



Algoritma Dasar Javascript

Silver - Chapter 2 - Topic 3

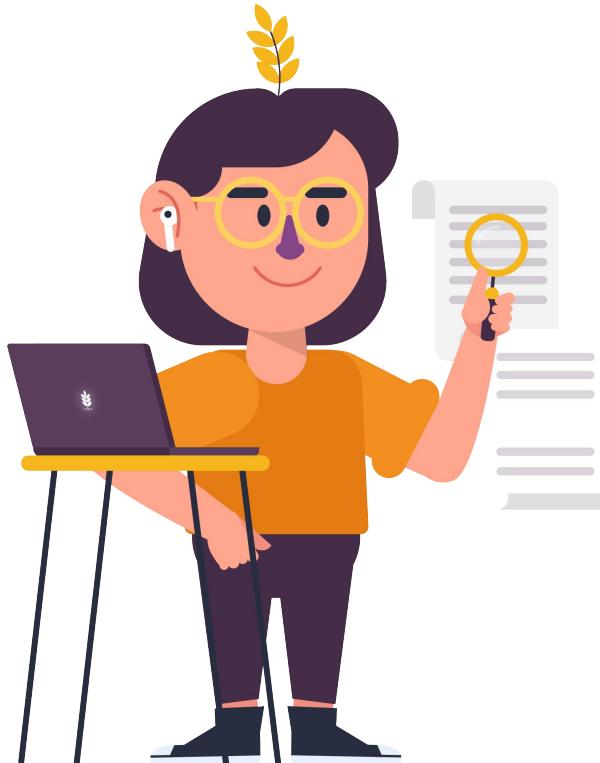
**Selamat datang di Chapter 2 Topik 3 online
course Full Stack Web dari Binar Academy!**





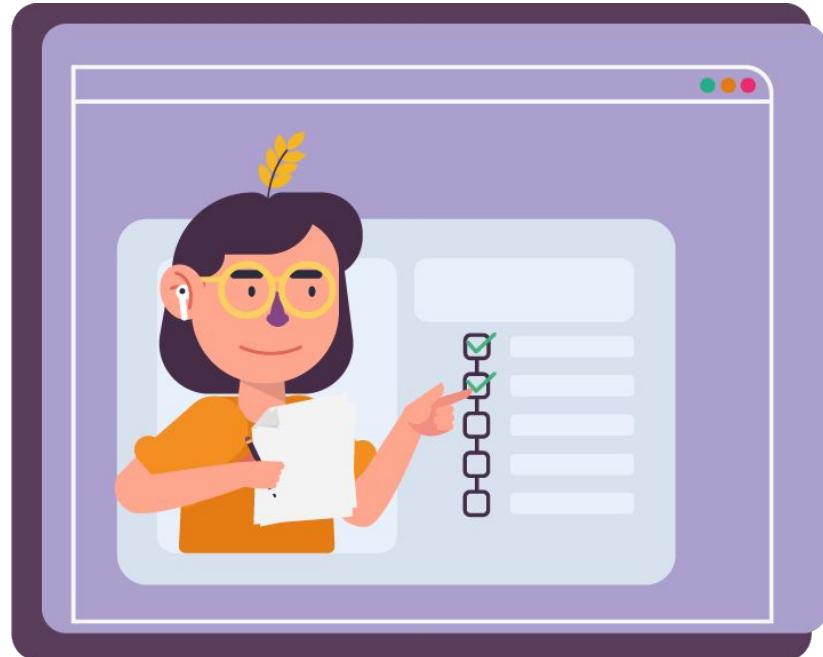
Pada topik kedua, kita udah mengelaborasi tentang penggunaan operator dan pembuatan expression pada JavaScript.

Pada topik ketiga ini, kita bakal bahas tentang **dasar-dasar algoritma pada JavaScript**. Cus langsung aja~



Detailnya, kita bakal bahas hal-hal berikut ini:

- Mengenal algoritma
- Alur berpikir algoritma
- Alur perulangan (Looping)





Kalau mau bikin web kita gak bisa tiba-tiba ngoding tanpa perencanaan lho. Kita perlu melakukan perencanaan untuk menggambarkan proses dari pengolahan data dari aplikasi kita ~

Nah alur kerja di dalam aplikasi itu biasa disebut sebagai **algoritma**. Penasaran? Yuk langsung cus ~



Kita mulai dari Algoritma ya!

Kamu pasti pernah melakukan top up pulsa kan?

Kayak pas kamu lagi ngelakuin langkah-langkah buat top up pulsa, itu berarti kamu lagi menerapkan algoritma lho.

Nah, Algoritma adalah suatu **metode step by step untuk menyelesaikan suatu masalah**. Ketika kita berpikir untuk menyelesaikan masalah selangkah demi selangkah.



Coba deh bayangin pas kamu lagi top up pulsa pake voucher gosok~

Pas kamu beli voucher, kamu harus ngegosok bagian kartu biar kode voucher bisa keliatan.

