



**Fundamental JavaScript,
Struktur Data, Operator & Expression
Silver - Chapter 1 - Topic 2**

**Selamat datang kembali di Chapter 1 Topic 2
pada course React Native dari Binar Academy!**

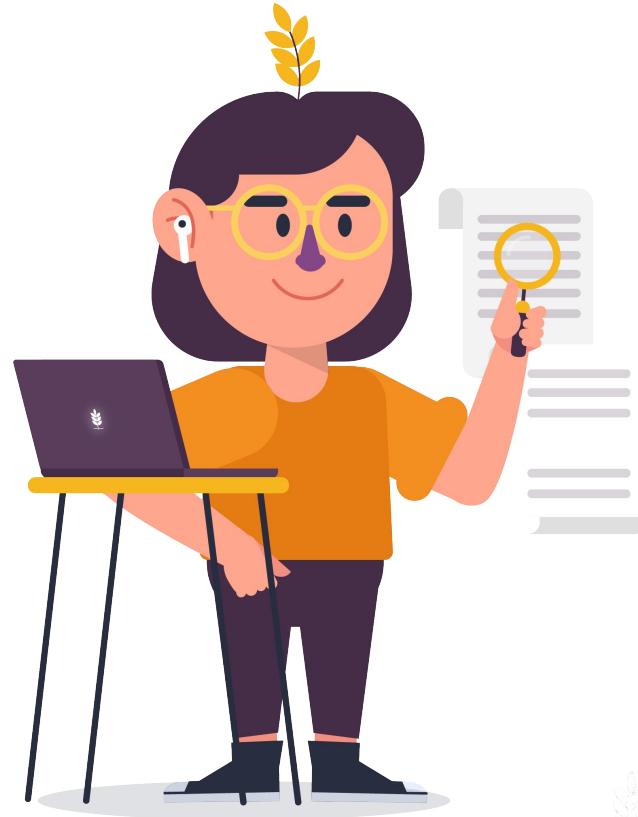


Haalloo Binarian! 🙌

Pada **Topic 1** kemarin, kita sudah belajar banyak nih seputar **React Native** beserta **software dan berbagai tools** yang terdapat di dalamnya.

Nah, sekarang kita akan belajar tentang **JavaScript** dan **hal-hal dasar yang mendukung JavaScript**.

Udah siap belum? Let's go! 🚀





Detailnya, kita bakal bahas hal-hal berikut ini:

- Javascript Dasar
- Struktur Kode
- Tipe Data
- Operator
- Logical Operator



Ngomong-ngomong soal JavaScript, hal pertama yang terlintas di pikiranmu apa?

Cangkir kopi? Game jadul? Atau justru nama Jawa (Java) itu sendiri?





Tahu nggak geng?

Javascript itu salah satu bahasa pemrograman komputer selain Python dan C++ lho! Nah, mereka ini sering banget nih **dipakai buat ngerjain web development.**

Jadi, jangan heran ya kalau pas kamu browsing di internet, kamu bakal sering ketemu sama JavaScript ini 😊





Karena **JavaScript itu bahasa pemrograman**, maka isinya masih berupa kode-kode khusus yang harus diterjemahkan ke bahasa kita.

Biar apa sih? Biar nyambung dong 😎





Terus, penerjemahnya siapa nih?

Tadi udah dijelasin kan kalau JavaScript ini bakal sering kamu temuin ketika browsing di internet.

Nah, tahu nggak sih kalau browser yang kita pakai itu punya penerjemah JavaScript?

