



Algoritma JavaScript Dasar

Silver- Chapter 1 - Topic 3

**Selamat datang di Chapter 1 Topic 3 pada course
React Native dari Binar Academy!**





Haalloo Binarian! 🙌

Chapter 1 belum selesai nih. Masih seputar JavaScript, di **Topic 3** React Native ini, kita akan belajar tentang **dasar algoritma, alur pengambilan keputusan pada pemrograman** dan **alur pengulangan pada pemrograman**.

Kayak biasa ya, jangan cuma dihafal tapi juga harus dipahami lhoo..



Detailnya, kita bakal bahas hal-hal berikut ini:

- Algoritma
- Alur Pengambilan Keputusan
- Alur Perulangan





Menurut kamu, **ALGORITMA** itu apa sih?





Hidup dan masalah seperti dua mata uang yang nggak bisa dipisahin, jadi menghindari masalah bukanlah solusi. Justru kita harus bisa berpikir cerdas supaya masalah bisa terpecahkan.

Kalau dianalogikan dengan cara mengatasi masalah kehidupan, **Algoritma** adalah *step by step* untuk menyelesaikan sebuah masalah.





“Setiap hari selalu saja ada masalah ya!? Gimana nih ngatasinnya?”

Nah ini nih.

Ketika kamu berpikir untuk menyelesaikan masalah selangkah demi selangkah, maka sebenarnya kamu lagi menerapkan algoritma.

**PENGEN TOPUP
PULSA EUY..!**





Contohnya pas lagi deadline, eh pulsa kamu malah habis. Jadilah kamu harus beli voucher pulsa dulu di kios langganan.

Tapi apakah habis beli voucher masalah kamu selesai? Belum. Kamu masih harus gosok bagian kartu supaya **kode voucher** bisa dipakai.

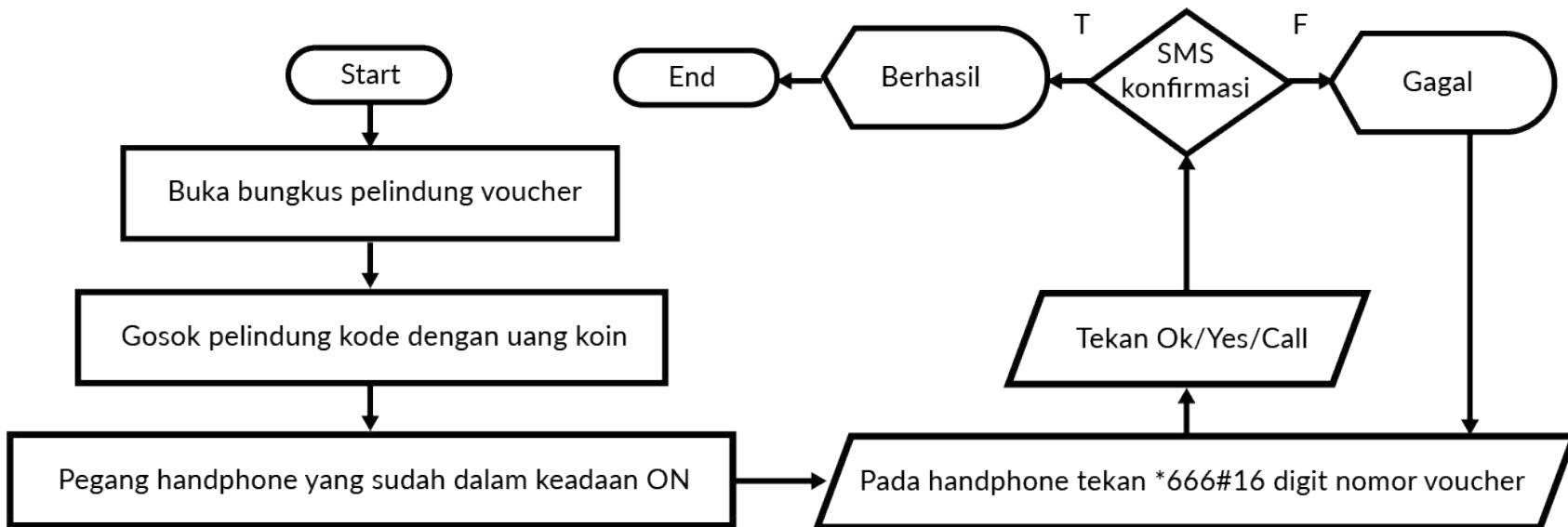
Kalau *input* kodennya benar, proses *top-up* berhasil, tapi pulsa nggak akan terisi kalau kamu salah input kode.