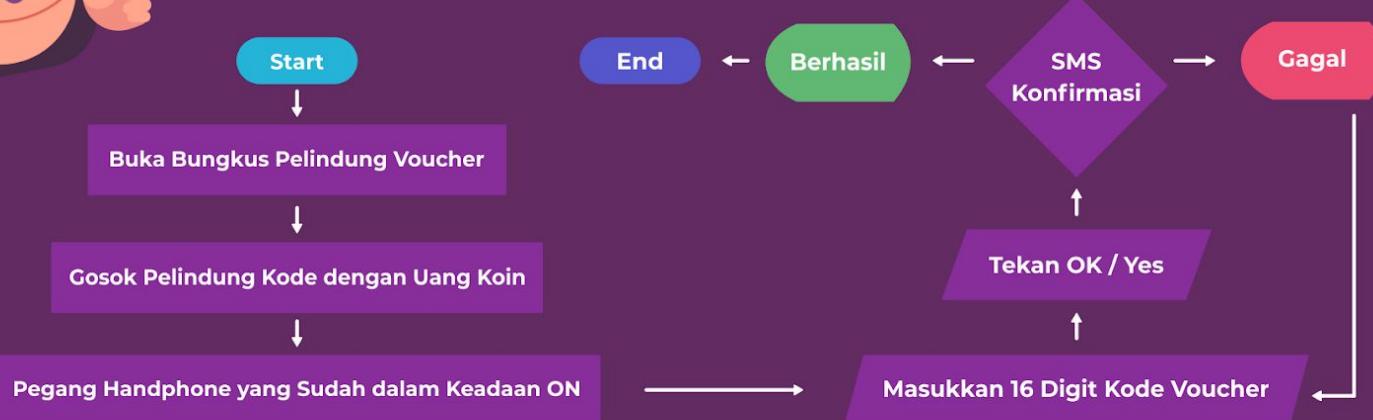
Kalo udah keliatan, kamu harus input kode itu ke dalam hp kamu.

Kalo input kodenya bener, proses top-up pulsa bakal berhasil. Tapi, kalo kodenya salah, pulsa yang kita beli juga nggak bakal masuk. Jadinya malah rugi, kan? 😞





Nah, proses top up pulsa pake voucher tadi, algoritmanya bisa digambarkan kayak di bawah ini!



Contoh penggambaran algoritma voucher tadi itu pake cara flowchart. Selain **flowchart**, ada juga cara **pseudocode**.

Gimana sih maksudnya? Berangkat~



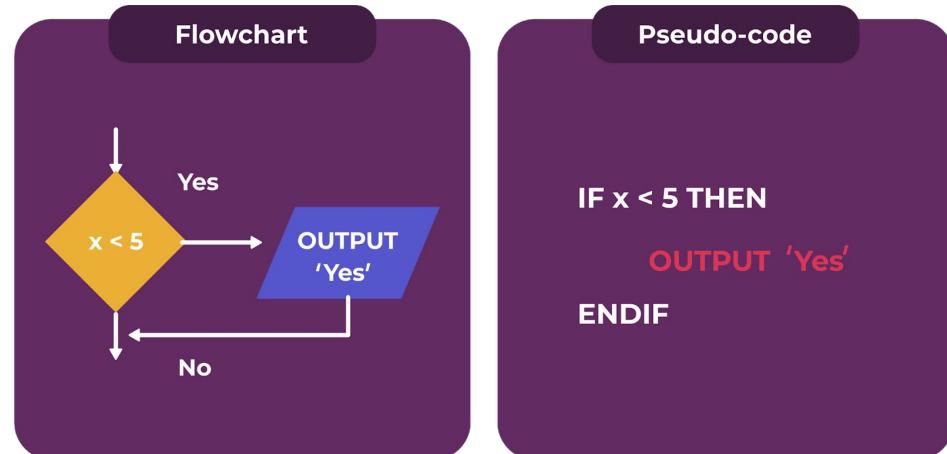
Cara penggambaran ini disebut juga alur berpikir algoritma.

Ada **flowchart** dan **pseudocode**.

Flowchart adalah alur berpikir yang direpresentasikan lewat diagram alir dengan berbagai simbol.

Kalo **pseudocode**, alur berpikir yang direpresentasikan lewat teks atau kata-kata.

Kalo kamu perhatikan gambar di samping, keduanya merepresentasikan hal yang sama, tapi cara merepresentasikannya berbeda.



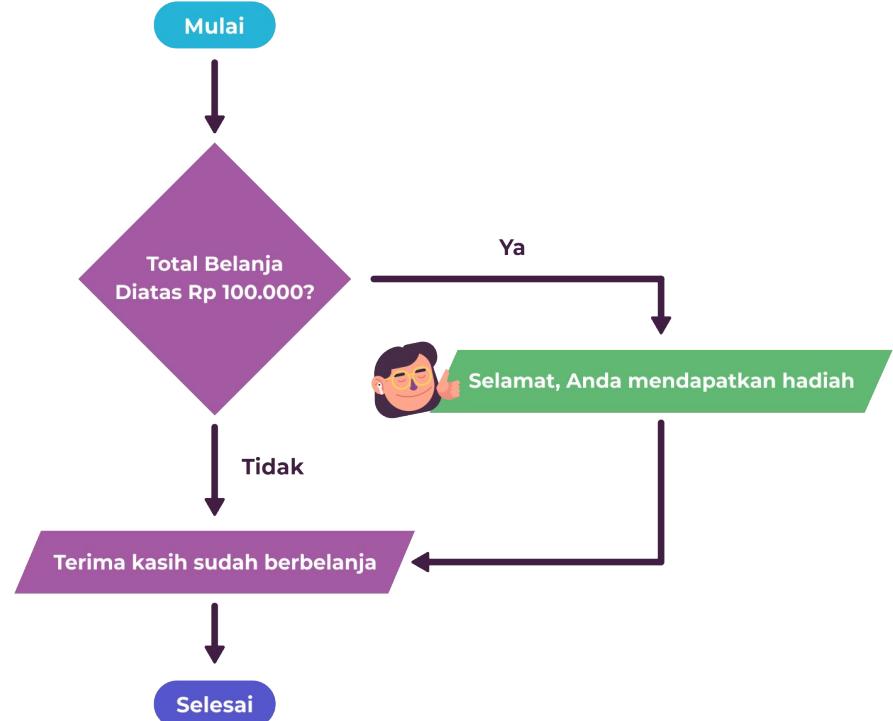


Bagaimana alur berpikir Flowchart?

