertentu.

3.1.4 Konversi *Date* ke *String*

Metode *global String()* dapat mengonversikan type data *date* menjadi *string.* Objek *date* pada JavaScript menyimpan tanggal, waktu serta menyediakan metode untuk manajemen tanggal atau waktu. Berikut ini merupakan contoh penerapan tanggal dan waktu pada JavaScript dengan menggunakan *Date*

Object (DosenIT.com, 2022). Berikut ini contoh penerapan tanggal dan waktu pada JavaScript dengan menggunakan *Date Object:*

```
String(Date()) //Metode global String()
Date().toString() //Metode date ke String

let tgl = new Date() // Menggunakan date object JavaScript
console.log(tgl.toString())
//Output
"Wed Dec 28 2022 15:10:07 GTM+0530 (IST)"
```

Gambar 3.4 : Konversi *Date* ke *String*
Sumber: https://codekey.id/javascript/date-format-javascript/

3.1.5 *Date Format JavaScript*

Waktu tanggal yang ditampilkan pada gambar 3.4 adalah *format default*. Berikut ini beberapa cara yang dapat digunakan untuk mengatur format tanggal pada JavaScript seperti:

1. *Set Date Format* JavaScript dd-mmm-yyyy

   Langkah yang diperlukan untuk *set date format* JavaScript dalam bentuk dd-mmm-yyyy adalah dengan mengekstrak tanggal, bulan dan tahun dari *object date.*

```
let tgl = new Date ()
let format_tgl = tgl.getDate() + "-" + (tgl.getMonth() + 1) + "-"
+ tgl.getFullYear()
console.log(format_tgl)

// output
"28-DEC-2022"
```

Gambar 3.5 : Konversi *Set Date Format* dd-mmm-yyyy
Sumber: https://codekey.id/javascript/date-format-javascript/

2. *Set Date Format* JavaScript yyyy-mm-dd

Langkah yang diperlukan untu *set date format* JavaScript dalam bentuk yyyy-mm-dd adalah mengekstrak tahun, bulan dan tanggal dari objek namun tidak memerlukan *array* bulan.

```
let tgl = new Date ()
let format_tgl = tgl.getFullYear() + "-" + (tgl.getMonth() + 1)
+ "-" + tgl.getDate()
console.log(formatted_date)

// output
"2022-12-28"
```

Gambar 3.6 : Konversi *Set Date Format* yyyy-mm-dd
Sumber: https://codekey.id/javascript/date-format-javascript/

3. *Set Date Format* JavaScript yyyy-mm-dd hh:mm:ss

Langkah yang diperlukan untuk *set date format* JavaScript dalam bentuk yyyy-mm-dd mirip dengan cara sebelumnya, dengan metode ini tidak perlu lagi menggunakan plugin JQuery lain untuk mengubah format tanggal waktu pada JavaScript. Berikut *source code* dapat dilihat pada gambar 3.7 (codekey.id, 2022a).

```
hh:mm:ss.
let tgl = new Date()
let format_tgl = tgl.getFullYear() + "-" + (tgl.getMonth() + 1)
+ "-" + tgl.getDate() + " " + tgl.getHours() + ":" + tgl.getMinutes()
+ ":" + tgl.getSeconds()
console.log(format_tgl)

// output
"2022-12-28 09:12:23"
```

Gambar 3.7 : Konversi *Set Date Format* yyyy-mm-dd
Sumber: https://codekey.id/javascript/date-format-javascript/

Beberapa *method Date* yang mengkonversikan ke *String* dapat dilihat pada table berikut (codekey.id, 2022b):

Tabel 3.2 : *Method* Date Konversi ke *String*

getDate()	Mengambil hari sebagai angka (1-31)
getDay	Mengambil hari kerja sebagai nomor (0-6)
getFullYear()	Mengambil tahun empat digit (yyyy)
getHours()	Mengambil jam (0-23)
getMilliseconds()	Mengambil milidetik (0-999)
getMinutes()	Mengambil menit (0-59)
getMonth()	Mengambil bulan (0-11)
getSeconds()	Mengambil detik (0-59)
getTime()	Mengambil waktu (milidetik sejak 1 januari 1970 UTC "Universal Time Coordinat")

3.1.6 Konversi *Boolean* ke *Number*

Untuk melakukan konversi *Boolean* Ke *Number* pada JavaScript dapat dilakukan dengan menggunakan metode Number():

```
Number (false)        // nilai akan menjadi 0
Number (true)         // nilai akan menjadi 1
```

Gambar 3.8 : Konversi *Boolean* Ke *Number*
Sumber: https://codekey.id/javascript/javascript-type-conversion/

3.1.7 Konversi *Boolean* ke *String*

Untuk melakukan konversi *Boolean* ke *String* pada JavaScript dapat dilakukan dengan menggunakan metode String():

```
String (false)      // nilai akan menjadi "false"
String (true)       // nilai akan menjadi "true"
```

Gambar 3.9 : Konversi *Boolean* Ke *String*
Sumber: https://codekey.id/javascript/javascript-type-conversion/

3.2 Tipe Data JavaScript

Tipe data dalam java script dibedakan menjadi tipe data primitif dan non-primitif (w3bai.com, 2022).

3.2.1 Tipe data primitive

- Boolean
- Null
- Undefined
- Number
- BigInt
- String
- Symbol

3.2.2 Tipe data non-primitif

- Object
- Array

3.2.3 Tipe data yang tidak dapat menyimpan nilai

- Undefined Type
- Null Type

3.3 Manipulasi String JavaScript

Berikut ini, beberapa *function javascript* untuk memanipulasi *string* dengan penjelasan serta fungsinya dapat di lihat pada tabel berikut (malasngoding.com, 2022):

Tabel 3.3 : Jenis Konversi JavaScript

Nama Fungsi	**Kegunaan**
length	Menghitung jumlah karakter pada *string*
indexOf()	Menemukan letak string dalam sebuah *string* (di urutan ke berapa)
lastindexOf()	Menemukan letak string dalam sebuah *string* (di urutan ke berapa) di hitung dari akhir
search()	Mencari *string* di dalam string
slice	Menampilkan *string* dari karakter ke-berapa sampai yang ke berapa
substring()	Menampilkan *string* dari karakter ke berapa sampai yang ke berapa (sama seperti slice())
substr()	Menampilkan *string* dari karakter ke berapa sampai yang ke berapa (sama seperti slice())
replace()	Mengganti *string*
toUpperCase()	Mengubah *string* menjadi huruf besar semua
toLowerCase()	Mengubah *string* menjadi huruf kecil semua
concat()	Menggabungkan *string*
split()	Mengubah *string* menjadi *array*

3.4 Jenis Konversi JavaScript

Pada tabel 3.2 menunjukkan hasil dari konversi nilai pada JavaScript yang berbeda untuk *Number, String* dan *Boolean* dapat dilihat pada tabel berikut:

Tabel 3.4 : Jenis Konversi JavaScript

Asli Nilai	**Dikonversi Nomor**	**Dikonversi String**	**Dikonversi Boolean**
false	0	”false”	false
true	1	”true”	true
0	0	”0”	false
1	1	”1”	true
”0”	0	”0”	true
”1”	1	”1”	true
NaN	NaN	”NaN”	false
Infinity	Infinity	”Infinity”	true
-Infinity	-Infinity	”-Infinity”	true
””	0	””	false
”20”	20	”20”	true
”twenty”	NaN	”twenty”	true
[ ]	0	””	true
[20]	20	”20”	true
[10,20]	NaN	”10,20”	true
[”twenty”]	NaN	”twenty”	true
[”ten”,”twenty”]	NaN	”ten, twenty”	true
function(){}	NaN	”function(){}”	true
{ }	NaN	”[object Object]”	true
null	0	”null”	false
underfined	NaN	”underfined”	false

3.5 Operator Typeof

Typeof operator digunakan ketika ingin mengetahui tipe data dalam sebuah variabel javaScript. Hal tersebut penting sebelum melakukan *type conversion* javaScript seperti:

1. Type data NaN adalah angka
2. Type data *array* adalah objek
3. Type data pada tanggal adalah objek
4. Type data null adalah objek
5. Type data dari variabel tidak terdefinisi undefined
6. Type data variabel yang tidak memiliki nilai undefined.

Berikut contoh penggunaan operator typeof untuk menemukan tipe data pada variabel JavaScript dapat dilihat pada gambar di bawah:

```
typeof "John"                 // Return "string"
typeof 3.14                   // Return "number"
typeof NaN                    // Return "number"
typeof false                  // Return "boolean"
typeof [1,2,3,4]              // Return "object"
typeof {name:'John', age: 34} // Return "object"
typeof new Date()             // Return "object"
typeof function () {}         // Return "function"
typeof myCar                  // Return "undefined"
typeof null                   // Return "object"
```

Gambar 3.10 : Operator Typeof
Sumber: https://www.w3schools.com/js/js_typeof.asp

DAFTAR PUSTAKA

codekey.id. 2022a. Tutorial Javascript 14 : Javascript Type Conversion, Bagaimana Cara Mengkonversi Tipe Data pada Javascript. Retrieved December 14, 2022, from https://codekey.id/javascript/javascript-type-conversion/

codekey.id. 2022b. Tutorial JavaScript 24 : Memahami Fungsi Date Format JavaScript. Retrieved December 20, 2022, from https://codekey.id/javascript/date-format-javascript/

DosenIT.com. 2022. Konversi di JavaScript: Cara - Tipe Data dan Jenisnya. Retrieved December 12, 2022, from https://dosenit.com/javascript/konversi-di-javascript#Tipe_Data_JavaScript

Id.javascript.info. 2022. Konversi Tipe. Retrieved December 10, 2022, from https://id.javascript.info/type-conversions# konversi-boolean

kodealgo.com. 2022. Konversi Tipe Data JavaScript. Retrieved December 12, 2022, from https://kodealgo.com/javascript/ konversi-tipe/

malasngoding.com. 2022. Belajar Javascript Part 12 : Manipulasi String Pada Javascript. Retrieved December 18, 2022, from https://www.malasngoding.com/manipulasi-string-pada-javascript/

sitepoint.com. 2022. Quick Tip: How to Convert a String to a Number in JavaScript - SitePoint. Retrieved December 31, 2022, from https://www.sitepoint.com/convert-string-to-number-javascript/

w3bai.com. 2022. JavaScript Jenis Konversi. Retrieved December 18, 2022, from https://www.w3bai.com/id/js/js_type_ conversion. html

BAB 4
PENCABANGAN IF

Oleh Diana

4.1 Struktur Pencabangan

Struktur pencabangan adalah instruksi yang memungkinkan instruksi untuk dieksekusi ketika apakah syarat tersebut terpenuhi atau tidak. Jika kondisi terpenuhi, perintah dijalankan, jika kondisi tidak terpenuhi, perintah lain atau struktur program lain dijalankan, yang menjalankan proses pengujian untuk membuat keputusan atau tindakan apakah akan mengeksekusi baris perintah atau memblokir instruksi.

Struktur pencabangan disebut juga struktur pilihan karena memungkinkan arah program dipilih keputusan atau tindakan apa yang dipenuhi, atau struktur bersyarat, yang disebut struktur bersyarat karena arah program ditentukan oleh kondisi yang terpenuhi. Perintah ini digunakan untuk mengambil keputusan. Dalam percabangan atau struktur bersyarat, setiap baris pernyataan tidak dieksekusi jika memenuhi kondisi.

Dalam pemrograman, pencabangan digunakan untuk menentukan langkah kerja instruksi, operator kondisional atau ekspresi logika digunakan selama percabangan. Ekspresi logis adalah cabang yang hanya bernilai BENAR (*True*) atau SALAH (*False*). Pernyataan-pernyataan tersebut membutuhkan suatu kondisi sebagai dasar pengambilan keputusan. Istilah umum digunakan dalam bentuk istilah benar dan salah.