Proses top-up pulsa dengan voucher tadi adalah contoh algoritma.

Lebih jelas, algoritmanya isi pulsa bisa direpresentasikan kayak gambar berikut ini nih:





Eh tapi untuk bikin gambar
algoritma, caranya gimana ya?



REPRESENTASI ALGORITMA

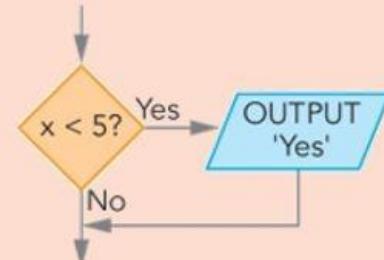


Untuk bikin algoritma, yang perlu kamu lakukan sebelum menulis kode adalah tentuin alur algoritmanya dulu.

Ada 2 opsi yang bisa kamu pakai untuk bisa nyusun alur algoritma, yaitu:

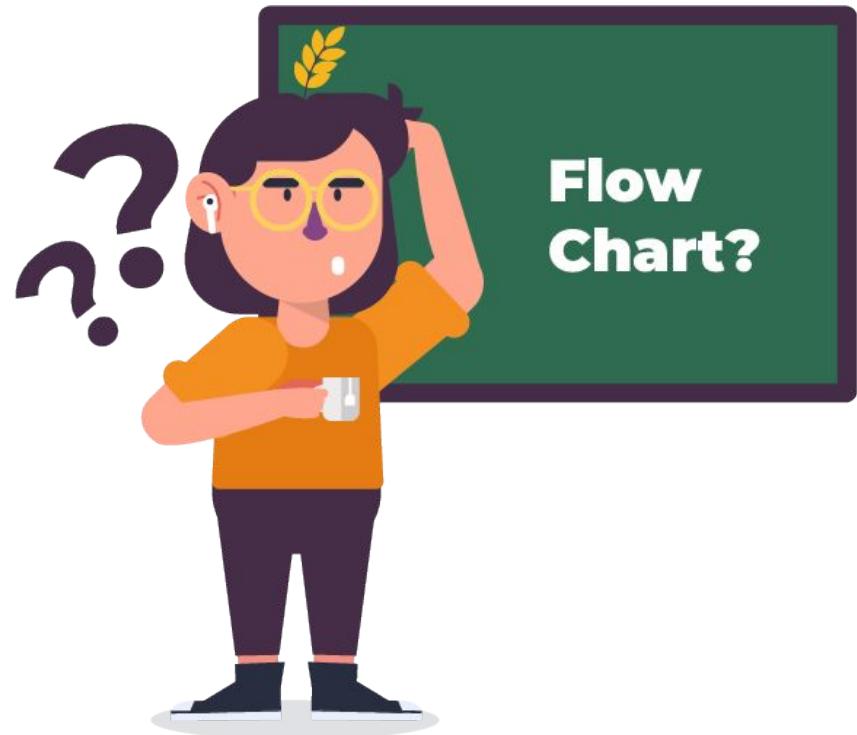
Opsi pertama

Opsi kedua

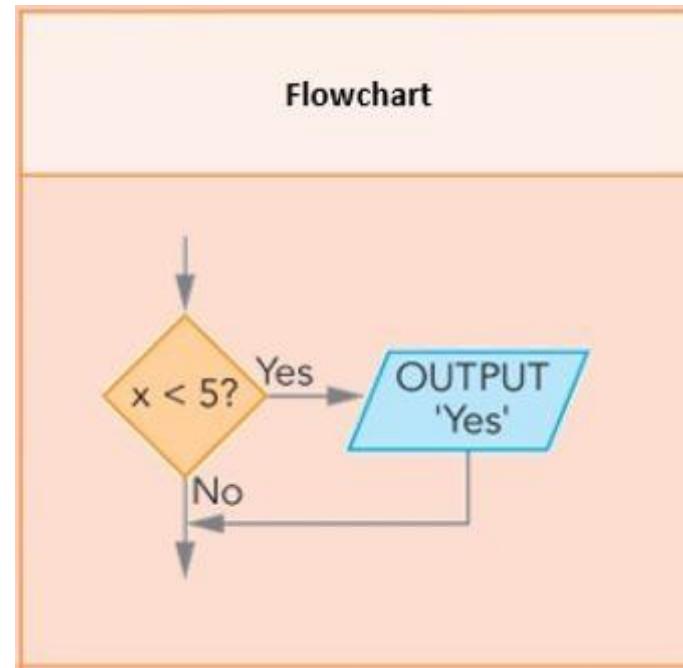
Flowchart	Pseudo-code
	IF $x < 5$ THEN OUTPUT 'Yes' ENDIF

Opsi Pertama

Gimana cara bikin gambar **flowchart?**



Flowchart memakai **berbagai simbol** untuk **gambarin algoritma**.