Buat memahami lebih lanjut tentang flowchart, coba perhatikan gambar di samping ya!

Gambar di samping adalah contoh flowchart sederhana. Flowchart tersebut memvisualisasikan sebuah alur buat memahami suatu hal dengan menggunakan diagram. Misalnya, "Apakah setelah berbelanja kita berhak untuk mendapatkan hadiah?".

Dalam kasus ini, kita akan mendapatkan hadiah jika total belanja lebih dari 100 ribu rupiah.



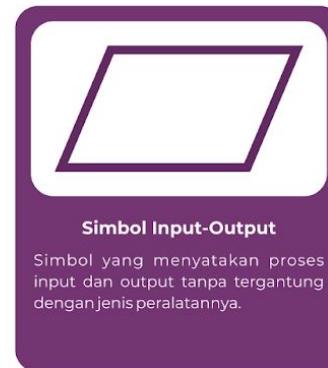
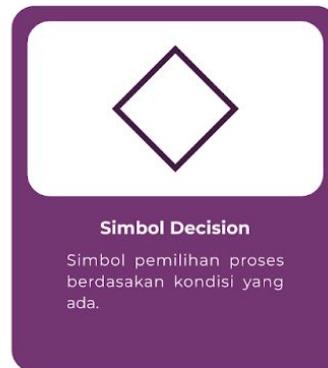


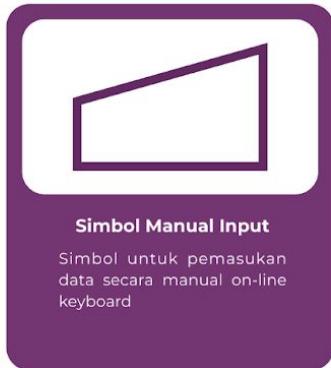
Eh, coba deh kamu perhatiin lagi flowchart tadi? Simbolnya beda-beda, kan? Ada yang persegi belah ketupat, jajar genjang, dan macem-macem.

Nah, **dalam flowchart, terdapat berbagai simbol yang memiliki fungsi berbeda.**

Cek halaman setelah ini ya untuk melihat berbagai simbol dalam flowchart.







Flowchart ini punya kelebihan dan kekurangan.

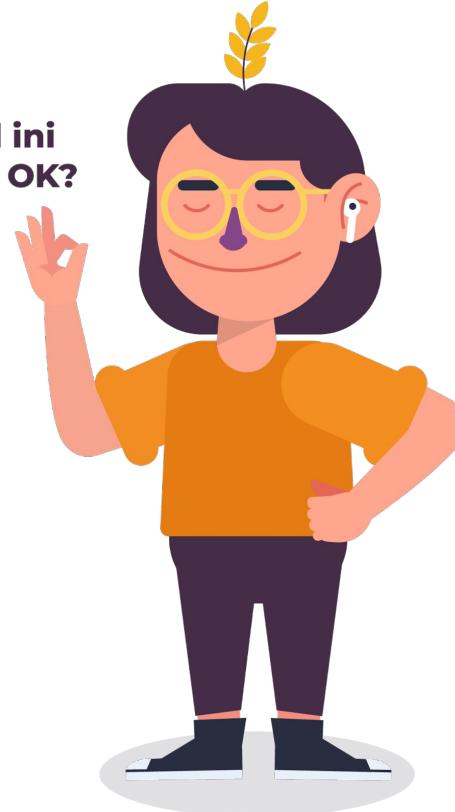
Kelebihannya:

1. Flowchart mampu menyajikan control flow algoritma secara visual. Jadi, kita lebih gampang buat ngerti alur penyelesaian suatu masalah atau kondisi.
2. Lebih gampang mendeteksi kesalahan atau ketidakakuratan suatu flowchart yang kompleks dan detail.

Kekurangannya:

1. Pengeraannya memakan waktu alias lama banget. Prosesnya nih, kita harus memposisikan, ngasih label, dan ngehubungin simbol dalam flowchart.
2. Kalo pake tools khusus buat bikin flowchart, itu malah berpotensi buat menghambat pemahaman kita tentang algoritma.

Ingat hal ini baik-baik. OK?



Setelah flowchart, lanjut pseudocode. Apa sih pseudocode itu? 🤔

Kalo tadi flowchart itu pake ilustrasi diagram, jadi visual-visual gitu. Nah kalo pake pseudocode, kita bakal **gambarin algoritma itu dengan kata-kata atau teks**.