JavaScript mendukung beberapa bentuk pencabangan yaitu:

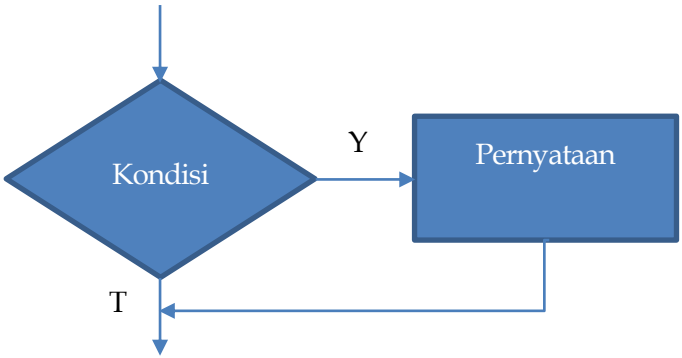
- If (*pernyantaan*)
- If (*pernyantaan*) else (*pernyantaan*)

- If else if (*pernyantaan*)

4.2 Pencabangan IF

Pencabangan IF merupakan pencabangan yang hanya memiliki satu blok pilihan saat kondisi berniali benar.
Bentuk dari pencabangan IF :

- Jika kondisi yang diseleksi adalah benar (bernilai logika = 1), maka pernyataan yang mengikutinya akan diproses.
- Sebaliknya, jika kondisi yang diseleksi adalah tidak benar (bernilai logika = 0), maka pernyataan yang mengikutinya tidak akan diproses.



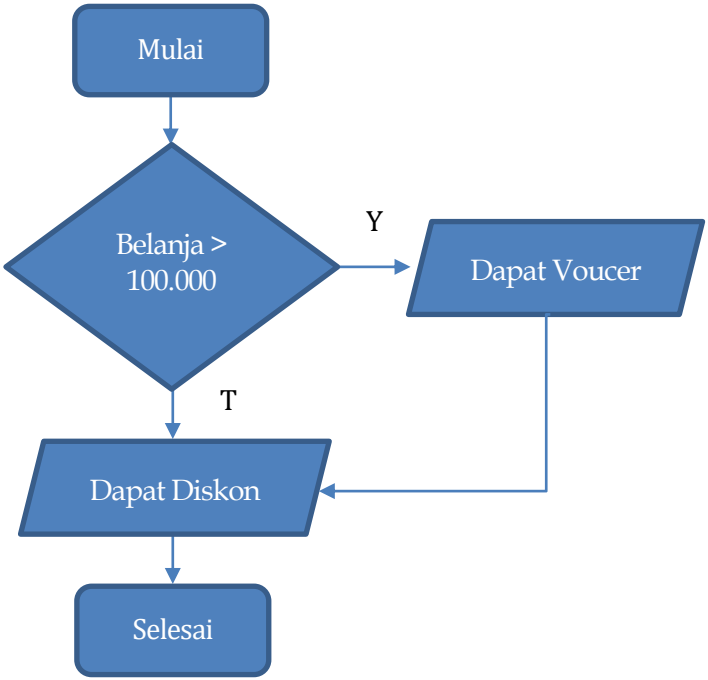
Gambar 4.1 : Struktur Pencabangan IF

Adapun sintak pencabangan IF adalah :

if (kondisi) {
Code yang akan dieksekusi jika memenuhi kondisi
}

Kondisi adalah ekspresi JavaScript yang hasilnya adalah nilai Boolean True atau False.

Contoh Flowchart menggunakan pencabangan IF :



Gambar 4.2 : Contoh Flowchart Menggunakan Pencabangan IF

Kode Javascript pencabangan IF :

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Percabangan if</title>
</head>
<body>
<script>
var totalBelanja = prompt("Belanja?", 0);
if(Belanja > 100000){
document.write("<h2>Dapat Voucer</h2>");
}
```

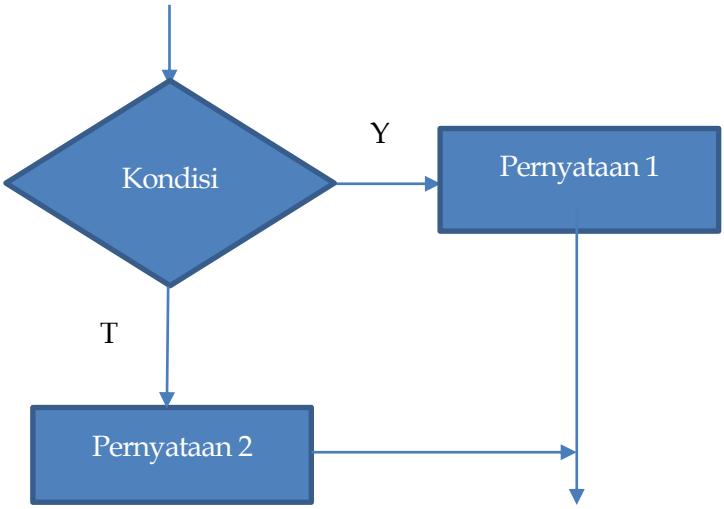
```
document.write("<p>Dapat Diskon</p>");
</script>
</body>
</html>
```

4.3 Pencabangan IF/ELSE

Pencabangan IF/ELSE merupakan pencabangan yang memiliki dua blok pilihan. Pilihan pertama untuk kondisi benar *(True*), dan pilihan kedua untuk kondisi salah (*False*)

Bentuk dari pencabangan IF /ELSE :

- Jika kondisi yang diseleksi adalah benar (bernilai logika = 1), maka pernyataan yang mengikutinya akan diproses.
- Sebaliknya, jika kondisi yang diseleksi adalah tidak benar (bernilai logika = 0), maka pernyataan selanjutnya yang akan diproses.

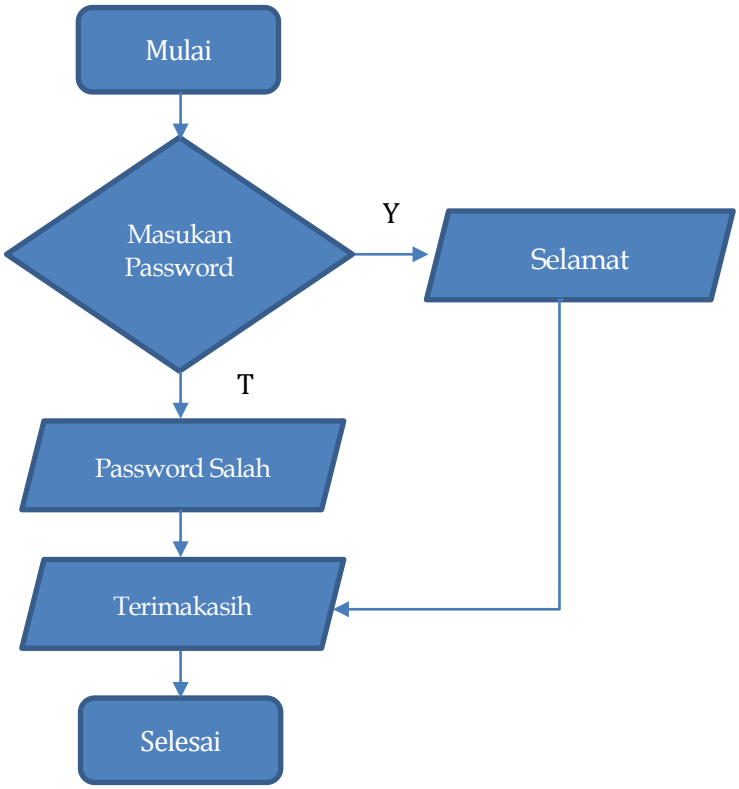


Gambar 4.3 : Struktur Pencabangan IF/ELSE

Adapun sintak pencabangan IF/ELSE adalah :

if (kondisi) {
Code yang akan dieksekusi jika memenuhi kondisi
} else {
Code yang akan dieksekusi jika kondisi tidak terpenuhi
}

Contoh Flowchart menggunakan IF/ELSE :



Gambar 4.4 : Contoh Flowchart Menggunakan Pencabangan IF/ELSE

Kode Javascript pencabangan IF/ELSE :

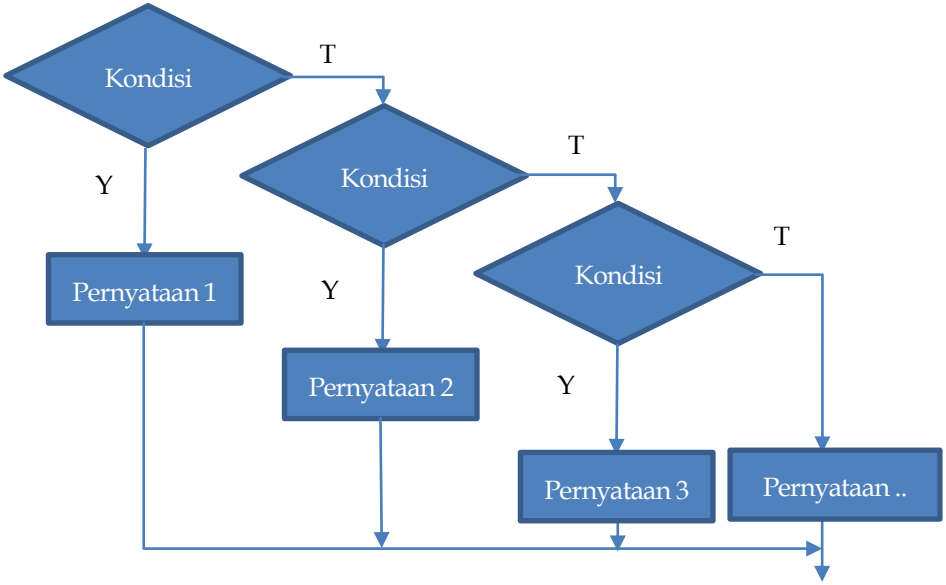
```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Percabangan if/else</title>
</head>
<body>
<script>
var password = prompt("Masukan Password:");
if(password == "kopi"){
document.write("<h2>Selamat</h2>");
} else {
document.write("<p>Password salah</p>");
}
document.write("<p>Terima kasih </p>");
</script>
</body>
</html>
```

4.4 Pencabangan IF/ELSE/IF

Pencabangan IF/ELSE/IF merupakan pencabangan yang memiliki lebih dari dua blok pilihan atau disebut pencabangan bertingkat.

Bentuk dari pencabangan IF /ELSE/IF :

- Jika kondisi yang diseleksi adalah benar (bernilai logika = 1), maka pernyataan yang mengikutinya akan diproses.
- Sebaliknya, jika kondisi yang diseleksi adalah tidak benar (bernilai logika = 0), maka pernyataan selanjutnya yang akan diproses.

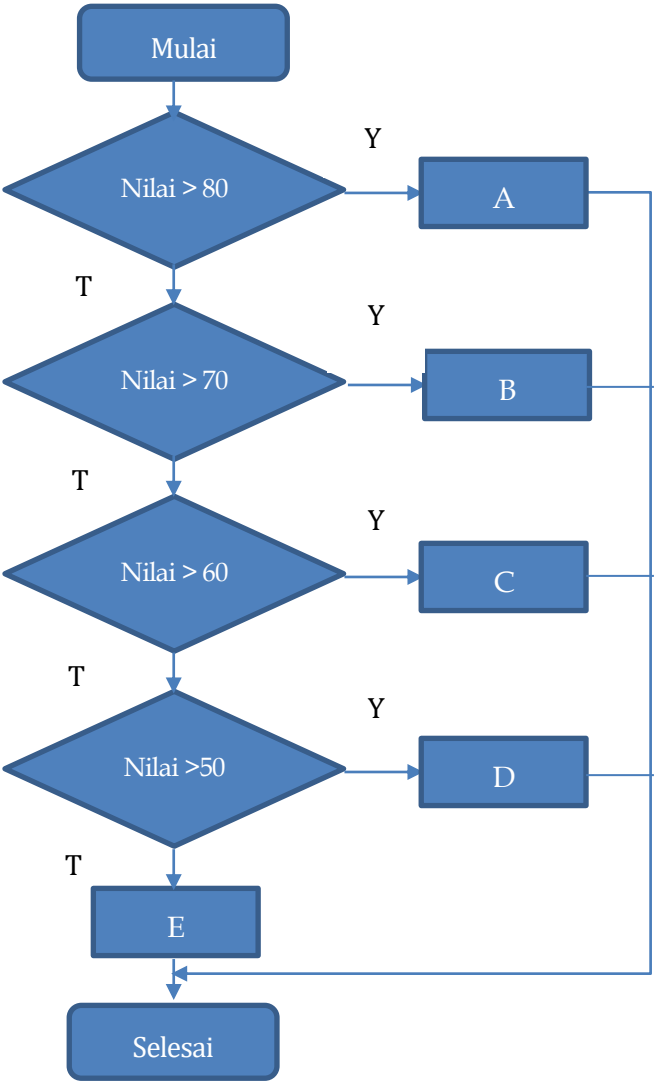


Gambar 4.5 : Struktur Pencabangan IF/ELSE/IF

Adapun sintak pencabangan IF/ELSE/IF adalah :

if (kondisi 1) {
Code yang akan dieksekusi jika memenuhi kondisi 1
} else if (kondisi 2) {
Code yang akan dieksekusi jika memenuhi kondisi 2
} else if (kondisi 3) {
Code yang akan dieksekusi jika memenuhi kondisi 3
} else {

Contoh Flowchart menggunakan IF/ELSE/IF :



Gambar 4.6 : Contoh Flowchart Menggunakan Pencabangan IF/ELSE/IF

Kode Javascript pencabangan IF/ELSE/IF :

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Percabangan if/else/if</title>
</head>
<body>
<script>
var nilai = prompt("Inputkan nilai akhir:");
var grade = "";

if(nilai >= 90) grade = "A"
else if(nilai >= 80) grade = "B+"
else if(nilai >= 70) grade = "B"
else if(nilai >= 60) grade = "C+"
else if(nilai >= 50) grade = "C"
else if(nilai >= 40) grade = "D"
else if(nilai >= 30) grade = "E"
else grade = "F";

document.write(`<p>Grade anda: ${grade}</p>`);
</script>
</body>
</html>
// blok kode
}
```