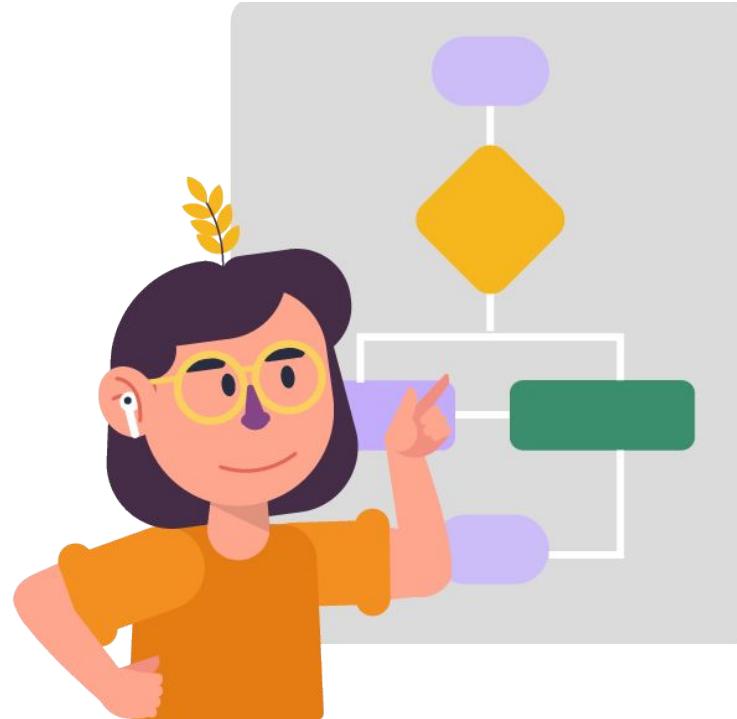


Flowchart

Adalah representasi visual dari *control flow* suatu algoritma.

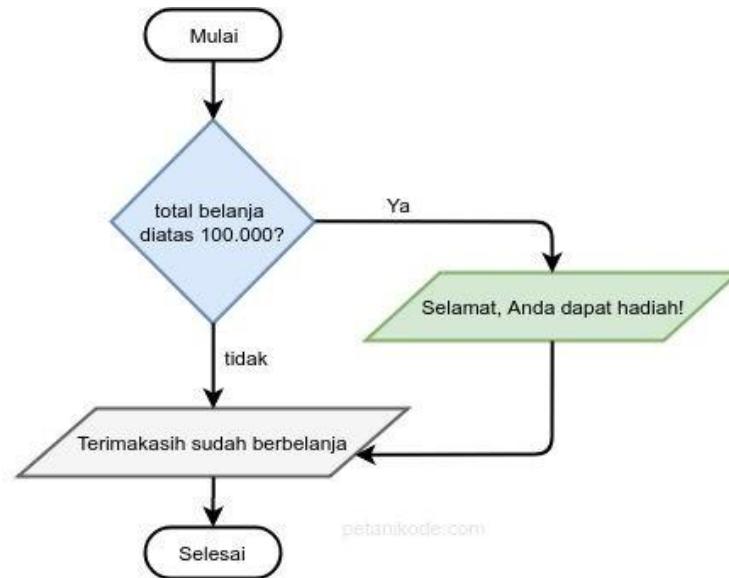
Representasi ini nunjukin beberapa hal, yaitu:

- **Statement** yang perlu **dieksekusi**,
- **Keputusan** yang perlu **dibuat**,
- **Flow logika** (untuk iterasi dan tujuan lain)
- **Terminal** yang menunjukkan **titik awal dan akhir**



Flowchart sederhana ini nunjukin sebuah alur untuk mengetahui suatu keputusan, yaitu apakah setelah belanja kamu berhak dapat hadiah atau nggak. Dalam kasus ini, kamu bakal dapat hadiah kalau total belanjamu lebih dari 100 ribu rupiah.

Contoh flowchart sederhana:



Simbol-simbol flowchart!

Kamu sadar nggak sih? Pada flowchart ada berbagai bentuk simbol yang dipakai. Simbol ini juga jadi keunikan flowchart, karena itu..

Sebelum bikin flowchart, sebaiknya kamu tahu dulu nih berbagai bentuk simbol ini dan fungsinya. Perhatikan dan pahami simbol flowchart pada tabel di samping.