Bukan ngata-ngatain loh yaa ~

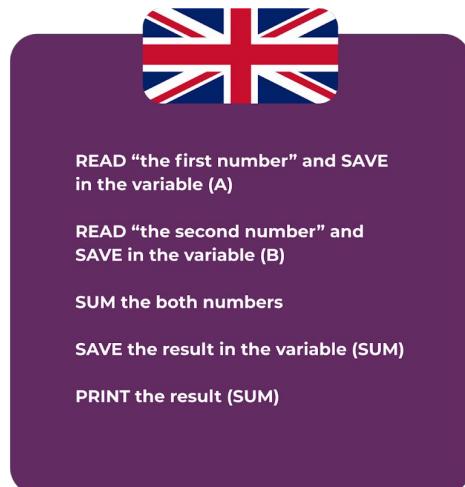
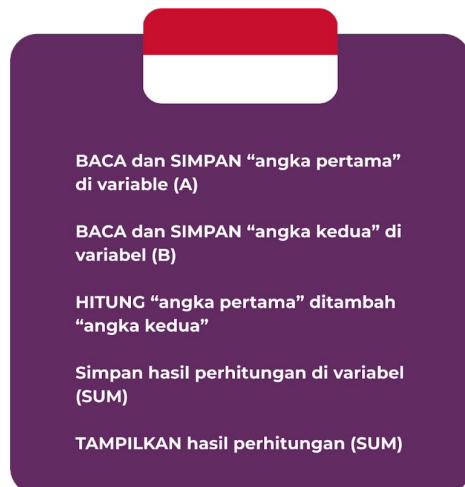


```
x = 10;  
y = 200;  
z = 0;  
  
IF x >= 10 THEN  
    OUTPUT "IYA BRO, x tuh lebih dari sama dengan 10"  
ENDIF
```



Untuk pseudocode ini, sebenarnya nggak ada aturan yang tegas tentang cara menulisnya. Karena, syntax nya fleksibel.

Buat gambaran biar lebih jelas, coba cek komparasi yang ada di gambar ya!





Pas udah ngebuat pseudocode kayak contoh tadi, selanjutnya bisa dikonversikan dalam bahasa pemrograman kayak gambar di samping ini!



```
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)
```



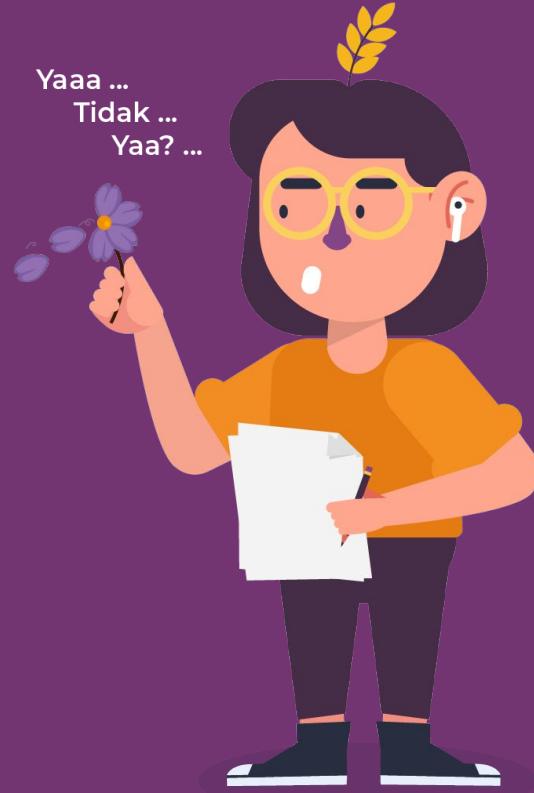
Pseudocode juga punya kelebihan, Iho!

Kalau masalah yang mau diberesin itu kompleks, pseudocode bisa dibuat terlebih dahulu biar nanti gampang pas proses ngoding. Mantul, kan?



Setelah kita tau gimana cara gambarin pola penyelesaian masalah pakai 2 pendekatan tadi. Sekarang kita perlu tau nih, dalam sebuah proses penyelesaian masalah itu pasti akan ada decision yang perlu diambil kan? Dan itu pasti masuk jadi salah satu gambaran di algoritma kita.

Nahhh ~ kalau di Javascript decision making ini kita sering sebut **Conditional Statement**.



Sab! Coba jelaskan secara singkat dong conditional statement itu gimana maksudnya?

Jadi gini, conditional statement ini dipake buat mendefinisikan decision yang akan dilakuin di dalam aplikasi kita.

Contoh ya, kita ingin bikin program yang bisa nyapa user kalo user lagi online. Nah untuk buat program kaya gitu, kita butuh yang namanya **conditional statement**.

Conditional statement di Javascript itu bisa dibikin dengan 2 cara, yaitu :

1. if else statement.
2. switch statement.





```
const gebetan = {  
    name: "Joko",  
    isOnline: true  
}  
  
if (gebetan.isOnline) {  
    console.log(`Halo, ${gebetan.name}!`);  
    console.log("Lagi ngapain nih! Sleepcall yuk!");  
}  
  
/**  
 * Output:  
 * Halo, Joko!  
 * Lagi ngapain nih! Sleepcall yuk!  
 */
```

If Else Statement

If else statement itu adalah sintaks yang nentuin apakah kode di dalam statement tersebut harus dijalankan atau enggak.

Dari contoh kode disamping, blok kode yang ada di dalam kurung kurawal setelah **if** akan dieksekusi, karena nilai dari **gebetan.isOnline** bernilai **true**.

Jadi if itu mirip lah kayak unary operator, yang mana dia butuh operan yang bernilai **boolean**. Kalau nilainya **truthy**, maka blok kode yang ada di dalamnya akan dijalankan.