Wah, beneran nih? 😱



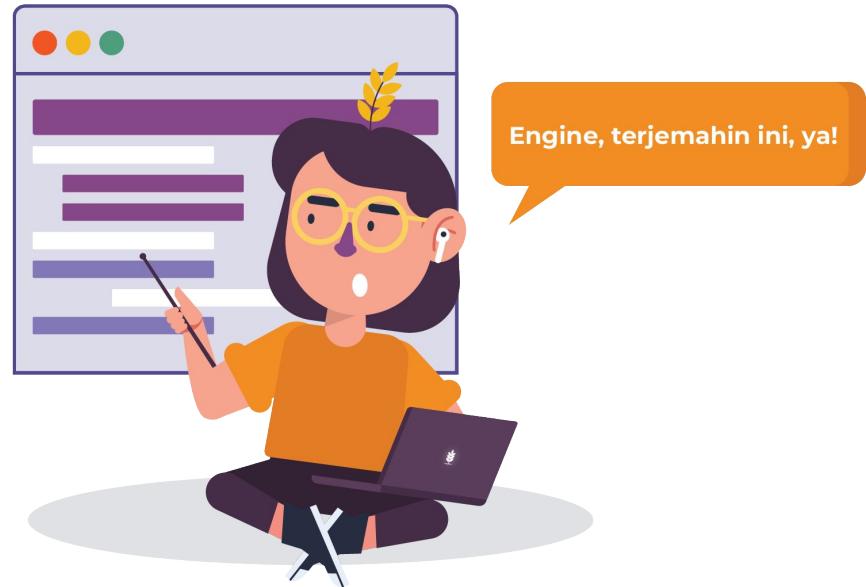


Beneran dong!

Penerjemahnya itu bernama **engine** 🙌🙌🙌

FYI, Engine atau **mesin pencari** ini bertugas untuk menerjemahkan bahasa **JavaScript** sekaligus menghubungkannya ke komputer kita nih sob!

Seperti jembatan, Engine **berfungsi untuk meneruskan perintah kita dan ngasih tau komputer apa yang harus dilakukan.**

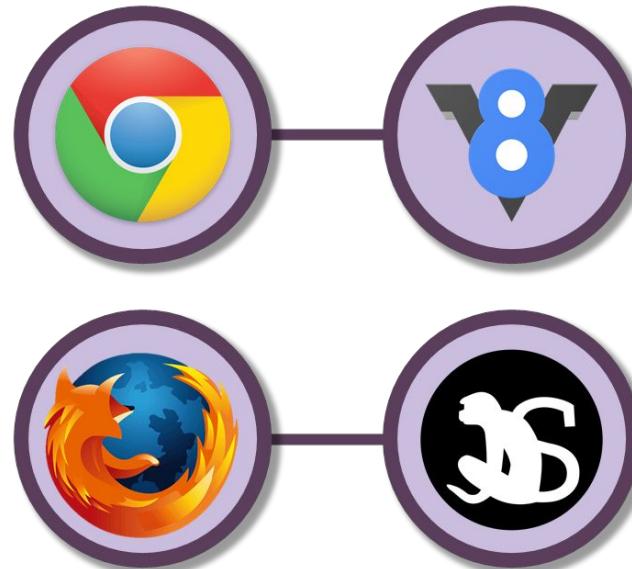




Engine di Chrome dan Firefox

Dari sekian jenis JavaScript Engine, ada dua engine atau mesin pencari yang sudah pasti kita kenal nih. Apa aja ya?

Dua engine itu adalah **V8 Engine** yang bisa kita temuin di browser **Chrome** dan **Spider Monkey** di **Firefox** 🔥





But, wait

Kalau aksesnya lewat browser, berarti harus pakai internet dong! 🤔

Terus kalau nggak ada internet, gimana nih? Masih bisa pakai JavaScript nggak ya? 😞

Internet mati, masih
bisa ngoding gak ya?





Kalem, Tenang, Rileks~

Meski nggak ada internet, kamu tetap bisa pakai JavaScript lho. Nggak perlu pakai browser, kamu justru bisa akses langsung nih di komputermu pakai **Node.JS**

Kok bisa?