	Flow Direction symbol Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga connecting line.
	Terminator Symbol Yaitu simbol untuk permulaan (start) atau akhir (stop) dari suatu kegiatan
	Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses dalam lembar / halaman yang sama.
	Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses pada lembar / halaman yang berbeda.
	Processing Symbol Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer
	Simbol Manual Operation Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh computer
	Simbol Decision Simbol pemilihan proses berdasarkan kondisi yang ada.
	Simbol Input-Output Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya
	Simbol Manual Input Simbol untuk pemasukan data secara manual on-line keyboard
	Simbol Preparation Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage.
	Simbol Predefine Proses Simbol untuk pelaksanaan suatu bagian (sub-program)/prosedure
	Simbol Display Simbol yang menyatakan peralatan output yang digunakan yaitu layar, plotter, printer dan sebagainya.
	Simbol disk and On-line Storage Simbol yang menyatakan input yang berasal dari disk atau disimpan ke disk.
	Simbol magnetik tape Unit Simbol yang menyatakan input berasal dari pita magnetik atau output disimpan ke pita magnetik.
	Simbol Punch Card Simbol yang menyatakan bahwa input berasal dari kartu atau output ditulis ke kartu
	Simbol Dokumen Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas.

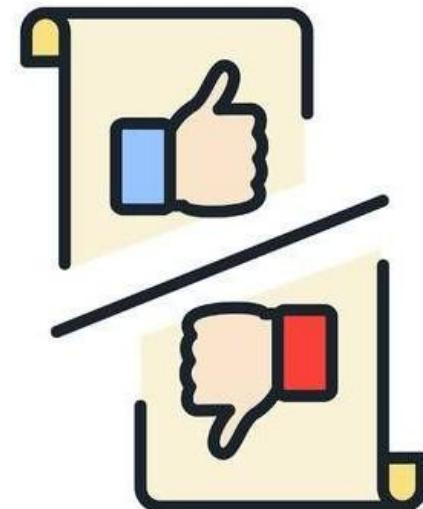
Kelebihan dan kekurangan *flowchart*

Kelebihan:

1. Menyajikan *control flow* algoritma secara visual. Jadi lebih gampang untuk ngerti alur penyelesaian masalah atau kondisi.
2. Lebih gampang mendeteksi kesalahan atau ketidakakuratan *flowchart* yang kompleks dan detail.

Kekurangan:

1. Pengerjaannya lumayan lama. Kamu harus atur posisi, ngasih label, dan hubungin simbol-simbolnya.
2. Kalau bikin *flowchart* pakai *tools* khusus, justru menghambat pemahaman tentang algoritma.



Opsi Kedua

Bagaimana cara bikin **pseudocode?**





Kamu juga bisa merepresentasikan algoritma dengan **pseudocode**.

Kalau dengan pseudocode, kamu bisa pakai **teks** atau **kata-kata** untuk nunjukin suatu alur.

Opsi kedua

Pseudo-code

```
IF x < 5 THEN  
    OUTPUT 'Yes'  
ENDIF
```

Contoh pseudocode sederhana



BACA dan SIMPAN "angka pertama" di variabel (A)
BACA dan SIMPAN "angka kedua" di variabel (B)
HITUNG "angka pertama" ditambah "angka kedua"
SIMPAN hasil perhitungan di variabel (SUM)
TAMPILKAN hasil perhitungan (SUM)

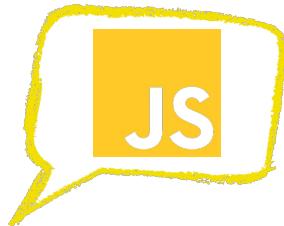


READ "the first number" and SAVE in the variable (A) READ
"the second number" and SAVE in the variable (B) SUM the
both numbers
SAVE the result in the variable (SUM)
PRINT the result (SUM)

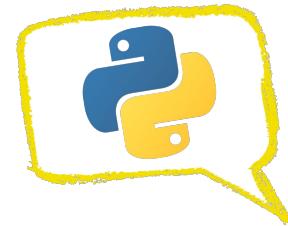
Cata tan:
Karena sintaksnya fleksibel,
maka tidak ada aturan yang
tegas untuk menuliskan
pseudocode.



Setelah kamu bikin pseudocode seperti contoh tadi,
sekarang tinggal kamu konversikan ke dalam bahasa pemrograman.



```
let no1, no2, no3;
no1 = prompt("Nomor pertama?");
no2 = prompt("Nomor kedua?");
hasil = Number(no1) + Number(no2);
console.log(hasil);
alert("Hasil = " + hasil);
```



```
no1 = input("Nomor pertama? ")
no2 = input("Nomor kedua? ")
hasil = int(no1) + int(no2)

print("Hasil", hasil)
```

Catatan:

Jika masalah yang ingin diselesaikan kompleks, dengan membuat pseudocode terlebih dahulu akan memudahkan kita ngoding nantinya .