Nah selain if ada juga else, yang mana kalo kamu pake else wajib ada statemen if sebelumnya. Nah **else ini adalah sebuah blok kode yang mana isi dari blok tersebut akan dieksekusi kalo nilai dari operan if bernilai falsy.**

Dari sini, kita bisa simpulkan kalo if else statement ini dipake buat ngebikin dua decision. Nah kalo kamu butuh 1 kasus lagi dan ketika kasus itu muncul kamu bakal mengeksekusi hal yang berbeda, kamu bisa pake yang namanya else if.

Detail lebih lanjutnya bisa elaborasi lagi [disini](#).

```
const gebetan = {
  name: "Joko",
  isOnline: true,
  blocked: true
}

if (gebetan.isOnline && !gebetan.blocked) {
  console.log(`Halo, ${gebetan.name}!`);
  console.log("Lagi ngapain nih! Sleepcall yuk!");
} else {
  console.log("Sebenarnya ada yang pengen aku sampein ke kamu.");
  console.log("Tapi sayangnya aku diblock sama kamunya");
};

/**
 * Output:
 * Sebenarnya ada yang pengen aku sampein ke kamu.!
 * Tapi sayangnya aku diblock sama kamunya
 */
```



Oke, udah tau ya if else statement apa. Terus satunya lagi gimana? Yang switch statement ~

Nah selain if else statement. Kita bisa buat conditional statement pakai switch statement.

Switch statement ini mirip kayak if else statement, bedanya si **switch statement akan minta satu nilai, yang nantinya dia bakal cek nilai itu dalam berbagai kondisi**, ga cuma true atau false aja.

Dari contoh disamping, kita hanya masukan **“day” sebagai parameter di switch statement**. Nanti **switch statement akan mendefinisikan kasus kalau “day” ini nilainya sama dengan kasus yang kita tulis**. Jadi kalau sama, dia bakal eksekusi kode yang ada di dalam kasus tersebut.

```
const day = "MONDAY";

switch(day): {
  case "MONDAY":
    console.log("Hari senin? Gue sih santai aja!");
    break;
  case "TUESDAY":
    console.log("Hari selasa gini gatau sih mau ngapain!");
    break;
  case "WEDNESDAY":
    console.log("Rabu manja!");
    break;
  case "THURSDAY":
    console.log("Kamis mantab!");
    break;
  case "FRIDAY":
    console.log("Thanks God It's Friday!");
    break;
  case "SATURDAY":
    console.log("Healing dulu, bos!");
    break;
  case "SUNDAY":
    console.log("Besok senin? Gue sih santai aja!");
    break;
  default:
    console.log("Lu masukin hari apa bro? Hari kiamat?")
}

/**
 * Output:
 * Hari senin? Gue sih santai aja!
 **/
```



Pastinya kamu udah tahu kan apa itu algoritma dan gimana alur berpikirnya?

Sekarang, kita bakal bahas soal alur perulangan (**looping**) pada algoritma.

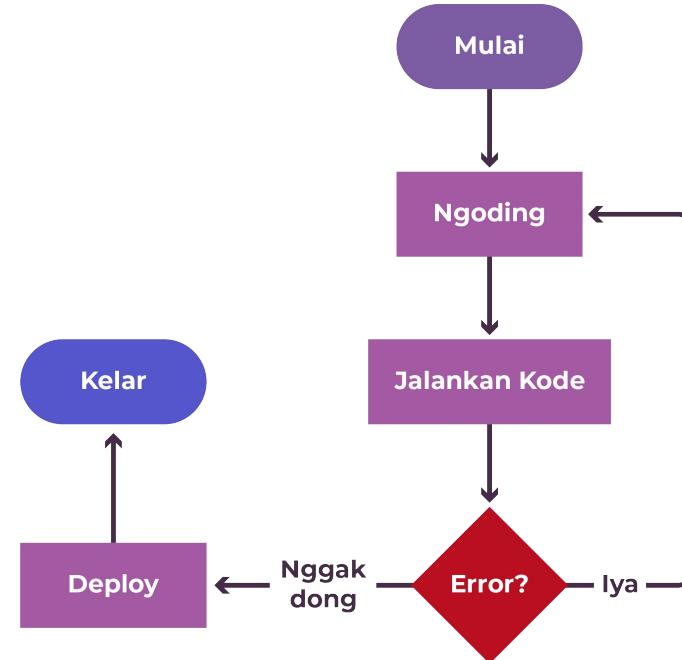
Letsgoo~



Cek gambar disamping dulu, yuk!

Kalau kita lihat flowchart di samping di bagian **Error?** Terus andaikan hasil dari pertanyaan tersebut jawabannya **Iya**, maka dia akan mengulangi step **ngoding** lagi. Bener ga?

Nah hal kayak gini tuh biasa disebut dengan loop atau iterasi.





Berikut tiga jenis alur perulangan (looping) yang perlu kamu tahu!

- Perulangan “for”
- Perulangan “while”
- Perulangan “do...while”

Apaan sih maksudnya? Kuy kita bahas satu-persatu!





For Loop

Pada alur ini, **perulangan akan terus dilakukan sampai kondisi tertentu tercapai**. Saat kondisi sudah tercapai, maka perulangan akan dihentikan.

Alur perulangan "**for**" dapat digambarkan dalam kegiatan sehari-hari, misalnya ketika petugas kebersihan mengepel lantai di sebuah gedung.

Dia diberikan tugas buat ngepel dari lantai 1 sampe lantai 5, dia bakal mengulangi kegiatan dia di lantai 1 (ngepel, meres pel, ngepel lagi) di tiap lantai sampai lantai ke 5.