DAFTAR PUSTAKA

Enterprise, J. 2017. Otodidak Pemrograman JavaScript. Jakarta: Elex MediaKomputindo.

Ringo Anggiat Marubah. 2012. Javascript Programing Medan-Ind.University Work.

RH. Sianipar. 2017. Dasar Pemrograman JavaScript. Penerbit Andi Yogyakarta.

Jubilee Enterprise. 2021. JavaScript Untuk Pemula. Elex Media Komputindo. Jakarta.

BAB 5
OPERATOR DENGAN JS

Oleh Andy Rachman

5.1 Pendahuluan

Operator dalam bahasa pemrograman merupakan sebuah karakter yang merepresentasikan tindakan atau logika proses, sebagai contoh karakter "+" mengidikasikan proses penjumlahan, karakter "-" mengindikasikan proses pengurangan, karakter "x" mengindikasikan proses perkalian, karakter "/" mengindikasikan proses pembagian, serta karakter "&&" mengindikasikan operator logika untuk AND (Awati, 2022). Operator juga memberikan informasi kepada *compiler, editor,* ataupun *interpreter* untuk melakukan unjuk kerja matematika, operasi relasional, ataupun operasi logika dan menghasilkan sebuah luaran (output/hasil akhir) (Tutorials Point, 2022). Menurut techopedia, operator pada bidang pemrograman adalah simbol yang mewakili sebuah tindakan atau proses. Simbol-simbol tersebut diadaptasi dari ilmu matematika dan logika. Operator mampu memanipulasi nilai atau operan tertentu (techopedia, 2022).

JavaScript disingkat JS, merupakan bahasa skrip yang saat ini banyak dimanfaatkan oleh *programmer* terutama untuk mengembangkan aplikasi berbasis web. JavaScript tidak sama dengan Java dimana JavaScript sangat ringan dijalankan pada setiap komputer dan tidak perlu melakukan instalasi aplikasi layaknya bahasa Pemrograman Java. JavaScript mirip dengan VBScript dimana untuk mengetahui hasilnya harus dijalankan pada tempat lain. Artinya JavaScript merupakan bahasa pemrograman yang berjenis *interpreter*, dimana JavaScript tidak perlu memerlukan *compiler* untuk menjalankannya (Dicoding, 2020).

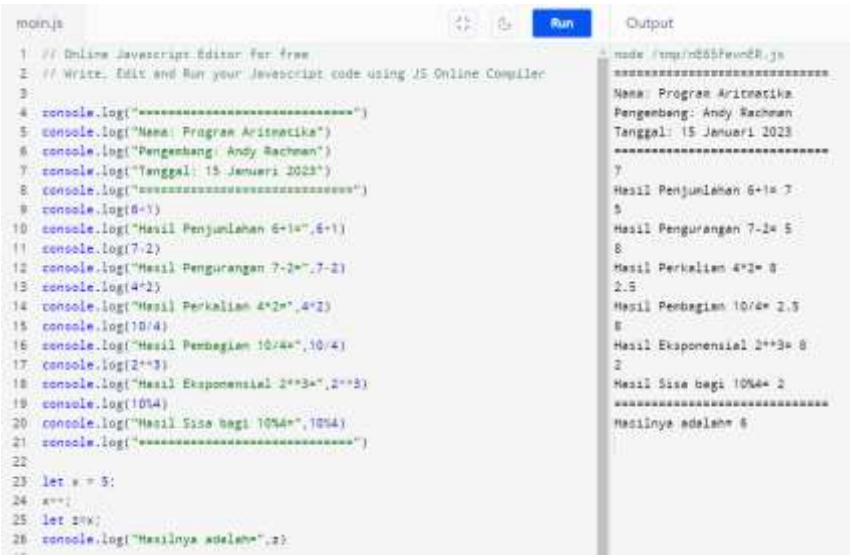
Didalam JavaScript terdapat enam jenis operator, yaitu operator aritmatika, operator penugasan, operator perbandingan, operator logika, operator logika, dan operator bit.

5.2 Operator Aritmatika

Operator Aritmatika terdiri dari delapan jenis operator, yaitu penjumlahan, pengurangan, perkalian, pembagian, eksponensial, sisa bagi, peningkatan, dan penurunan. Lebih detail dapat dilihat pada tabel 5.1.

Tabel 5.5 : Operator Aritmatikan Pada JavaScript

OPERATOR	KETERANGAN
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
**	Eksponensial
%	Sisa Bagi (Modulus)
++	Peningkatan (*Increment*)
--	Penurunan (*Decrement*)



Gambar 5.1 : Program Aritmatika Dua Buah Bilangan

Penjelasan <u>Program Aritmatika</u>.

Baris 1-2	Informasi tentang editor JavaScript yang digunakan. (tidak ditampilkan pada layar)
Baris 4-8	Informasi Program yang dicetak dilayar monitor saat program dieksekusi
Baris 9	Menampilkan hasil penjumlahan dari dua buah bilangan (tanpa informasi)
Baris 10	Menampilkan informasi **Hasil Penjumlahan 6+1=** dan disertai dengan hasil penjumlahan operand 6 dan operand 1 dalam hal ini adalah hasilnya adalah 7
Baris 11	Menampilkan hasil pengurangan dari dua buah bilangan (tanpa informasi)
Baris 12	Menampilkan informasi **Hasil Pengurangan 7-2=** dan disertai dengan hasil pengurangan operand 7 dan operand 2 dalam hal ini adalah

	hasilnya adalah 5.
Baris 13	Menampilkan hasil perkalian dari dua buah bilangan (tanpa informasi)
Baris 14	Menampilkan informasi **Hasil Perkalian 4*2=** dan disertai dengan hasil perkalian operand 4 dan operand 2 dalam hal ini adalah hasilnya adalah 8
Baris 15	Menampilkan hasil pembagian dari dua buah bilangan (tanpa informasi)
Baris 16	Menampilkan informasi **Hasil Pembagian 10/4=** dan disertai dengan hasil perkalian operand 10 dan operand 4 dalam hal ini adalah hasilnya adalah 2.5
Baris 17	Menampilkan hasil eksponensial dari dua buah bilangan (tanpa informasi)
Baris 18	Menampilkan informasi **Hasil Eksponensial 2**3=** dan disertai dengan hasil perkalian operand 2 dan operand 3 dalam hal ini adalah hasilnya adalah 8
Baris 19	Menampilkan hasil sisa hasil bagi (modulus) dari dua buah bilangan (tanpa informasi)
Baris 20	Menampilkan informasi **Hasil sisa bagi 10%4=** dan disertai dengan hasil perkalian operand 10 dan operand 4 dalam hal ini adalah hasilnya adalah 2.5
Baris 23-26	Pada baris 23, diberikan x dengan nilai 5. Baris 24 meningkatkan nilai x dengan 1 secara otomatis. Baris 25 menyimpan hasil pada z. Baris 26 menampilkan hasil increment pada layar monitor dengan info: **Hasilnya adalah=** dengan isi nilai dari z dalam hal ini adalah 6

5.3 Operator Penugasan

Operator penugasan pada JavaScript terdiri dari tujuh jenis seperti terlihat pada Tabel 2.

Tabel 5.6 : Operator Penugasan Pada JavaScript

OPERATOR	CONTOH	PENJELASAN
=	M=P	Bahwa M=P
+=	M+=P	Bahwa M=M+P
-=	M-=P	Bahwa M=M-P
*=	M*=P	Bahwa M=M*P
/=	M/=P	Bahwa M=M/p
%=	M%=P	Bahwa M=M%P
**=	M**=P	Bahwa M=M**P

main.js

Run

```
3
4  console.log("============================")
5  console.log("Nama: Program OPERATOR PENUGASAN")
6  console.log("Pengembang: Andy Rachman")
7  console.log("Tanggal: 15 Januari 2023")
8  console.log("============================")
9
10 let nilai=50
11 console.log("Nilai Awal=",nilai)
12 console.log("--------------------")
13 nilai+=10
14 console.log("1. Nilai Sekarang=",nilai)
15 nilai-=15
16 console.log("2. Nilai Saat ini=",nilai)
17 nilai*=2
18 console.log("3. Nilai Setelah Update=",nilai)
19 nilai/=3
20 console.log("4. Nilai Sekarang menjadi=",nilai)
21 nilai**=2
22 console.log("5. Nilai Saat ini  menjadi=",nilai)
23 nilai%=7
24 console.log("6. Nilai Sekarang  menjadi=",nilai)
```

Gambar 5.3 : Program Operator Penugasan Pada JavaScript

Gambar 5.4 : Hasil Program Operator Penugasan Pada JavaScript

5.4 Operator Perbandingan

Operator perbandingan pada JavaScript memiliki sembilan buah operator, yaitu

Tabel 5.7 : Operator Perbandingan Pada JavaScript

OPERATOR	PENJELASAN
==	Sama dengan
===	Nilai yang sama dan tipe yang sama
!=	Tidak sama
!==	Nilai tidak sama atau tipe tidak sama
>	Lebih besar
<	Lebih kecil
>=	Lebih Besar dari atau sama dengan
<=	Lebih kecil dari atau sama dengan
?	Operator Tenary

5.5 Operator Logika

Operator logika pada JavaScript terdiri dari tiga jenis operator yaitu logika **and**, logika **or**, dan logika **not**. Hal ini dapat dilihat pada tabel 5.4 dibawah ini

Tabel 5.8 : Operator Perbandingan Pada JavaScript

OPERATOR	PENJELASAN
&&	Operator logika **and**
||	Operator logika **or**
!	Operator logika **not**

5.6 Operator bit

Operator bit pada JavaScript terdiri dari tujuh jenis operator bit, yaitu operator AND (&), operator OR (|), operator NOT (~), operator XOR (^), operator left shift (geser kiri) (<<), operator reight shift (geser kanan) (>>), dan operator unsigned right shift. Operator bit dapat dilihat pada tabel 5.5.