Sudah tahu cara bikin alur algoritma,
checked!

Selanjutnya, gimana caranya bikin
alur pengambilan keputusan?



Pilih Ini

Pilih Itu

Pilih Semua



ALUR PENGAMBILAN KEPUTUSAN

Dalam kegiatan sehari-hari, kita selalu melakukan **tindakan yang berbeda**, tergantung **kondisinya**.

Misalnya, pas kita mau pergi keluar rumah dengan kendaraan. Kita akan perhitungin ramalan cuaca untuk nentuin kendaraan apa yang paling nyaman kita pakai berdasarkan kondisi cuaca.





Karena itulah kita butuh **alur pengambilan keputusan**.

Dengan skema tersebut, kita bisa ngambil langkah tepat sesuai dengan kondisi.

Alur pengambilan keputusan ini bisa pakai beberapa cara, yaitu:

1. **Kondisional *if-else***
2. **Kondisional operator ('?)**
3. **Switch case**



Kondisional *if-else*



1. Kondisional *if-else*



```
const umurPeserta = 24;  
  
if(umurPeserta > 20) console.log('Peserta lolos');
```



```
const umurPeserta = 24;  
  
if(umurPeserta > 20){  
    console.log('Umur peserta lebih dari 20');  
    console.log('Peserta lolos');  
}
```

Pernyataan ***if(...)*** akan **mengevaluasi** kondisi di dalam **tanda kurung ()**. Jika hasilnya **sesuai (true)**, maka kode yang ada di dalam blok akan **dieksekusi**.

Kalau kamu mau menjalankan **lebih dari satu pernyataan**, maka kamu harus **bungkus/blok kode** dengan **tanda kurung kurawal {}**.



Nah, sebenarnya **statement if** akan ngevaluasi ekspresi di dalam **tanda kurung (...)**, lalu akan ngubah hasilnya jadi tipe data **boolean**



Catatan:

- Angka 0, string kosong ("") dan **false** menghasilkan nilai **false**, sehingga disebut nilai **falsy**.
- **falsy** yang lainnya akan menjadi **true**, sehingga disebut nilai **truthy**.

Contoh:

Kode dalam kondisi ini tidak akan dijalankan:

```
if (0) { // 0 adalah falsy
  ...
}
```

Sebaliknya, kode dalam kondisi di bawah ini akan dijalankan:

```
if (1) { // 1 adalah truthy
  ...
}
```

Kode dengan nilai boolean yang telah dievaluasi ke if:

```
let cond = (tahun == 2015); // true
atau false
if (cond) {
  ...
}
```



Contoh statement else:

Pernyataan **if(...)** bisa berisi blok kode lain yaitu **else**, Pernyataan **else** ini bakal dijalankan hanya saat **kondisi nggak sesuai atau salah**.

```
const umurPeserta = 24;

// Jika umur peserta lebih dari 20
// Maka peserta tersebut lolos
// Jika kurang dari dua puluh
// Maka tidak lolos
if(umurPeserta > 20) {
    console.log('Peserta lolos!!')
} else {
    console.log('Peserta Tidak Lolos!')
}
```



Contoh **statement if-else if:**

Kalau kamu mau nguji beberapa kondisi, kamu bisa pakai **statement if-else if**.

JavaScript akan memeriksa kondisi pertama. Kalau kondisinya sesuai, maka akan dieksekusi. Kalau nggak sesuai, kondisi kedua akan diperiksa. Kalau kondisi kedua masih nggak sesuai, maka kondisi terakhir yang akan dijalankan.



```
const nilaiUjian = 80;

// Jika nilai ujian lebih dari atau sama dengan 85 maka dapat A
// Jika nilai ujian kurang dari 85 dan lebih dari atau sama dengan 70 maka dapat B
// Jika nilai ujian kurang dari 70 dan lebih dari atau sama dengan 60 maka dapat C
// Jika kurang dari 60 dapat D

if(nilaiUjian >= 85) {
    console.log('Mantap, kamu dapat A Gan!')
} else if(nilaiUjian >= 70 && nilaiUjian < 85) {
    console.log('Selamat, kamu dapat B!')
} else if(nilaiUjian >= 60 && nilaiUjian < 70) {
    console.log('Yahhh :, kamu dapat C!')
} else {
    console.log('Yok bisa yok, kamu dapat D loh!')
}
```



Kondisional *Operator ('?')*



2. Kondisional operator

'?'