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

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



```
const initialFloorLevel = 1;

for (
  let floorLevel = initialFloorLevel;
  floorLevel <= 5;
  floorLevel++
) {
  console.log("Aku lagi di lantai ", floorLevel);
  console.log("Ngepel!");
  console.log("Meres pel!");
  console.log("Ngepel lagi!");
}
```

Kode disamping itu adalah contoh implementasi dari **for loop**.

Kalau kamu coba jalankan kode disamping, maka tiap log yang ada di dalam blok for loop itu akan dieksekusi sejumlah 5 kali.

Kenapa kok 5 kali? Karena kita memberi conditional statement **floorLevel <= 5**, inilah yang menentukan kapan loop tersebut harus berhenti, yang mana jika **floorLevel** bernilai **5**, maka loop tersebut akan berhenti.

Kalau kita tidak berhati-hati dalam menentukan conditional statement untuk membuat loop tersebut berhenti, maka hal terburuk yang akan terjadi adalah terjadinya **infinite loop** (perulangan tak berujung).



While Loop

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

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

Petugas ingin memeriksa adanya noda pada lantai. Jadi, selama terdapat noda pada lantai, maka ia akan mengulangi tahap kedua sampai kelima secara terus-menerus.

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

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



While Loop



```
const maxFloorLevel = 5;
let floorLevel = 1;

function getRandomInt(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(Math.random() * (max - min + 1)) + min;
}

while (floorLevel <= maxFloorLevel) {
    let isSpotless = false;
    console.log("Ngepel lantai", floorLevel);

    // Simulasi peluang lantai
    // spotless enggak setelah dipel
    let godsWill = getRandomInt(0, 1);

    // 0 berarti lantainya ga
    // spotless setelah dipel
    if (godswill === 0) isSpotless = true;

    // Kalau lantainya udah gaada noda,
    // bisa lanjut ke lantai berikutnya
    if (isSpotless) floorLevel++;
}
```

Kode disamping adalah contoh while loop yang mana blok yang ada di dalam while loop tersebut akan selalu dijalankan selama **isSpotless** bernilai **false**.

Nah, di dalam **blok while loop** ada implementasi untuk generate angka antara 1 dan 0 sebagai penanda apakah lantai itu masih bernoda atau tidak setelah dipel. Nah, kalau lantainya masih bernoda, maka tukang ngepel itu akan mengepel lantai tersebut sampai nodanya hilang.



Perulangan “do...while”

Alur perulangan "do...while" mirip sama "while". Bedanya, **"do...while" bakal langsung ngelakuin loop, terus baru ngecek kondisi.** Sedangkan **"while", harus ngecek kondisi dulu di awal.**

Kita coba ubah contoh kondisi alur perulangan "while" menjadi kondisi alur perulangan dengan "do while" ya ~

Jadi, petugas bakal ngelakuin tahap pertama mulai dari mengambil kain pel, mencuci kain pel, sampai tahap keempat yaitu mengepel lantai secara berurut.

Setelah itu, ia akan memeriksa kondisi kain pel: "Udah kotor atau belum ya?". Kalo kondisi kain pel nya kotor, maka ia bakal ngulang tahap ke-2 sampai ke-5.

Tapi, sebenarnya jenis looping yang satu ini akan jarang banget kita temuin lho~



Dari algoritma tadi, bisa jadi kita menemukan case algoritma yang sama, namun inputnya berbeda.

Agar kita gak mengulangi code yang berulang-ulang kita bisa **simpan algoritma tadi di dalam function**





Ceritanya Sabrina lagi latihan if statement nih, buat ngecek apakah suatu angka itu ganjil atau enggak. Setelah ditulis codingnya, jadinya kayak contoh disamping. Repetitif banget. Kalo misal angkanya 100 gimana coba?

Sebenarnya, kalau case kayak gitu, kamu bisa pake yang namanya function. Nah **function ini bisa dipake buat menyimpan algoritma yang sama, yang nantinya bakal dijalankan lagi dengan input yang berbeda.**

Jadi kalo inputnya ada 100 kita cukup panggil function itu lagi untuk ngejalankan algoritma yang sama tanpa harus menulis algoritmanya berkali-kali.



```
const x = 1;
const y = 2;
const z = 3;

if (x % 2 !== 0) {
  console.log(x, "itu ganjil!");
}

if (y % 2 !== 0) {
  console.log(y, "itu ganjil!");
}

if (z % 2 !== 0) {
  console.log(z, "itu ganjil!");
}

/**
 * Output:
 * 1 itu ganjil
 * 3 itu ganjil
 */
```



```
● ● ●  
const x = 1;  
const y = 2;  
const z = 3;  
  
function sayIfOdd(number) {  
    if (number % 2 !== 0) {  
        console.log(number, "itu ganjil");  
    }  
}  
  
sayIfOdd(x); // Output: 1 itu ganjil  
sayIfOdd(y);  
sayIfOdd(z); // Output: 3 itu ganjil
```

Nah kalo dilihat dari contoh kode disamping, kita cuma perlu menulis algoritma untuk mengecek bahwa suatu angka itu ganjil atau enggak. Setelah itu kita tinggal panggil fungsi **sayIfOdd** dengan input yang berbeda.

Di dalam fungsi tersebut **number inilah yang kita sebut sebagai parameter atau argumen**, yang mana sifatnya mirip variabel yang nanti kita panggil menggunakan namanya dibanding menggunakan nilainya secara langsung.

Menulis algoritma di dalam sebuah fungsi itu sangat direkomendasikan meskipun kita hanya memanggilnya sekali.