Tabel 5.9 : Operator Perbandingan Pada JavaScript

OPERATOR	PENJELASAN	CONTOH	HASIL
&	AND	0101 & 0001	0001
|	OR	0101 | 0001	0101
~	NOT	~0101	1010
^	XOR	0101 ^ 0001	0100
<<	Left shift	0101 << 1	1010
>>	Right shift	0101 >> 1	0010
>>>	Unsigned right shift	0101 >>> 1	0010

DAFTAR PUSTAKA

Awati, R. 2022. *Operator*. Operator. https://www.techtarget.com/whatis/definition/operator

Dicoding, D. 2020. *Apa Itu JavaScript? Fungsi dan Contohnya*. Apa Itu JavaScript? Fungsi Dan Contohnya. https://www.dicoding.com/blog/apa-itu-javascript-fungsi-dan-contohnya/

Mansyur, A. R. 2020. Dampak COVID-19 Terhadap Dinamika Pembelajaran Di Indonesia. *Education and Learning Journal*, *Vol. 1, No*, 113–123.

Santyasa, I. W. 2005. Model Pembelajaran inovatif dalam implementasi kurikulum berbasis kompetensi. *Makalah Disampaikan Dalam Penataran Guru-Guru SMP, SMA, Dan SMK Se Kabupaten JembranaJuni–Juli*.

techopedia, techopedia. 2022. *Operator*. Operator. https://www.techopedia.com/definition/3485/operator-programming

Tutorials Point, T. P. 2022. *Computer Programming—Operators*. Computer Programming - Operators. https://www.tutorialspoint.com/computer_programming/computer_programming_operators.htm

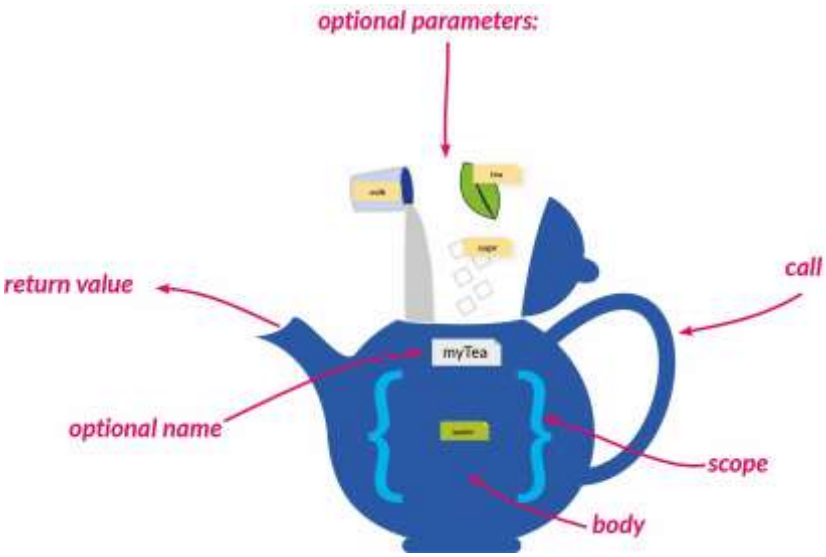
BAB 6
FUNCTION DALAM PEMROGRAMAN WEB DENGAN JS

Oleh Ronny Makhfuddin Akbar

6.1 Definisi Fungsi

Function (*Fun Action*) adalah sekelompok kode yang dapat dipanggil berulang dan melakukan tugas tertentu (Demirov, 2015). Semakin besar program Anda dan semakin kompleks perangkat lunak yang Anda kembangkan, semakin banyak *script* yang harus Anda kelola. Selain itu, penulisan *script* tertentu menjadi berulang dari waktu ke waktu. Sehingga, lebih baik mengelompokkan *script* ke dalam blok kode yang dapat digunakan kembali agar Anda dapat menggunakan kembali fungsi-fungsi tersebut daripada menulis ulang pernyataan setiap saat. Suatu fungsi dapat dipanggil di tempat yang berbeda dalam suatu program.

Sebuah fungsi dapat digunakan untuk menulis pesan kepada pengguna, menghitung total pajak atas suatu pesanan, atau memperkirakan tanggal pengiriman suatu produk. Perintah yang disimpan dalam suatu fungsi dapat dipanggil berulang kali di seluruh program Anda, menyederhanakan kode Anda dan membuatnya lebih efisien untuk dijalankan (Carey and Vodnik, 2021). Alur kerja fungsi dapat dilihat pada gambar 6.1.



Gambar 6.1 : Alur Kerja Fungsi
(Demirov, 2015)

6.1.1 Deklarasi Fungsi

Dalam *JavaScript* (JS), Anda mendefinisikan fungsi dengan kata kunci *function*. Setelah kata kunci *function*, Anda menulis nama fungsi diikuti dengan tanda kurung, dan pernyataan didalamnya. Istilah teknis di atas disebut dengan deklarasi fungsi.

Berikut contoh sederhana dari deklarasi fungsi pada Segmen Program 6.1. Ketika fungsi bernama tampilPesan() dipanggil, mencetak tulisan Selamat Datang di konsol. Namun, biasanya fungsi terdiri dari beberapa pernyataan yang melakukan tugas yang jauh lebih kompleks.

Segmen Program 6.1. Contoh Deklarasi Fungsi

```
function tampilPesan(){
console.log('Selamat Datang');
}
```

6.1.2 Definisi Fungsi dengan Ekspresi

Dalam *JavaScript*, fungsi dapat dibuat tidak hanya melalui deklarasi fungsi, tetapi juga dengan ekspresi fungsi. Dalam hal ini, Anda biasanya tidak menentukan nama fungsi, yang membuat fungsi menjadi fungsi anonim. Berikut contoh ditunjukkan pada Segmen Program 6.2. Fungsi yang ditampilkan ditugaskan ke variabel tampilPesan. Setelah itu, fungsi tersebut dapat dipanggil melalui variabel tampilPesan().

Segmen Program 6.2. Contoh Ekspresi Fungsi

```
const tampilPesan = function() {
console.log('Selamat Datang');
}
tampilPesan();
```

6.2 Parameter Fungsi

Seringkali suatu fungsi memerlukan informasi tertentu untuk dapat dieksekusi. Misalkan kita ingin memodifikasi fungsi tampilPesan() sehingga dapat digunakan untuk mengeluarkan pesan pada konsol. Untuk melakukan ini Anda membutuhkan cara untuk mengirim pesan ke dalam fungsi.

Di sinilah parameter fungsi berperan. Dengan parameter fungsi, Anda dapat meneruskan informasi ke suatu fungsi, dan kemudian tersedia di dalam fungsi tersebut. Terkait nilai yang diteruskan ke fungsi, Anda akan melihatnya disebut sebagai parameter atau sebagai argumen. Secara teknis, variabel yang tercantum dalam deklarasi fungsi adalah parameter, dan nilai yang diteruskan ke fungsi saat dipanggil adalah argumen (Lemay, Coburn and Kyrnin, 2018). Saat Anda memanggil suatu fungsi, Anda memberikannya argumen dan argumen tersebut kemudian dicocokkan dengan parameter dalam definisi fungsi. Anda dapat mengirimkan hampir semua nilai *JavaScript* sebagai argumen, seperti *string*, *boolean*, atau angka. (Freeman and Robson, 2014)

Untuk membuat parameter fungsi, cukup tulis nama parameter dalam tanda kurung setelah nama fungsi. Segmen program 7.3 menunjukkan fungsi tampilPesan() yang dimodifikasi dengan parameter pesan. Berikut contoh hanya mencetak parameter pesan ke konsol.

Segmen Program 6.3. Deklarasi Fungsi dengan Parameter

```
function tampilPesan(pesan) {
console.log(pesan);
}
tampilPesan('Budi: Helo Ron');
tampilPesan('Ronny: Helo Bud');
```

6.2.1 Fungsi dengan Banyak Parameter

Fungsi juga dapat menerima banyak parameter. Dalam hal ini, Anda cukup menambahkan parameter yang dipisahkan dengan koma dalam deklarasi fungsi. Segmen Program 7.4 menunjukkan contoh penggunaan fungsi dengan banyak parameter. Di sini, fungsi cetakInfoOrang() menerima tiga parameter: namaDepan, namaBelakang, dan umur.

Segmen Program 6.4. Deklarasi Fungsi dengan Banyak Parameter

```
function cetakInfoOrang(namaDepan,namaBelakang,umur) {
  console.log('Nama Depan: ${namaDepan}');
  console.log('Nama Belakang: ${namaBelakang}');
  console.log('Umur: ${umur}');
}
cetakInfoOrang('Budi','Ahmad',33);
cetakInfoOrang('Ronny','Akbar',27);
```

6.2.2 Memanggil Fungsi dengan Argumen Lebih Sedikit

Salah satu fitur khusus dalam menggunakan fungsi pada bahasa pemrograman *JavaScript* dibandingkan dengan *Java* adalah fungsi tersebut dapat dipanggil dengan argumen yang lebih sedikit daripada parameter yang ditentukan dalam deklarasi fungsi.

Fungsi cetakInfoOrang() yang telah dibahas sebelumnya dapat dipanggil tidak hanya dengan tiga argumen, tetapi juga dengan hanya dua argumen, dengan satu argumen, atau tanpa argumen sama sekali (lihat Segmen Program 6.5).

Segmen Program 6.5. Memanggi Fungsi dengan Argumen Sedikit

```
cetakInfoOrang('Ronny','Akbar');
/* Output
Nama Depan: Ronny
Nama Belakang: Akbar
Umur: Undefined
*/
```

Pada segmen program 6.5 menampilkan *output* umur *undefined* karena paramater umur tidak ditentukan dalam fungsi cetakInfoOrang(). Oleh karena itu, parameter yang Anda tidak berikan argumen apa pun saat memanggil fungsi sama dengan seperti variabel yang belum diinisialisasi di dalam fungsi. Dengan kata lain, mereka memiliki nilai *undefined*. Selain itu, pada Segmen Program 7.5 terdapat komentar yang menunjukkan hasil *output* program. Bagian kode ini adalah komentar untuk membantu Anda mengingat tujuan kode jika Anda memeriksa kode kembali. Anda dapat menggunakan // garis miring ganda di depan teks apa pun di *JavaScript* untuk mengomentari satu baris, serta /* sebagai pembuka dan */ sebagai penutup untuk mengomentari banyak baris. (Owen, 2022)

Di dalam fungsi, Anda dapat memeriksa apakah argumen telah diteruskan untuk masing-masing parameter dengan memeriksa apakah argumen tersebut berisi nilai *undefined*. Segmen program 6.6 menunjukkan fungsi cetakInfoOrang() yang dimodifikasi, di mana setiap parameter diperiksa apakah argumen ada nilainya.

Segmen Program 6.6. Fungsi dengan Cek Argumen

```
function cetakInfoOrang(namaDepan,namaBelakang,umur) {
  if(namaDepan !== undefined){
```

```
console.log('Nama Depan: ${namaDepan}');
}
if(namaBelakang !== undefined){
console.log('Nama Belakang: ${namaBelakang}');
if(umur !== umur){
console.log('Umur: ${umur}');
}
}
```

6.2.3 Fungsi dengan Argumen Angka

Rest parameter sangat berguna untuk fungsi yang memungkinkan sejumlah argumen variabel yang tak terbatas atau dinamis. Ini disebut fungsi *variadic*. Sebagai contoh, kita membuat fungsi *SUM* untuk menjumlahkan sejumlah angka dan menuliskan hasilnya di konsol.

Implementasi menggunakan array ditampilkan di Segmen Program 6.7. Dalam deklarasi fungsi, angka adalah singkatan dari *array* angka. Jumlah angka kemudian dihitung menggunakan loop penghitungan. Satu-satunya hal yang penting pada titik ini adalah bahwa *array* harus selalu dikirimkan ketika fungsi dipanggil.

Segmen Program 6.7. Implementasi Fungsi *Variadic* melalui *Array*

```
function sum(angka) {
let hasil = 0;
for (let i = 0; i < angka.length; i++) {
hasil += angka[i];
}
console.log(hasil);
}
sum([2,3,4,5]);                    // Output: 14
sum([2,3,4,5,6,7,8,9]);     // Output: 44
sum([2,3,4,5,6,7,8,9,10,11]); // Output: 65
```

Implementasi fungsi menggunakan *rest* parameter hampir tidak berbeda dari implementasi menggunakan *array*. Satu-satunya

perbedaan adalah dalam deklarasi fungsi, seperti yang dapat kita lihat di Segmen Program 6.8. Pernyataan dalam fungsi tetap sama dengan *rest* parameter dengan *array* dan karenanya dapat diulang dengan cara yang sama seperti dalam Segmen Program 6.7. Namun, yang berbeda adalah panggilan fungsi: memanggil argumen secara individual dibandingkan menggunakan array.