Kadang kita perlu netapin variabel **tergantung pada suatu kondisi yang ada**.

Di contoh ini, sejak awal kita sudah lebih dulu nyiapin variabel dengan nama "akses". Variabel itu akan ditetapkan tergantung dari kondisinya.

Misal, kalau umur kita lebih dari 18, maka kondisi pertama kita tetapkan agar terpenuhi sehingga kita mendapatkan akses masuk. Kalau nggak, maka kita dilarang masuk.

```
let akses;  
const umur = 17;  
  
if(umur > 18){  
    akses = true;  
} else {  
    akses = false;  
}  
  
console.log(akses)  
// akan menghasilkan false
```

3. Switch case

Pada bahasan sebelumnya, kamu sudah belajar tentang penggunaan If-else. Nah pemakaian If-else ini dapat diubah bentuknya jadi lebih simpel kalau suatu kondisi punya banyak pilihan. Caranya, pakai pernyataan **switch case**.

Contoh:

Dengan statement switch, kita bisa dengan gampang ganti kostum suatu karakter permainan online dengan banyak opsi tergantung kondisi mood-nya.





Switch case syntax

```
switch(x) {  
    case 'nilai1': // if (x === 'nilai1')  
        ...  
        [break]  
  
    case 'nilai2': // if (x === 'nilai2')  
        ...  
        [break]  
  
    default:  
        ...  
        [break]  
}
```

- Nilai-nilai diperiksa untuk dilakukan **perbandingan yang ketat**. Pada contoh, 'nilai1', 'nilai2', dan seterusnya akan dibandingkan.
- Kalau **ada persamaan**, switch akan menjalankan kode di tempat case yang sesuai.
- **Kalau nggak tidak ada case yang cocok**, maka kode default dieksekusi (jika ada).



Contoh *switch case*

```
let a = 2 + 2;

switch (a) {
    case 3:
        alert( 'Terlalu kecil' );
        break;
    case 4:
        alert( 'Benar sekali' );
        break;
    case 5:
        alert( 'Terlalu besar' );
        break;
    default:
        alert( "Tidak tahu" );
}
```





Alur Perulangan





Setiap pemrograman memungkinkanmu untuk **ngulangin suatu tindakan**.

Misalnya pas belanja online, kamu mau **nambahin 10 item** atau mau **ngurangin 5 item** di keranjang belanja.

Nah, dengan menerapkan **alur perulangan (loops)**, kamu jadi lebih gampang untuk memakai kode yang sama beberapa kali.



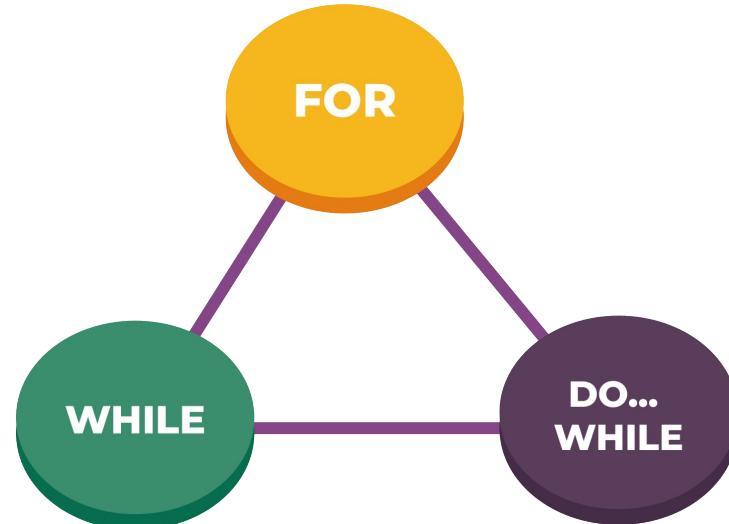
Jenis alur perulangan:

Ada **tiga jenis alur perulangan** yang perlu kamu tahu, yaitu:

1. “*for*” loop
2. “*while*” loop
3. “*do...while*” loop

Bagaimana cara aplikasinya?

Yuk, langsung saja lanjut ke halaman selanjutnya.





Contoh sederhana alur perulangan dengan “for”

Alur perulangan *for* bisa digambarin dalam kegiatan sehari-hari, misalnya petugas kebersihan yang ngepel lantai di sebuah gedung.

Ia akan mencuci kain pel, memeras pel, lalu mengepel sampai ke lantai lima, kemudian istirahat.



Catatan:

Pengulangan akan terus dilakukan sampai kondisi tertentu tercapai. Saat kondisi tersebut tercapai, maka perulangan akan dihentikan.