Karena ini akan membuat aplikasimu jauh lebih mudah dibaca dan scalable.



Cara untuk buat function itu gampang lho. Biasanya kita pake keyword **function**. Dan ada 3 cara yang bisa kita pake untuk bikin function , yaitu :

- Function keyword
- Function keyword tapi anonim
- Arrow function





```
● ● ●  
// Fungsi untuk menghitung luas persegi  
// dimana meminta size sebagai parameter  
function computeSquareArea(size) {  
    // Ketika kita ingin menggunakan hasil dari  
    // fungsi ini, kita perlu menggunakan keyword  
    // return, karena return membuat nilai yang ada di fungsi  
    // ini dapat diakses secara global  
    return size * size;  
  
const squareAreaOf4 = computeSquareArea(4);  
console.log(squareAreaOf4); // Output: 16
```

Function Keyword

Sama kayak contoh sebelumnya, untuk mendefinisikan sebuah function dengan keyword aja, **kita hanya perlu menambahkan keyword function dan diikuti oleh nama function-nya.**



Function keyword tapi anonim

Mirip kayak bikin function dengan keyword aja, bedanya kalo di cara ini, kita **ga perlu ngasih nama function setelah keyword function.**

Nah nama function tersebut ditentukan oleh nama variabel yang menyimpannya.

```
// Fungsi untuk menghitung luas persegi
// dimana meminta size sebagai parameter
const computeSquareArea = function(size) {
    // Ketika kita ingin menggunakan hasil dari
    // fungsi ini, kita perlu menggunakan keyword
    // return, karena return membuat nilai yang ada di fungsi
    // ini dapat diakses secara global
    return size * size;
}

const squareAreaOf4 = computeSquareArea(4);
console.log(squareAreaOf4); // Output: 16
```



```
● ● ●  
  
// Fungsi untuk menghitung luas persegi  
// dimana meminta size sebagai parameter  
const computeSquareArea = function(size) {  
    // Ketika kita ingin menggunakan hasil dari  
    // fungsi ini, kita perlu menggunakan keyword  
    // return, karena return membuat nilai yang ada di fungsi  
    // ini dapat diakses secara global  
    return size * size;  
}  
  
const squareAreaOf4 = computeSquareArea(4);  
console.log(squareAreaOf4); // Output: 16
```

Arrow Function

Nah untuk arrow function, cara mendefinisikannya mirip dengan function pake keyword tapi anonim. Bedanya, disini kita **ga perlu pake keyword function sama sekali, kita hanya perlu notasi panah (=>) aja buat mendefinisikan function.**

Saatnya kita Quiz!





- 1. Sabrina ingin menggambarkan alur dari aplikasinya secara visual. Sebagai sahabat Sabrina, kira-kira kamu akan merekomendasikan bentuk algoritma apa?**
 - A. Flowchart
 - B. Pseudocode
 - C. Diagram Venn

- 1. Sabrina ingin menggambarkan alur dari aplikasinya secara visual. Sebagai sahabat Sabrina, kira-kira kamu akan merekomendasikan bentuk algoritma apa?**
 - A. Flowchart
 - B. Pseudocode
 - C. Diagram Venn

Untuk ngegambarin sebuah alur atau algoritma secara visual, Sabrina bisa pake flowchart, karena di dalam flowchart ada simbol-simbol yang merepresentasikan suatu alur.



2. Terdapat sebuah variabel bernama **isRaining** yang bernilai **false**. Jika variabel tersebut digunakan pada kode dibawah, apa output yang keluar di dalam log nanti?

- A. Aku bakal pake payung!
- B. Gaada output sama sekali
- C. false



```
if (isRaining) { console.log("Aku bakal pake payung!") }
```



2. Terdapat sebuah variabel bernama **isRaining** yang bernilai **false**. Jika variabel tersebut digunakan pada kode dibawah, apa output yang keluar di dalam log nanti?

- A. Aku bakal pake payung!
- B. **Gaada output sama sekali**
- C. false



```
if (isRaining) { console.log("Aku bakal pake payung!") }
```

Karena **isRaining** bernilai **false**, maka dari itu blok kode yang ada di dalam **if statement** tersebut tidak akan dijalankan, makanya gaada output.

3. Jika ditulis loop dibawah ini, maka output apa yang akan dihasilkan?

- A. **Halo** sebanyak 10 kali
- B. **Halo** sebanyak 9 kali
- C. Ga ada output sama sekali



```
for (let i = 1; i < 10; i++) { console.log("Halo!") }
```

3. Jika ditulis loop dibawah ini, maka output apa yang akan dihasilkan?

- A. **Halo** sebanyak 10 kali
- B. **Halo** sebanyak 9 kali
- C. Ga ada output sama sekali



```
for (let i = 1; i < 10; i++) { console.log("Halo!") }
```

Karena nilai awal i adalah 1, dan kalau loop diatas akan berhenti apabila i sudah tidak kurang dari 10, maka dari itu loop tersebut akan dieksekusi sebanyak 9 kali



4. Manakah pernyataan yang benar terkait fungsi berikut:

- A. Fungsi tersebut meminta 1 parameter yang bernama size, dan hasil dari fungsi tersebut adalah number.
- B. Fungsi tersebut meminta 1 parameter yang bertipe number, dan hasil dari fungsi tersebut adalah **null**.
- C. Hasil dari fungsi tersebut adalah "Square Area: 100" apabila kita memanggil fungsi tersebut dengan 10 sebagai parameter.



```
function getSquareArea(size) { return size * size }
```



4. Manakah pernyataan yang benar terkait fungsi berikut

- A. Fungsi tersebut meminta 1 parameter yang bernama size, dan hasil dari fungsi tersebut adalah number.
- B. Fungsi tersebut meminta 1 parameter yang bertipe number, dan hasil dari fungsi tersebut adalah null.
- C. Hasil dari fungsi tersebut adalah "Square Area: 100" apabila kita memanggil fungsi tersebut dengan 10 sebagai parameter.



```
function getSquareArea(size) { return size * size }
```

Karena di dalam fungsi tersebut ada keyword return, maka hasil dari fungsi tersebut adalah apa yang ditulis setelah keyword return, yang mana hasil dari operasi size dikali dengan size.