Segmen Program 6.8. Implementasi Fungsi *Variadic* melalui *Rest* Parameter

```
function sum(...angka) {
let hasil = 0;
for (let i = 0; i < angka.length; i++) {
hasil += angka[i];
}
console.log(hasil);
}
sum(2,3,4,5);              // Output: 14
sum(2,3,4,5,6,7,8,9);      // Output: 44
sum(2,3,4,5,6,7,8,9,10,11); // Output: 65
```

6.2.4 Definisi Fungsi dalam Fungsi

Berbeda dengan bahasa lain, seperti *Java*, *JavaScript* memungkinkan Anda untuk mendefinisikan fungsi dalam fungsi lain. Fungsi seperti itu kemudian hanya terlihat atau valid dalam fungsi di mana mereka didefinisikan.

Segmen Program 6.9 menunjukkan contoh. Pertama, fungsi SUM() didefinisikan untuk menambahkan dua angka. Di dalam fungsi, fungsi normalisasi() didefinisikan, yang hanya mengembalikan nilai 0 untuk argumen negatif. Namun, fungsi ini hanya dapat dipanggil dalam fungsi SUM(). Di luar fungsi ini, fungsi normalisasi() tidak terlihat, jadi jika dipanggil akan menghasilkan error.

Segmen Program 6.9. Fungsi Didefinisikan dalam Fungsi

```
function sum(x,y) {
```

```
const hasil = normalisasi(x) + normalisasi(y);
function normalisasi(i) {
if(i<0){
return 0;
}
return i
}
return hasil;
}
console.log(sum(-5,5));              // Output: 5
console.log(normalisasi(-5)); // ReferenceError: ....
```

6.3 Definisi Nilai *Return*

Mengembalikan informasi dari dalam fungsi ke kode panggilan disebut nilai *return*. Kita menentukan nilai *return* dalam fungsi menggunakan kata kunci *return* diikuti oleh nilai yang akan dikembalikan.

Sebagai contoh, kita akan kembali menggunakan fungsi SUM yang mirip dengan bagian sebelumnya. Segmen Program 6.10 menunjukkan fungsi dengan menambahkan dua parameter x dan y, menyimpan hasil dalam variabel hasil, dan mengembalikannya.

Segmen Program 6.10. Fungsi dengan Nilai *Return*

```
function sum(x,y) {
const hasil = x + y; // Menambahkan 2 paramater
return hasil;       // mengembalikan nilai hasil
}
```

Hasil dari fungsi dapat disimpan dalam variabel, misalnya, digunakan kembali dalam kode panggilan, seperti yang ditunjukkan dalam Segmen Program 6.11. Di sini, hasil dari jumlah panggilan fungsi (5, 5) disimpan dalam variabel hasilSatu, dan hasil dari jumlah fungsi (8, 8) juga disimpan dalam variabel hasilDua.

Segmen Program 6.11. Hasil Fungsi disimpan dalam Variabel

```
const hasilSatu = sum(5,5);
```

```
const hasilDua = sum(8,8);
console.log(hasilSatu); // 10
console.log(hasilDua) // 16
```

Pada prinsipnya, fungsi dapat mengembalikan nilai: nilai numerik, seperti yang baru saja ditunjukkan, tetapi juga string, nilai boolean, objek, array, dan bahkan fungsi. Dalam Segmen Program 6.12, Kita dapat melihat contoh fungsi yang mengembalikan objek.

Segmen Program 6.12. Fungsi dengan Nilai *Return* Objek

```
function buatUser(username, email, password) {
const user = {
username: username,
email: email,
password: password
}
return user;
}
const ronny = buatUser('Ronny', 'ronny@contoh.com', 'rahasia');
console.log(ronny.username) // Ronny
console.log(ronny.email)   // ronny@contoh.com
console.log(ronny.password) // rahasia
```

6.4 Definisi Nilai *Default* untuk Parameter

Bagaimana jika di dalam fungsi mempunyai lebih sedikit argumen yang diteruskan sebagai parameter. Tentu saja, tidak ada jawaban umum untuk pertanyaan ini. Sebagai gantinya, setiap fungsi harus dipertimbangkan secara individual: untuk fungsi tambahan yang mengharapkan dua parameter tetapi hanya diberikan satu argumen, Anda mungkin akan mendapatkan peringatan atau bahkan *error*.

Namun, fungsi dapat terus bekerja menggunakan nilai *default* untuk argumen yang dihilangkan. Misal fungsi buatUser() yang dibahas sebelumnya sebagai contoh. Fungsi tersebut mengharapkan tiga argumen untuk parameter *username*, *email*,

dan *password*. Namun, jika tidak ada *password* yang diberikan, Anda dapat menggunakan *password default*.

Sejak ES2015, ada notasi yang relatif sederhana untuk menggunakan parameter *default*. Seperti yang Anda lihat di parameter *password* di Segmen Program 6.13, nilai *default* parameter hanya ditulis setelah parameter, dipisahkan dengan tanda sama dengan.

Segmen Program 6.13. Nilai Default dengan ES2015

```
function buatUser (username, email, password = 'DeFaUlTPaSsWoRd') {
const user = {
username: username,
email: email,
password: password
}
return user;
}
const ronny = buatUser('Ronny', 'ronny@contoh.com', 'rahasia');
console.log(ronny.username) // Ronny
console.log(ronny.email)   // ronny@contoh.com
console.log(ronny.password) // rahasia
const toni = buatUser('Toni', 'toni@contoh.com');
console.log(toni.username) // Toni
console.log(toni.email)   // toni@contoh.com
console.log(toni.password) // DeFaUlTPaSsWoRd
```

Segmen Program.14 secara singkat menunjukkan bagaimana nilai default diimplementasikan menggunakan metode ES5. Idenya adalah untuk menimpa atau menginisialisasi ulang parameter yang sesuai di dalam fungsi tergantung pada nilai argumen yang sesuai.

Segmen Program 6.14. Nilai *Default* menggunakan Method ES5

```
function buatUser (username, email, password) {
 password = password !== undefined ? password : 'DeFaUlTPaSsWoRd';
```

```
const user = {
username: username,
email: email,
password: password
}
return user;
}
const ronny = buatUser('Ronny', 'ronny@contoh.com', 'rahasia');
console.log(ronny.username) // Ronny
console.log(ronny.email)   // ronny@contoh.com
console.log(ronny.password) // rahasia
const toni = buatUser('Toni', 'toni@contoh.com');
console.log(toni.username) // Toni
console.log(toni.email)   // toni@contoh.com
console.log(toni.password) // DeFaUlTPaSsWoRd
```

6.5 Elemen Array sebagai Parameter

Sesekali, Kita ingin menggunakan nilai dalam *array* sebagai argumen individual untuk suatu fungsi. Sebagai contoh, kita akan menggunakan fungsi buatUser() yang sudah ditampilkan di bab ini, yang menggunakan parameter *username*, *email*, dan *password* untuk membuat objek yang mewakili masing-masing pengguna. Anggaplah data ini, nama pengguna, alamat email, dan kata sandi, tersedia untuk Anda sebagai array, dan Anda ingin memanggil fungsi dengan data ini. Ada dua opsi yang akan kita bahas untuk menggunakan elemen *array* sebagai parameter.

Opsi 1 ditunjukkan pada Segmen Program 6.15. Data pengguna tersedia di *array* userData1 dan userData2. Untuk membuat objek yang sesuai dari dua *array* ini, Anda harus mengakses elemen individu dalam array melalui userData1[0], userData1[1], dan userData1[2] dan kemudian menggunakannya sebagai argumen.

Segmen Program 6.15. Nilai *Array* sebagai Argumen

```
function buatUser (username, email, password) {
const user = {
username: username,
email: email,
password: password
}
return user;
}
const userData1 = ['Ronny', 'ronny@contoh.com', 'rahasia'];
const userData2 = ['Toni', 'toni@contoh.com', 'password'];
const ronny = buatUser(userData1[0], userData1[1], userData1[2]);
const toni = buatUser(userData2[0], userData2[1], userData2[2]);
```

Opsi 2 menggunakan operator *spread*. Karena opsi 1 membuat jumlah pengetikan yang banyak, ES2015 memperkenalkan operator *spread*. Operator ini mengizinkan elemen-elemen *array* untuk disebar pada parameter-parameter suatu fungsi (sehingga dinamakan *spread*) sehingga elemen-elemen tersebut dapat digunakan secara langsung sebagai argumen.

Untuk menerapkan operator spread ke array, cukup tambahkan tiga titik ke nama array yang sesuai. Segmen Program 6.16 menunjukkan penggunaan operator yang terkait dengan contoh sebelumnya. Hasilnya sama seperti di Segmen Program 6.15, tetapi Anda menghemat beberapa pengetikan karena Anda tidak perlu mengakses indeks individual array secara manual.

Segmen Program 6.16. Nilai Array dengan Operator *Spread*

```
const userData1 = ['Ronny', 'ronny@contoh.com', 'rahasia'];
const userData2 = ['Toni', 'toni@contoh.com', 'password'];
const ronny = buatUser(...userData1);
const toni = buatUser(...userData2);
```

6.6 Fungsi Arrow

Sejak ES2015, ada notasi ringkas untuk mendefinisikan fungsi. Atau notasi alternatif untuk ekspresi fungsi. Karena sintaks yang digunakan yaitu string dengan karakter =>, fungsi ini disebut fungsi arrow. Sintaks tersebut terletak di bagian tengah dari fungsi *arrow*, yang memisahkan parameter fungsi (di sisi kiri) dari badan fungsi (di sisi kanan). Fungsi *arrow* ditambahkan ke JavaScript karena dua alasan: (Rauschmayer, 2022)

1. Untuk memberikan cara yang lebih ringkas untuk membuat fungsi.
2. Fungsi *arrow* berkerja lebih baik sebagai fungsi didalam *method*.

Sintaks untuk parameter dan badan fungsi bervariasi untuk fungsi *arrow* tergantung pada berapa banyak parameter yang diharapkan fungsi dan apa yang ingin Anda ekspresikan. Metode sum(), yang sudah Anda ketahui dari bagian sebelumnya, dapat ditulis sebagai fungsi *arrow*, seperti pada contoh yang ditunjukkan pada Segmen Program 6.17.

Segmen Program 6.17. Fungsi *Arrow*

```
const sum = (x,y) => {return x + y;}
// ... ini sama dengan ...
const sum = function(x,y){
return x + y;
}
```

Fungsi *arrow* dapat disederhanakan seperti yang ditunjukkan pada Segmen Program 6.18 untuk menghilangkan tanda kurung kurawal dan kata kunci *return*.

Segmen Program 6.18. Fungsi *Arrow* tanpa *return* dan kurung kurawal

```
const sum = (x,y) => x + y;
```

Untuk fungsi dengan satu parameter, tanda kurung di sekitar parameter juga dapat dihilangkan (lihat Segmen Program 6.19).

Segmen Program 6.19. Fungsi *Arrow* dengan Satu Parameter

```
const tampilPesan = pesan => console.log(pesan);
```

Sebaliknya, untuk fungsi tanpa parameter, Anda akan menggunakan sepasang tanda kurung kosong (lihat Segmen Program 6.20).

Segmen Program 6.20. Fungsi *Arrow* tanpa Parameter

```
const cetakHello = () => console.log('Hello World');
```

Fungsi *arrow* dapat mengembalikan objek, berikut Segmen Program 6.21 sebagai contoh penggunaan fungsi *arrow* dengan mengembalikan suatu objek.

Segmen Program 6.21. Fungsi Arrow yang mengembalikan Objek

```
const buatUser = (username, email, password) =>
{
{
username: username,
email: email,
password: password
}
};
// ... ini sama dengan ...
function buatUser (username, email, password) {
const user = {
username: username,
email: email,
password: password
}
return user;
}
```

Jadi, secara ringkas, notasi yang ditunjukkan pada Tabel 6.1 diperbolehkan.