Nah, kalau dituliskan, alurnya kayak ini:

1. Ambil alat pel
2. Cuci kain pel pakai air bersih
3. Peras kain pel
4. Pel lantai secara urut
5. **Lakukan tahap 2-4, naik ke lantai berikutnya sampai ke lantai 5**
6. Istirahat



“for” loop syntax:

```
for (inisialisasi; kondisi; pengubah nilai) {  
    // ... pernyataan/perintah ...  
}
```

Contoh penulisan “for” loop

(dari alur sederhana yang sebelumnya):

```
for (let i = 1; i <= 5; i++) {  
    //mengepel lantai dari lantai 1 sampai 5  
    alert(i);  
}
```

Catatan:

- Inisialisasi:
Untuk menentukan variabel awal untuk pengulangannya.
- Kondisi:
Untuk menghentikan fungsi pengulangan.
- Pengubah nilai:
Untuk mengaktifkan pengulangannya, bisa bertambah atau berkurang.



Contoh sederhana alur perulangan dengan “while”

Nah, sekarang kita coba ubah contoh kondisi alur perulangan “for” jadi kondisi alur perulangan dengan “while” yuk.

Petugas mau meriksa adanya noda pada lantai. Jadi, selama ada noda di lantai, maka ia akan ngulang tahap kedua sampai kelima secara terus-menerus.



Kalau ditulis, alurnya akan seperti ini:

1. Ambil alat pel
2. **Periksa adanya noda pada lantai**
3. Cuci kain pel pakai air bersih
4. Peras kain pel
5. Pel lantai berurutan
6. Ulangi tahap 2-5

Cata tan:

Pengulangan akan dilakukan bila suatu kondisi tercapai atau benar. Selama kondisi **itu** benar dilakukan, perulangan akan terus



“while” loop syntax:

```
while (kondisi) {  
    // ... pernyataan/perintah ...  
}
```

Contoh penulisan “while” loop

(dari alur sederhana yang sebelumnya):

```
let i = 1  
while (i<=5) {  
    //mengepel lantai dari lantai 1 sampai 5  
    alert(i);  
    i++;  
}
```

Catatan:
Pada contoh ini, jika `i++` dihilangkan maka loop berulang secara terus menerus atau tidak terhingga.
Pada kasus ini, browser menyediakan cara untuk menghentikan loop.
Tapi, kalau JavaScript dari sisi server kita yang harus menghentikan prosesnya.



Gimana ya kalau kita ingin menghentikan perulangan yang terus-menerus tanpa henti (*infinite loop*)?

Contoh *infinite loop*:



```
const lantaiBersih = true;
let lantaiGedung = 5;

while (lantaiBersih) {
    alert(`Saya sudah mengepel lantai
${lantaiGedung}.`);
    lantaiGedung--;
}
```

Solusi *infinite loop*:



```
const lantaiBersih = true;
let lantaiGedung = 5;

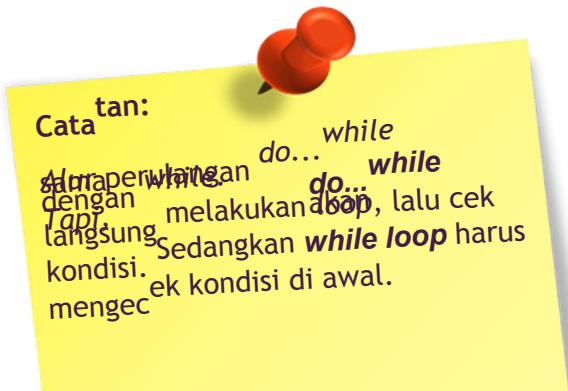
while (lantaiBersih) {
    alert(`Saya sudah mengepel lantai
${lantaiGedung}.`);
    lantaiGedung--;
    if (lantaiGedung === 0) {
        alert ("Semua lantai sudah
bersih."); break;
}
}
```



Contoh sederhana alur perulangan dengan “do...while”

Sekarang kita ubah contoh kondisi alur perulangan “while” jadi kondisi alur perulangan dengan “do...while”.

Petugas akan melakukan tahap pertama sampai keempat terlebih dahulu. Kemudian, ia memeriksa kondisi kain pel. Jika kain pel sudah kotor, maka ia akan mengulang langkah kedua hingga kelima.