Bisa banget! Node.JS ini bisa **dijalankan langsung di komputer** karena doi **punya pustaka server HTTP sendiri** nih sob~



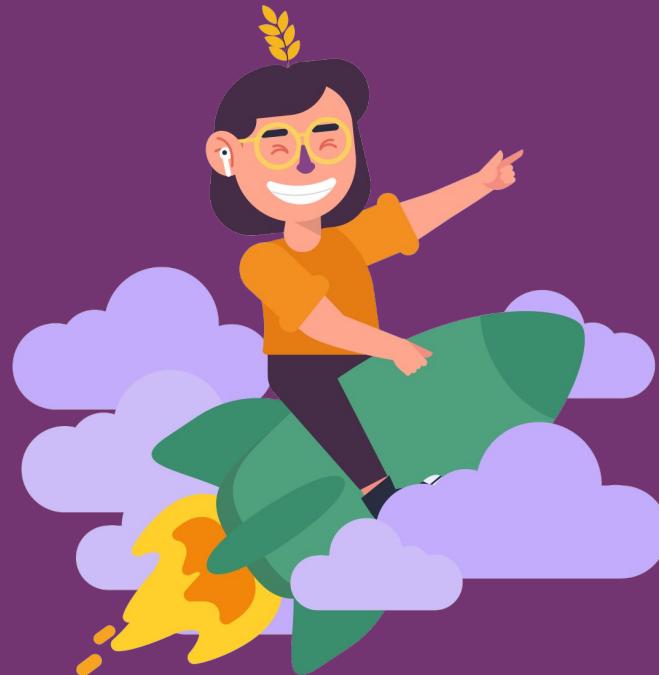


Selanjutnya, yuk kita bahas topik-topik yang berkaitan dengan Javascript berikut ini:

- Struktur Kode
- Variabel
- Tipe Data
- Operator
- Operator Logika (*Logical Operator*)

Setelah paham beberapa hal dasar terkait JavaScript, kita bakal berjalan lebih jauh nih gengs biar **makin paham seputar JavaScript!**

ayo Sob, kencangkan sabukmu dan belajarlah bersamaku~



Seperti bahasa pemrograman pada umumnya, JavaScript juga punya struktur kode tersendiri lho~

sebenarnya, struktur kode ini yang gimana sih?





JavaScript juga bahasa Iho~

Sama seperti bahasa yang kita pakai sehari-hari, JavaScript juga berupa bahasa yang punya struktur kalimat kayak bahasa kita Iho.

Kalau bahasa Indonesia struktur kalimatnya berupa subjek-predikat-objek, struktur dalam JavaScript berupa rangkaian berbagai kode.





Nah, struktur kode ini ada beberapa jenis nih Sob! Jenisnya itu ada 3 bentuk, yakni:

- **Statement** (Pernyataan)
- **Semicolon** (Titik Koma)
- **Comment** (Komentar/Kode yang dihiraukan) → Kita pernah bahas ini di chapter 1 HTML dan CSS

A

Statement

B

Semicolon

C

Comment



Statement (Pernyataan)

Sama kayak fungsi pernyataan pada pelajaran bahasa Indonesia, **pernyataan di JavaScript juga berfungsi untuk menyatakan suatu perintah** atau sekadar ngasih info ke komputer nih.

Nah, statement ini kayak petunjuk alur gitu sob! kalau petunjuknya nggak urut, eksekusinya juga bakal berantakan nih~



Semicolon (Titik Koma)

Semicolon ini berfungsi untuk memisahkan statement yang satu dengan yang lainnya nih sob! Tapi, penggunaannya nggak wajib ya~

Wah, kenapa nggak wajib ya? Kan sama-sama struktur kode di JavaScript? 😕





Biar tahu alasannya apa, coba yuk perhatiin gambar berikut.

```
» console.log("Pernyataan 1")
  console.log("Pernyataan 2")
```

```
← undefined
```

Pernyataan 1

Pernyataan 2

Contoh nggak pakai titik koma

```
» console.log("Pernyataan 1");
  console.log("Pernyataan 2");
```

```
← undefined
```

Pernyataan 1

Pernyataan 2

Contoh pakai titik koma



Di situ, kita nggak perlu nambahin titik dua kalau pengen bedain statement dengan memakai jeda garis baru atau yang biasa kita sebut dengan implisit~

Tapi nggak masalah juga sih kalau kamu pengen nambahin titik dua di belakang *statement*.