Tabel 6.1 : Perbedaan Notasi untuk Fungsi *Arrow*

Sintaks	**Arti**
( ) => { < badan fungsi > }	Fungsi tanpa parameter
x => { < badan fungsi > }	Fungsi dengan satu paramater
(x, y) => { < badan fungsi > }	Fungsi dengan banyak parameter
< fungsi parameter > => { return x*x }	Badan fungsi sebagai blok
< fungsi parameter > => x*x	Badan fungsi sebagai pernyataan tunggal

Sumber: (Ackermann, 2022)

6.7 Modifikasi *String* dengan Fungsi

Fungsi untuk mempengaruhi hasil evaluasi *string* untuk *string template* ditawarkan oleh *template* yang diberi *tag*. *String template* diawali dengan nama fungsi yang disebut *tag function* (lihat Segmen Program 6.22). Pada contoh tagFungsi dipanggil dengan dua parameter: parameter pertama mewakili bagian tetap, parameter kedua bagian variabel yaitu, nilai untuk *placeholder* dari *template* yang sesuai rangkaian.

Segmen Program 6.22. Contoh *Template Tag*

```
const nama = 'Ronny Akbar';
const umur = 33;
function tagFungsi(strings, ...replacements){
  ...
}
const pesan = tagFungsi'Nama saya ${nama}, umur saya ${umur}.';
console.log(pesan);
```

Di dalam fungsi *tag*, Anda memiliki opsi untuk mengakses nilai-nilai ini dan menggunakannya, misalnya, untuk memengaruhi nilai hasil dari fungsi dan hasil dari *string template*. Di Segmen Program 7.23, fungsi *tag* memastikan bahwa string yang berbeda dihasilkan tergantung pada variabel umur yang dikirimkan. Parameter *replacements* berisi nilai dinamis dari *string* yaitu: 'Ronny Akbar', 33, dan 82. Parameter strings berisi string yang dibagi dengan nilai dinamis: 'Nama saya ', dan ', umur saya'.

Segmen Program 6.23. Implementasi Fungsi Tag

```
function tagFungsi(strings, ...replacements){
const nama = replacements[0]; // "Ronny Akbar"
const umur = replacements[1]; // 33 atau 82
if (umur > 80) {
return '${strings[0]}${replacements[0]}.';
// --> "Nama saya Ronny Akbar"
}
return '${strings[0]}${nama}${strings[1]}${umur}.';
// --> "Nama saya Ronny Akbar, umur saya 33."
}
const nama = 'Ronny Akbar';
let umur = 33
let pesan = tagFungsi'Nama saya ${nama}, umur saya ${umur}';
console.log(pesan);
// Nama saya Ronny Akbar, umur saya 33
umur = 82;
pesan = tagFungsi'Nama saya ${nama}, umur saya ${umur}';
console.log(pesan);
// Nama saya Ronny Akbar
```

6.8 Fungsi dengan Interaksi Pengguna

Di bagian sebelumnya, Anda belajar banyak tentang fungsi. Pada titik ini, kita akan menunjukkan cara memanggil fungsi di dalam halaman *web* dengan menekan sebuah tombol. *Event listener*

yang memungkinkan Anda memanggil fungsi tertentu saat suatu *event* terjadi.

Sekarang, untuk bereaksi terhadap klik tombol dan memanggil fungsi, cukup tambahkan *event listener* untuk *event* klik tombol yang sesuai. Kode HTML untuk ini ditampilkan di Segmen Program 6.24, kode *JavaScript* di Segmen Program 6.25. Ini adalah contoh halaman web sederhana di mana *form* digunakan untuk menghitung jumlah dua angka. *Form* tersebut berisi tiga *input field* (untuk dua input angka dan hasil) serta tombol untuk memicu perhitungan.

Dalam kode *JavaScript*, metode *addEventListener*() digunakan untuk mendaftarkan *event listener* untuk *event DOMContentLoaded*. Fungsi anonim yang diteruskan di sini sebagai *event lisnener* tidak dijalankan sampai semua elemen di halaman *web* dimuat. Ini memastikan bahwa *input field* dan tombol ada saat diakses di dalam kode.

Di dalam *event listener* ini, fungsi lain didaftarkan untuk dieksekusi saat *event* klik dipicu: fungsi hitungJumlah(). Pada fungsi tersebut, mengakses dua *input field*, membaca nilainya, menjumlahkannya, dan menggunakan fungsi tampilHasil() untuk menampilkan hasil di *input field* ketiga.

Segmen Program 6.24. Kode HTML

```
<!DOCTYPE html>
<html>
<head lang=”en”>
<meta charset=”UTF-8”>
<title>Contoh<title>
<link rel=”stylesheet” href=”styles.main.css” type =”text/css”>
</head>
<body>
<div class=”container”>
<div class=”row”>
```

```
 <label for=”field1”>X</label> <input id=”field1” type=”text” value=”5”>
</div>
<div class=”row”>
<label for=”field2”>Y</label> <input id=”field2” type=”text” value=”5”>
</div>
<div class=”row”>
<label for=”result”>Hasil: </label> <input id= ”result” type=”text”>
<button id=”tombol-hitung”>Hitung jumlah </button>
</div>
</div>
<script src=”scripts/main.js”></script>
</body>
</html>
```

Segmen Program 6.25. Fungsi Sederhana

```
document.addEventListener('DOMContentLoaded', function() {
const button = document.getElementById('tombol-hitung');
button.addEventListener('click', hitungJumlah);
});
function hitungJumlah() {
const x = parseInt(document.getElementById('field1').value);
const y = parseInt(document.getElementById('field2').value);
const hasil = x + y;
tampilHasil(hasil);
}
function tampilHasil(hasil) {
const fieldHasil = document.getElementById('result');
fieldHasil.value = hasil;
}
```

DAFTAR PUSTAKA

Ackermann, P. 2022. *JavaScript: The Comprehensive Guide to Learning Professional JavaScript Programming*. 1st edn. Boston: Rheinwerk Publishing Inc.

Carey, P. and Vodnik, S. 2021. *JavaScript for Web Warriors*. 7th edn. Cengage Learning. Available at: https://books.google.co.id/books?id=Iwk1EAAAQBAJ.

Demirov, I. 2015. *Learn JavaScript Visually*. Createspace Independent Pub.

Freeman, E. and Robson, E. 2014. *Head First: JavaScript programming*. 1st edn. O'Reilly Media.

Lemay, L., Coburn, R. and Kyrnin, J. 2018. *Sams Teach Yourself HTML, CSS & JavaScript Web Publishing*, *Pearson Education, INC.*

Owen, G. 2022. *Learn to Code: Learn HTML, CSS and JavaScript and build a website, an app and a game*. Independently published.

Rauschmayer, A. 2022. *JavaScript for Impatient Programmers*. Independently published.

BAB 7
OPERASI ARITMATIKA DENGAN PHP

Oleh Eferoni Ndruru

7.1 Pendahuluan

Pada bab sebelumnya kita telah mempelajari penulisan bahasa pemrograman php diantaranya adalah komentar, variabel, konstanta, data integer, data float, data string, data array data boolean dan cara penulisan semuanya kedalam bahasa pemrograman php. Pada bab ini kita mempelajari operasi – operasi aritmatika dalam bahasa php. Dan banyak yang perlu di pelajari pada bab ini dan sangat terperinci dan juga mempelajari barisan huruf dan simbol dan contoh penggunaannya. Dalam php memiliki kesamaan dengan pelajaran matematika tentang operasi aritmatika. Operator dalam program PHP berfungsi untuk memanipulasi nilai. Dengan operator, sebuah fungsi dapat berjalan dan bersifat dinamis. Nilai-nilai pada sebuah operasi sering disebut dengan operan. Pada sebuah operasi perkalian 2 * 2, nilai 2 dan 2 dapat disebut dengan operan. (GetX Press, 2020).

7.2 Penggunaan Operand, operator, dan urutan operator dalam php

Operand adalah suatu nilai yang diperlkan didalam operasi, sedangkan operator adalah merupakan instruksi yang akan diberikan dalam mendapatkan suatu hasil berdasarkan proses tersebut. Dapat logikan konsep dasar operator kali ini seperti sistem penjumlahan saat pelajaran matematika agar lebih mudah dipahami. Apabila ada operasi 3+6(3 ditambah 6), angka 3 dan 6 dapat diartikan sebagai operand, sedangkan tanda tambah(+) merupakan operator. Ada beberapa jumlah operator terkadang

dapat mengubah hasil nilainya sendiri, meskipun teknik sebenarnya hanya sebagai hubungan antara operand. Terdapat tiga jenis operator adalalah unari, binary, dan ternary). Ada macam – macam jenis operator tersebut sangat beda dari pada yang di mendasarkan jumlah operand.

1. Operator unary yaitu suatu operator yang memiliki satu tanda operand, contohnya adalah tanda kurang (-). Tanda kurang berbentuk sebuah angka dan menjadi negatif, contohnya : -4, karakter + Untuk menegaskan yang nilainya positif dengan contoh adala : + 9.
2. Operator binary ialah merupakan operator dengan yang hanya memiliki dua operand. Misal, 8x5, 15/3.
3. Operator ternary merupakan suatu operator yang hanya dapat memiliki tiga operand. Dalam pelajaran PHP yanga hanya dapat mengenal satu operator ternary ialah operator kondisi [?:].

Dasar sistem operator dalam php sebenarnya memiliki persamaan dalam matematika. Persamaan dapat dilihat pada dua kesamaan yang mempunyai sistem pemrosesan terdiri dari (mana yang didahukukan dan mana yang terakhir).

Contoh

1. $hasil1=$A + $B / $C - $D;
2. $hasil2=$a and $B || $C AND $D;

PHP akan mengeksekusi operasi tersebut dengan menglihat urutan prioritasnya. Urutan prioritasnya didasarkan atas seberapa”dekat” operator dengan kedua operand-nya. Sebagai contoh,1+1*2 hasilnya adalah 3 bukan 4. Disebabkan karena operator kali [*] mempunyai nilai yang sangat tinggi dari pada operator penambahan (+). Apa bila ingin memaksa prioritas seperti kehendak pengguna, maka harus diletakan pada nilai yang akan diprioritaskan pada dalam kurung. Contoh, pada 1+2 *2, apa

bila ingin dijumlahkan 1(satu) dan 1(satu).Jadi, operasi dapat dituliskan menjadi (1+1)*2, Maka hasilnya adalah 4.

Terkadang Tanda kurung() dapat mempermda dalam membaca struktur program, meskipun tidak di perluka. Contoh $A And $B or $C, maka mudahkan dimengerti apabila ditulis ($a AND $b) OR $c, dengam seharusnya Operator (AND) Mempunyai Urutan system Prioritas yang Lebih Tinggi di bandingkan dengan OR.

Ada beberapa urutan operator pada konsep operasi ini yang di sediakan oleh PHP, Yaitu :

Tabel 7.1 : operator aritmatika

Simbol	Fungsi	Keterangan
+	**Penjumlahan**	**Penjumlahan untuk 2 operand (a + b)**
-	**Pengurangan**	**Pengurangan untuk 2 operand (a – b)**
*	**Perkalian**	**Perkalian untuk 2 operand (a * b)**
/	**Pembagian**	**Pembagian untuk 2 operand (a / b)**
%	**Modulus**	**Menghasilkan nilai hasil bagi (a %)**

Operator dapat di artikan sebagai simbol yang biasa diperlukan dalam perancangan program dalam melakukan sistem operasi atau cara manipulasi. beberapa operator C++ dapat digolongkan seperti operator sistem binary ialah operator dapat diubah dengan dua buah nilai.

Contoh :

b + b1

tanda ' + ' suatu operand yang dilakukan dalam penjumlahan nilai a dan b. yang dikarenakan oleh operator penjumlahan sehingga mempengaruhi 2 operand, operator binary dapat di artikan sebagai operator penjumlahan.

Contoh :

Tanda ‘– ‘ (kurang) suatu unary, dikarenakan hanya mempunyai sebuah operand.
Operator Aritmatika

Aritmatika merupakan ilmu hitung dasar matematika yang memiliki unsur-unsur dasar sebagai berikut : Penjumlahan(+), Pengurangan(-), Pembagian(/), Perkalian(x), untuk operasi aritmatika yang lebih canggih seperti Presentase, akar kuadrat, pemangkatan, dan logaritma).