Jika kita ubah, alurnya akan seperti ini:

1. Ambil alat pel
2. Cuci kain pel pakai air bersih
3. Peras kain pel
4. Pel lantai berurutan
5. **Setelah dirasa kain pel sudah kotor**
6. Lakukan tahap 2-5 (kalau kain pel nggak kotor, nggak perlu diulang)



“do...while” loop syntax:

```
● ● ●  
do {  
    // ... pernyataan/perintah ...  
} while (kondisi);
```

Contoh penulisan “do...while” loop
(dari alur sederhana yang sebelumnya):

```
● ● ●  
let i = 1  
do {  
    //mengepel lantai dari lantai 1  
    sampai 5  
    alert(i);  
    i++;  
} while (i<=5)
```

Catatan:
Loop akan langsung mengeksekusi pernyataan/perintah di awal, pernyataan akan melakukan pengecekan kemudian. Jadi, jika kondisinya masih true kode akan dijalankan lagi.

Saatnya kita Quiz!





1. Berdasarkan perkondisian pada kode di samping, apa yang akan terjadi?

- A. Akan kena error undefined
- B. Pada console log akan muncul “Peserta Terdaftar”
- C. Pada console log akan muncul “Peserta Tidak Terdaftar”

```
const namaPeserta = "";

if(namaPeserta){
    console.log('Peserta Terdaftar')
} else{
    console.log('Peserta Tidak Terdaftar')
}
```



1. Berdasarkan perkondisian pada kode di samping, apa yang akan terjadi?

- A. Akan kena error undefined
- B. Pada console log akan muncul “Peserta Terdaftar”
- C. Pada console log akan muncul “Peserta Tidak Terdaftar”

```
● ● ●  
const namaPeserta = "";  
  
if(namaPeserta){  
    console.log('Peserta Terdaftar')  
} else{  
    console.log('Peserta Tidak Terdaftar')  
}
```

Karena variabel `namaPeserta` nilainya string kosong (""), string kosong itu termasuk nilai falsy, sehingga kode yang di dalam blok `else` akan tereksekusi



2. Berdasarkan perkondisian pada kode di samping, apa yang akan terjadi?

- A. Akan kena error undefined
- B. Console.log akan muncul "Selamat"
- C. Console.log akan muncul "Cobalagi"



```
const apakahMemenuhiSyaratUmur = 20 > 17;
const namaPeserta = "Andini";
const tidakMerokok = false;

if(namaPeserta && apakahMemenuhiSyaratUmur && !tidakMerokok){
    console.log('Selamat')
} else{
    console.log('Cobalagi')
}
```



2. Berdasarkan perkondisian pada kode di samping, apa yang akan terjadi?

- A. Akan kena error undefined
- B. Console.log akan muncul "Selamat"
- C. Console.log akan muncul "Cobalagi"



```
const apakahMemenuhiSyaratUmur = 20 > 17;
const namaPeserta = "Andini";
const tidakMerokok = false;

if(namaPeserta && apakahMemenuhiSyaratUmur && !tidakMerokok){
    console.log('Selamat')
} else{
    console.log('Cobalagi')
}
```

Variabel `namaPeserta` dan `apakah MemenuhiSyaratUmur` besifat truthy, `!tidakMerokok` juga bernilai true, sehingga semua yang variabel yang berada di dalam kondisi if bersifat truthy, maka kode di dalam blok if yang akan dieksekusi.



```
const apakahMemenuhiSyaratUmur = 20 > 17;
const namaPeserta = "Andini";
const tidakMerokok = false;
const apakahJatuhCinta = true

if(namaPeserta && apakahMemenuhiSyaratUmur && !tidakMerokok && apakahJatuhCinta){
    console.log('Cieeee')
} else{
    console.log('Cobalagi')
}
```

3. Berdasarkan perkondisian pada kode di atas, apa yang akan terjadi?

- A. Akan kena error null is not nullable
- B. Akan kena error variabel undefined
- C. Akan kena error Invalid data type



```
const apakahMemenuhiSyaratUmur = 20 > 17;
const namaPeserta = "Andini";
const tidakMerokok = false;
const apakahJatuhCinta = true

if(namaPeserta && apakahMemenuhiSyartUmur && !tidakMerokok && apakahJatuhCinta){
    console.log('Cieeee')
} else{
    console.log('Cobalagi')
}
```

3. Berdasarkan perkondisian pada kode di atas, apa yang akan terjadi?

- A. Akan kena error null is not nullable
- B. Akan kena error variabel undefined
- C. Akan kena error Invalid data type

Karena variabel apakahMemenuhiSyartUmur tidak ada.



Nah, sampai sini dulu pembahasan kita kali ini ya. Chapter 1 Topic 3 masih akan berlanjut dengan topik yang nggak kalah seru dan bermanfaat lainnya.

See you!



Terima Kasih!



Next Topic

loading...