5. Apa yang terjadi jika kode disamping kita jalankan?

- A. Muncul kata **Muter** sebanyak 100 kali di dalam console.
- B. Terjadi infinite loop, artinya kata Muter muncul terus di console sampe aplikasi tersebut ditutup.
- C. Ga muncul apa-apa di console.



```
let i = 1;

while (false) {
    i++;
    console.log("Muter");

    if (i === 100) break;
}
```



5. Apa yang terjadi jika kode disamping kita jalankan?

- A. Muncul kata **Muter** sebanyak 100 kali di dalam console.
- B. Terjadi infinite loop, artinya kata Muter muncul terus di console sampe aplikasi tersebut ditutup.
- C. Ga muncul apa-apa di console.

Blok yang ada di dalam while loop itu ga bakal dijalankan, karena kondisi loopingnya selalu false.



```
let i = 1;

while (false) {
    i++;
    console.log("Muter");

    if (i === 100) break;
}
```



6. Perhatikan switch statement dari kode disamping! Apa output dari kode disamping?

- A. Ini bulan Januari!
- B. Ini bulan Februari!
- C. OK!



```
const month = "JULY"
const JANUARY = "JANUARY";
const FEBRUARY = "FEBRUARY";

switch(month) {
    case JANUARY:
        console.log("Ini bulan Januari!");
        break;
    case FEBRUARY:
        console.log("Ini bulan Februari!");
        break;
    default:
        console.log("OK!")
}
```



6. Perhatikan switch statement dari kode disamping! Apa output dari kode disamping?

- A. Ini bulan Januari!
- B. Ini bulan Februari!
- C. OK!

Karena nilai "JULY" tidak didefinisikan di dalam salah satu case di dalam switch statement disamping, maka kode yang ada di dalam kasus default akan dijalankan.



```
const month = "JULY"
const JANUARY = "JANUARY";
const FEBRUARY = "FEBRUARY";

switch(month) {
    case JANUARY:
        console.log("Ini bulan Januari!");
        break;
    case FEBRUARY:
        console.log("Ini bulan Februari!");
        break;
    default:
        console.log("OK!")
}
```



7. Perhatikan function dari kode disamping! Apa output dari kode disamping?

- A. January
- B. null
- C. July

```
const person = {  
    name: "Sabrina",  
    dateOfBirth: "30-07-1999"  
}  
  
function getBirthMonth(person) {  
    const month = person.dateOfBirth;  
  
    switch(month) {  
        case "01":  
            return "January";  
        case "07":  
            return "July";  
        default:  
            return null;  
    }  
}  
  
console.log(getBirthMonth(person));
```



7. Perhatikan function dari kode disamping! Apa output dari kode disamping?

- A. January
- B. null
- C. July

Karena nilai variabel month adalah 30-07-1999, yang mana tidak ada di dalam case di switch statement disamping, maka output dari fungsi tersebut adalah apa yang di-return di dalam default case.

```
const person = {  
    name: "Sabrina",  
    dateOfBirth: "30-07-1999"  
}  
  
function getBirthMonth(person) {  
    const month = person.dateOfBirth;  
  
    switch(month) {  
        case "01":  
            return "January";  
        case "07":  
            return "July";  
        default:  
            return null;  
    }  
}  
  
console.log(getBirthMonth(person));
```



8. Perhatikan function dari kode disamping! Apa output dari kode disamping?

- A. NaN
- B. 8
- C. null



```
function getSquareArea(size) {  
    size * size;  
}  
  
function getCubeVolume(size) {  
    return getSquareArea(size) * size;  
}  
  
const x = getCubeVolume(2);  
console.log(x);
```



8. Perhatikan function dari kode disamping! Apa output dari kode disamping?

- A. NaN
- B. 8
- C. null

Karena hasil dari getSquareArea adalah undefined, maka dari itu hasil perkalian antara hasil dari getSquareArea dan size adalah NaN.



```
function getSquareArea(size) {  
    size * size;  
}  
  
function getCubeVolume(size) {  
    return getSquareArea(size) * size;  
}  
  
const x = getCubeVolume(2);  
console.log(x);
```

Referensi

1. [MDN - Control Flow](#)
2. [Smartdraw - Flowchart](#)
3. [MDN - Loop and Iteration](#)
4. [MDN - Function](#)





Nah, sekarang kamu udah tau banyak tentang Javascript. Artinya, sudah selesai chapter ini.

Kalau kamu ingin semakin jago berbahasa dengan bahasa Javascript, jangan lupa banyak latihan ya! Bisa disini [Hackerrank - 10 days of Javascript](#)

Selanjutnya, kita akan lanjut masuk ke Chapter 3 ! See you ~



Terima Kasih!



Chapter ✓

completed