7.3 Aritmatika dalam PHP

Operator ini digunakan untuk operasi matematika, misalnya : penambahan,pengurangan, perkalian, dan pembagian.

Tabel 7.2 : tabel operasi dalam PHP

Operator	Operasi	Keterangan
$a + $b	Penjumlahan	$a ditambah $b
$a - $b	Pengurangan	$a dikurangi $b
$a * $b	Perkalian	$a dikalikan $b
$a / $b	Pembagian	$a dibagi $b
$a % $b	Modulus	Sisa hasil $a dibagi $b

7.4 Operator Increment / Decrement

Operator ini merupakan suatu operator penambah dan pengurang nilai dengan nilai selisih **1 (satu)**. Apabila operator diletakkan sebelum variabel misalkan : **++$a** atau **--$a**, maka nilai **$a** akan ditambah atau dikurangi dengan nilai 1 (satu) sebelum operasi dijalankan, dan apabila operasi diletakkan setelah variabel misalkan **: $a++** atau **$a--,** maka nilai **$a** akan ditambah atau dikurangi dengan nilai 1 (satu) setelah operasidijalankan:

Tabel 7.3 : Operator Increment

Operator	Operasi	Keterangan Proses
++$a	Pre-increment	$a ditambahsatu, kemudian operasi dijalankan.
$a++	Post-increment	Operasi dijalankan, kemudian $a ditambah satu.
--$a	Pre-decrement	$a dikurangisatu, kemudian operasidijalankan.
$a--	Post-decrement	Operasi dijalankan, kemudian $a ditambah satu.

Contoh programnya:

```
<?
//operator.php
$a=1;
Echo"<br>operasi ++\$a nilai \$a = ".++$a;
Echo"<br>operasi \$a++ nilai \$a = ".$a++;
Echo"<br>operasi --\$a nilai \$a = ".--$a;
Echo"<br>operasi \$a-- nilai \$a = ".$a--;
```

Hasilnya di browser akan tampak seperti berut ini :

operasi ++$a nilai $a = 2
operasi $a++ nilai $a = 2
operasi --$a nilai $a = 2
operasi $a-- nilai $a = 2

7.4 Operator String

Operator String adalah operator yang digunakan untuk menggabungkan teks, baik teks dengan variabel maupun suatu teks dengan teks yang lain. Operator ini dideklarasikan dengan memberikan tanda titik (.) di antara operan dalam suatu operasi. Perhatikan contoh berikut

```
<?
$a = "Hallo";
$b = "Selamat balajar PHP";
$c= $a." ".$b;
Echo"<br>$c";
// Variabel $c berisikan "Hallo Selamat belajar PHP
?>
```

Apabila program tersebut dijalankan di browser, maka akan menghasilkan tulisan Hallo Selamat belajar PHP. Variabel **$c** adalah gabungan antara variabel **$a** yang bernilai "**Hallo**", teks **" "** dan variabel **$b** yang bernilai "**Selamat belajar PHP**".

7.5 Operator Bitwise

Operator ini berfungsi untuk menghasilkan nilai bilangan binary (bit), yaitu: 1 atau 0.

Tabel 7.4 : Operator Bitwise

Operator	Nama	Contoh	Keterangan
&	And	$a & $b	Bit akan bernilai 1 jika $a dan $bbernilai 1.
|	Or	$a | $b	Bit akan bernilai 1 jika $a atau $b bernilai 1.
^	Xor	$a ^ $b	Bit akan bernilai 1 jika $a atau $b, tetapi tidak keduanya bernilai 1.

Operator	Nama	Contoh	Keterangan
~	Not	~ $a	Bit akan bernilai 1 jika $a dan $b bernilai 0.
<<	Shift Left	$a << $b	Menggeser bit $a sebanyak $b langkah ke kiri. Setiap langkah mempunyai pengertian "dikali-kan dengan nilai dua".
>>	Shift Right	$a >> $b	Menggeser bit $a sebanyak $b langkah ke kanan. Setiap lang-kah mempunyai pengertian "dibagi dengan nilai dua".

7.6 Operator Perbandingan

Operator ini dapat digunakan saat struktur operator sebagai salah satu Kontrol struktur program seperti penggunaan **if** & **else if**, di mana akan dilakukan perbandingan antara dua nilai.

Tabel 7.5 : Operator Perbandingan

Operator	Nama	Contoh	Keterangan
>	Lebih dari	$a > $b	True apa bila $ a lebih besar dari $ b.
<	Lebih kecil	$a < $b	True jika $a lebih kecil dari $b.
>=	Lebih besar dan sama dengan	$a >= $b	True apa bila $ a lebih besar or sama dengan $ b.

<=	Lebih kecilor sama dengan	$a <= $b	True apa bila $ a lebih kecil or samadengan $ b.
==	Sama deng an	$ a = $ b	True apabila $ a sama dari $b.
!=	Tidak sama	$ a != $ b	True apabila $ a tidak sama dari $ b.

7.7 Operator Logika

Operator logika hampir memiliki fungsi yang sama dengan operator perbandingan, yaitu sebagai pengendali struktur program.

Tabel 7.6 : Operator Logika

Operator	Nama	Contoh
and &&	And	$a and $b $a && $b
or ||	Or	$a or $b $a || $b
xor	Eksklusif or	$a xor $b
!	Not	!$a

7.8 Operator Presedence

Presedence adalah urutan operasi sesuai dengan tingkatan operator. Jadi operasi perhitungan dijalankan sesuai dengan urutan / tingkatan yang lebih tinggi terlebih dahulu. Berikut ini adalah tabel prioritas eksekusi operator dari level tertinggi hingga paling rendah

Tabel 7.7 : Operator Presedence.

Nomor Prioritas	Operator
1	( ) { }
2	~ ! ++ -- $ &
3	* / %
4	+ -
5	<> <= >=
6	== !=
7	&
8	^
9	!
10	&&
11	||
12	= += -= *= /= &= |= ^= .=
13	AND (&&)
14	XOR(||)
15	OR

Perhatikan contoh berikut :

$hasil = 10 – 5 * 4

Hasil operasi ini akan menghasilkan nilai **-10** dan bukannya **20**, sebab operator *** (perkalian)** mempunyai level yang lebih tinggi daripada **– (pengurangan),** sehingga urutan operasinya adalah **5** dikalikan **4** terlebih dahulu kemudian digunakan untuk mengurangi **10**.

7.9 Operasi Nilai Variabel dari Form Html.

Salah satu ciri web yang dinamis adalah menyediakan form yang dapat dimanfaatkan pengunjung untuk mengirim informasi kepada server. Dengan demikian ada interaksi dua arah antara pengunjung dan webmaster (pemilik website), sehingga suatu

website menjadi lebih hidup. Dalam pembuatan form ini yang perlu diperhatikan adalah tag **<FORM>** beserta atributnya, seperti : **method**, **action** dan input data yang lainnya. Action adalah atribut yang menunjukkan isian atau informasi, bahwa data akan dikirimkan ke suatu file proses data yang akan diakses. Atribut method mempunyai dua macam pilihan nilai yaitu **POST** dan **GET**.

1. Jika menggunakan pilihan **GET**, maka semua isian dikirimkan ke server dalam satu kesatuan URL-nya
2. Jika menggunakan **POST** data isian dikirimkan ke server terpisah dengan teks URL-nya

Kelemahan dari method GET adalah data yang digunakan sebagai parameter akan tampil pada UR, selain itu method GET tidak dapat megirim data dalam jumlah yang besar. Kemudian yang tidak kalah pentingnya adalah name (nama) dari input data, karena nama dari kontrol data ini akan menjadi variabel yang membawa data isian dari form. Contoh dari aplikasi operasi perkalian dengan menggunakan form sebagai input data variabel adalah sebagai berikut:

```
<?
//dataform.php
If($submit){
Echo"Nilai Satu adalah : ".$a; Echo"<br>Nilai
Dua adalah : ".$b; Echo"<br>Nilai Satu * Dua
adalah : ".$a*$b;
}
?>
<form method="POST" action="dataform.php"><p>Nilai Satu :
<input type="text" name="a" size="14"></p><p>Nilai Dua :
<input type="text" name="b" size="14"></p><p><input
type="submit" value="Hitung" name="submit"></p></form>
```

Jika program di atas dijalankan dalam browser maka akan tampak tampilan gambar seperti pada Gambar 7.2. Kemudian isi form Nilai Satu denganangka 44 dan Nilai Dua dengan angka 66.

Gambar 7.1 : Hasil Penjumlahan

Apabila ditekan tombol Hitung, hasilnya dapat dilihat seperti gambar 7.2 berikut

Gambar 7.2 : Hasil Hitung

7.19 Operator pemberi nilai atau penugasan

Operator pemberian nilai (penugasan) yang disimbolkan dengan tanda (=) sama dengan, berguna untuk memberikan suatu

nilai kesuatu variable. Operator ini dapat digunakan sebagai ungkapan ataupun berdiri sendiri sebagai pernyataan.

$a = 3; memberikan nilai 3 ke $a

$a = ($b = 4) + 5; // $a is equal to 9 now, and $b has been set to 4.

$a = 2 + 3;

7.11 Operator Eksekusi

PHP mendukung sebuah operator eksekusi yang disimpan dalam tanda backticks (``). pada tanda tersebut maka dapat dimasukan perintah yang akan dilakukan pada shell, kemudian hasilnya akan dilihat pada browser.

Contoh

```
$output = `ls -al`;
echo "<pre>$output</pre>";
```

7.12 Operator Penaikan dan Penurunan

PHP mendukung perintah C-style yang berhubungan dengan penaikan (*increment*) dan penurunan (*decrement*). Kedua operator ini digunakan pada operand bertipe bilangan bulat (integer).. Increment/decrement Operators.

Tabel 7.8 : operator dan penurunan

Contoh	Nama	Hasil
++$a	Pre-increment	Naikan nilai $a sebanyak 1, kembalikan di $a.
$a++	Post-increment	Kembalikan $a, kemudian naikan $a sebanyak 1.
--$a	Pre-decrement	Turunkan $a sebanyak 1, kembalikan di $a.
$a--	Post-decrement	Kembalikan $a, kemudian turunkan $a sebanyak 1.

7.13 Operator Prioritas

Operator ini dapat menentukan bagaimana ekpresi yang dikerjakan dalam pemrograman php. Dan Dapat mempunyai system nilai yang lebih tinggi akan diutamakan duluan untuk hal pengerjaannya dari pada dengan operator uang yang memiliki nilai yang paling rendah.

Sebagai Contoh :

1+6*3 maka hasilnya 18 bukan dengan 21, karna tanda operator kali '*' lebih tinggi derajatnya dari pada operator dengan penjumlahan '+'.

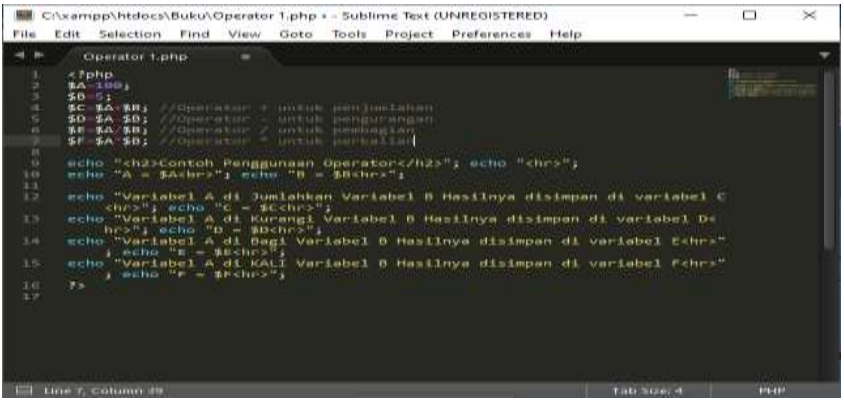
Berikut ini merupakan tabel operator prioritas yang mempunyai derajat yang lebih rendah akan disimpan pada awal.

Tabel 7.9 : Operator Prioritas

Associativity	Operators
left	,
left	Or
left	Xor
left	And
right	print
left	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
left	? :
left	||
left	&&
left	|
left	^
left	&
non-associative	== != ===
non-associative	< <= > >=
left	<< >>
left	+ - .
left	* / %
right	! ~ ++ -- (int) (double) (string) (array) (object) @
right	[
non-associative	new

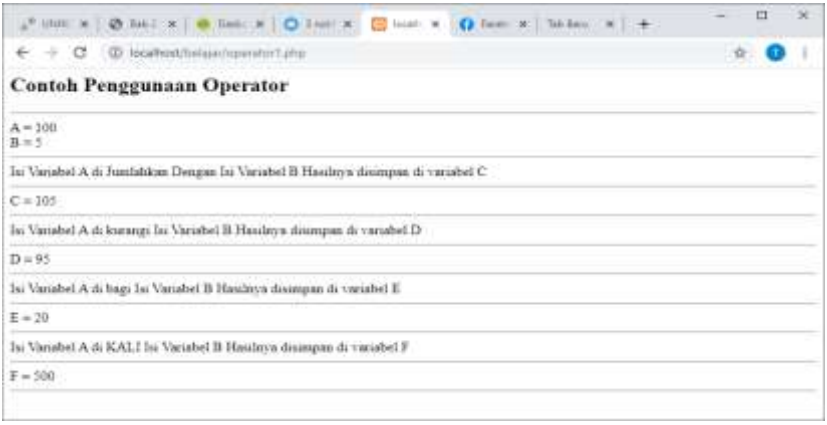
Latihan 1

Untuk dapat lebih di pahami, maka perlu diberikan latihan dan di simpan dengan nama file operator 1. Phpx



Gambar 78.3 : Latihan

Setelah dijalankan programnya Maka dapat hasilnya seperti



Gambar 7.4 : Hasil Latihan

DAFTAR PUSTAKA

Robin Nixon. 2012. *Learning PHp.* Jakarta: O'Reilly

Elgamar, S,Kom., M.Kom. 2020.*konsep dasar Pemrograman website dengan php*. Kota malang: Cv multimedia edukasi.

Aziz, M. Farid. 2005. Object Oriented Programing dengan PHP5, Jakarta: PT Elex Media Komputindo.

Bogdan, R. & Biklem, S. 1982. Qualitative research for education: An indtroduction to theory and methods.

Boston, MA:Allyn and Bacon. Creswell, J. 1998. Qualitative inquiry and research design: choosing among five traditions.

Thousand Oaks, CA: Sage Publications, Inc.

Gata, Windu. 2012. Asiknya Mengenal Java, Jakarta: PT Elex Media Komputindo.

Indrajani dan Martin. 2007. Pemrograman Berbasis Objek Dengan Bahasa Java, Jakarta: PT Elex Media Komputindo.

BAB 8
PERULANGAN

Oleh Mundzir

Perulangan (*looping*) adalah sebuah instruksi untuk mengeksekusi sejumlah pernyataan secara berulangkali selama kondisi tertentu masih terpenuhi. Ada 3 jenis instruksi perulangan yakni *while*, *do-while*, dan *for*.

8.1 While Loop

Jenis perulangan ***while loop*** ini digunakan untuk mengulang eksekusi sebuah blok pernyataan selama ***boolean_condition*** masih terpenuhi (bernilai *true*). (Priyanto Hidayatullah dkk, 2017).

Bentuk instruksi *while* :

```
while (boolean_condition) {
statement1;
statement2;
  .....
}
```

Contoh Javascript :

```
<html>
<head>
    <title>Belajar Web</title>
    <script language=javascript>
        var batas = 10;
        var nilai = 1;
        while (nilai <= batas) {
            document.write("Angka " +nilai+ "<br>");
            nilai++;
```

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

</body>
</html>
```

Contoh PHP :

```
<?php
$nilai = 1;
$batas = 10;
while ($nilai <= $batas) {
echo ”Angka $nilai <br>”;
$nilai++;
}
?>
```

8.2 Do-While Loop

Struktur kontrol ***do-while loop*** adalah untuk mengeksekusi sebuah blok pernyataan selama ***boolean_condition*** masih terpenuhi (bernilai *true*). Perulangan ini mirip dengan ***while-loop***, bedanya adalah pada ***do-while loop,*** perulangan yang pertama tidak diperiksa dulu. Itu berarti perulangan akan dieksekusi minimal sekali. Sedangkan pada perulangan ***while***, semua perulangan harus diperiksa. (Priyanto Hidayatullah dkk, 2017).

Bentuk instruksi *do-while* :

```
do {
statement1;
statement2;
  .....
} while (boolean_condition);
```

Contoh Javascript :

```
<html>
<head>
<title>Belajar Web</title>
<script language=javascript>
var batas = 10;
var nilai = 1;
do {
document.write("Angka " +nilai+ "<br>");
nilai++;
} while (nilai <= batas);
</script>
</head>
<body>

</body>
</html>
```

Contoh PHP :

```
<?php
$nilai = 1;
$batas = 10;
do {
echo "Angka $nilai <br>";
$nilai++;
} while ($nilai <= $batas);
?>
```

8.3 For Loop

Instruksi ***for loop*** yaitu memiliki batas yang lebih jelas dan melakukan perulangan untuk mengeksekusi sebuah blok pernyataan beberapa kali selama ***LoopCondition*** terpenuhi (bernilai *true*). (Priyanto Hidayatullah dkk, 2017).

Bentuk instruksi *for* :

```
for (Initialization; LoopCondition; StepExpression) {
statement1;
statement2;
  .....
}
```

Contoh Javascript :

```
<html>
<head>
<title>Belajar Web</title>
<script language=javascript>
var batas = 10;
for (nilai=1; nilai <= batas; nilai++) {
document.write("Angka " +nilai+ "<br>");
}
</script>
</head>
<body>

</body>
</html>
```

Contoh PHP :

```
<?php
$batas = 10;
for ($nilai=1; $nilai <= $batas; $nilai++) {
echo "Angka $nilai <br>";
}
?>
```

DAFTAR PUSTAKA

Avestro, J. 2007. *Pengenalan Pemrograman 1*. *Jeni. Juni*.

Hidayatullah, P., & Kawistara, J. K. 2017. *Pemrograman Web Edisi Revisi*. Informatika.

BIODATA PENULIS

Leo Willyanto Santoso
Dosen Program Studi Informatika
Fakultas Teknologi Industri Universitas Kristen Petra

Penulis merupakan dosen tetap di Program Studi Informatika Universitas Kristen Petra Surabaya. Penulis telah menyelesaikan pendidikan S2 di University of Melbourne, Australia

BIODATA PENULIS

Sulistyowati, S.T., M.Kom.
Dosen Jurusan Sistem Informasi
Fakultas Teknik Elektro dan Teknologi Informasi
Institut Teknologi Adhi Tama Surabaya

Penulis lahir di Banyuwangi tanggal 5 April 1977. Penulis adalah dosen tetap pada Jurusan Sistem Informasi Fakultas Teknik Elektro dan Teknologi Informasi, Institut Teknologi Adhi Tama Surabaya mulai tahun 2000 sampai sekarang. Penulis menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika di Institut Teknologi Adhi Tama Surabaya dan melanjutkan pendidikan S2 pada Jurusan Teknik Informatika di Institut Teknologi Sepuluh Nopember Surabaya. Bidang riset penulis adalah di bidang Riset Operasi, Rekayasa Perangkat Lunak, Sistem Pendukung Manajemen, dan Pemrograman. Kritik dan saran yang membangun dapat disampaikan melalui email sulis_tyowati@itats.ac.id.

BIODATA PENULIS

Wahyuddin S, S.Kom., M.Kom
Dosen STMIK Amika Soppeng

Wahyuddin S. was born at Malaka-Bone-Sulawesi Selatan in 1992. In 2011 he attended STMIK Dipanegara Makassar and was completed in 2015. He was completed after attending 7 semesters and active on an XPcom (Extreme Programmer Computer) campus organization. He was also active as a lecturer assistant for three semesters and taught several courses on programming. He continued his Master of Information systems at UNIKOM Bandung in 2016 and was completed in April 2019. He worked as a lecturer at a campus (STMIK AMIKA Soppeng) 2019 to present and also a Freelance Web Programmer. Has competence in the field of software engineer, application developer, multimedia, web developer, network security, and data analyst.

BIODATA PENULIS

Diana, S.Kom, M.Kom
Dosen Program Studi Teknik Informatika Fakultas Teknik
Universitas Muhammadiyah Bengkulu

Penulis lahir di Manna Bengkulu Selatan Provinsi Bengkulu. Penulis adalah dosen tetap pada Program Studi Teknik Informatika Fakultas Teknik Universitas Muhammadiyah Bengkulu. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika dan melanjutkan S2 pada Jurusan Ilmu Komputer. Penulis menekuni bidang Ilmu Komputer khususnya AI .

BIODATA PENULIS

Andy Rachman
Dosen Jurusan Teknik Informatika – Institut Teknologi Adhi Tama Surabaya

Andy Rachman adalah dosen Jurusan Teknik Informatika – Institut Teknologi Adhi Tama Surabaya. Penulis sampai buku ini ditulis merupakan Ketua Jurusan Teknik Informatika. Lahir di Surabaya tanggal 22 Februari 1977. Saat in penulis aktif sebagai Reviewer dibeberapa Jurnal Nasional maupun Jurnal Internasional. Penulis juga merupakan Asesor Kompetensi BNSP bidang TIK. Penulis saat ini juga aktif diberbagai keanggotaan bidang ilmu, mulai dari ACM, IEEE, KODEPENA, bahkan saat ini penulis telah mendapatkan sertifikat kompetensi bidang Penulis Non Fiksi dari BNSP. Saat ini penulis telah memiliki 30 buku ber-ISBN dan mendapatkan penghargaan sebagai Dosen Penulis Buku Terproduktif 1 di Kampus ITATS. Penulis juga memiliki 2 buah *book chapter* Internasional. Bidang penelitian penulis adalah Rekayasa Perangkat Lunak, Pemrograman Game dan Realitas Virtual, Interaksi Manusia dan Komputer, Sistem Operasi, Teknologi Informasi, dan UI/UX Design.

BIODATA PENULIS

Ronny Makhfuddin Akbar, S.Kom., M.Kom.
Dosen Program Studi Teknik Informatika

Ronny Makhfuddin Akbar lahir di Mojokerto, pada tanggal 9 Juli 1986. Mendapatkan gelar Magister Komputer dari Institut Sains dan Teknologi Terpadu Surabaya pada tahun 2014. Berprofesi dosen program studi Teknik Informatika di Perguruan Tinggi Swasta di Mojokerto. Mata kuliah yang diampu penulis antara lain Pemrograman Multimedia, Pemrograman Berorientasi Objek, Pemrograman Mobile, Komputer Grafis, dan Pengolahan Citra Digital. Selain itu penulis masih aktif menulis baik publikasi penelitian atau pengabdian kepada masyarakat. Mempunyai beberapa sertifikat di bidang Android, Fullstack Web, Azure, AWS, Machine Learning, serta sertifikasi global TensorFlow.

BIODATA PENULIS

Eferoni Ndruru, S.Kom., M.Kom.
Dosen Program Studi Manajemen Informatika
Fakultas Ilmu Komputer dan Teknologi Informasi
Universitas Budi darma Medan

Penulis lahir di Padang tanggal 27 Maret 1991. Penulis adalah dosen tetap pada Program Studi Manajemen Informatika Fakultas Ilmu Komputer Dan Teknologi Informasi, Universitas Budidarma Medan. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika dan melanjutkan S2 pada Jurusan Teknik Informatika. Penulis menekuni bidang Menulis.

BIODATA PENULIS

Mundzir, S.Kom., M.T
Dosen Program Studi Sistem Informasi
Fakultas Ilmu Komputer Universitas Mulia

Penulis lahir di Makassar tanggal 08 Agustus 1975. Penulis adalah dosen tetap pada Program Studi Sistem Informasi Fakultas Ilmu Komputer, Universitas Mulia. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika dan melanjutkan S2 pada Jurusan Teknik Elektro. Penulis menekuni bidang ilmu komputer.