Jadi, semicolon ini opsional ya sob! 😊





Oh iya, perlu diketahui nih kalau **nggak selamanya garis baru itu dianggap statement baru** lho ya~

Kalau kamu bikin suatu ekspresi dan belum selesai di baris itu, maka JavaScript akan terus melihat garis di bawahnya sampai ekspresi itu selesai.





Contohnya kayak ini nih sob~

6 + 10 adalah sebuah ekspresi yang akan dipindai JavaScript sampai selesai.

Lalu, terkait dengan ekspresi, kita **nggak boleh meletakkan titik koma di tengah-tengah ekspresi**.

Hal itu bakal bikin JavaScript bingung dalam memahami ekspresi yang kamu tulis lho 😱

```
» console.log(6  
+ 10  
);
```

```
← undefined
```

16

```
» console.log(6;  
+ 10  
);
```

❗ SyntaxError: missing) after argument list [Learn More]



Comment (Komentar)

Mirip seperti di HTML dan CSS, komentar dalam JavaScript merupakan **deretan kode yang dihiraukan oleh engine** yang menjalankannya.

Meski gitu, **komentar ini penting** lho!

Adanya komentar ini bakal **ngasih informasi ke developer** lain atau sebagai **pengingat ke diri sendiri tentang kode yang ditulis~**

// jangan lupa kerjain
baris ini ya

/* hey dibaca ya!
jangan lupa loh!

// baca plissss!!!

...aku dikacangin, ta?





Dalam menuliskan komentar di JavaScript, kita bisa pakai dua cara ini lho~

1. **Komentar Single-Line** (Satu Baris)
2. **Komentar Multi-Line** (Banyak Baris)

Biar lebih paham, lihat yuk contoh berikut ☺

```
// This is single line comment
```

```
/* This is multiline comment 1  
   This is multiline comment 2  
 */
```



Komentar Single-Line

```
» // Kode ini berguna untuk nge-hack NASA
  console.log("Hack NASA")
← undefined
Hack NASA
```

Komentar Single-Line mengikuti statement

```
» console.log("Bikin Google"); // Kode ini berfungsi untuk bikin startup bernama Google.
← undefined
Bikin Google
```



Komentar Multi-line

```
» /* Pernyataan 1, hanyalah pernyataan biasa, console.log */
  console.log("Pernyataan 1");
```

```
← undefined
```

Pernyataan 1

```
» /* Kode ini dibikin oleh saya,
  dibikin pada tanggal 2 Juli 2019.
  Fungsi kode ini buat nge-hack NASA
  */
  console.log("Hack NASA");
```

```
← undefined
```

Hack NASA

Jenis komentar ini dimulai dengan garis miring dan bintang `"/*"` dan diakhiri dengan menggunakan bintang dan garis miring `"*/"`.

Setelah paham tentang struktur kode dalam JavaScript, kita perlu paham juga nih **variabel** yang ada di JavaScript~



Tahu nggak, ketika belajar JavaScript,
kita nggak bisa lepas nih dari yang
namanya **variabel!**

Wah, sepenting itu ya?

Emangnya, variabel ini apa sih?





Ngomong-ngomong, variabel itu apa ya? 🤔

Dalam dunia pemrograman, kita pasti butuh data terus nih untuk memudahkan kita dalam membuat suatu program~

Oleh karena **data-data tersebut perlu wadah** untuk menyimpannya agar mudah kita akses, data tersebut bisa dimasukkan ke dalam **variabel** lho 😎





Variabel itu mirip KTP sob!

Ngomong-ngomong, variabel itu kayak KTP lho~

Coba amati. Dalam sebuah KTP, pasti di dalamnya memuat berbagai data diri yang cukup detail kan? Seperti NIK, nama lengkap, tempat dan tanggal lahir, alamat lengkap, status diri dan sebagainya.





Nah, tahu nggak, variabel dalam JavaScript ini bisa diibaratkan seperti “nama lengkap” dalam KTP lho. Sementara itu, nama “Sabrina” adalah nilai atau value dari variabel nama lengkap itu.

Jadi, bisa disimpulkan nih kalau variabel adalah sebuah nama yang mewakili sebuah nilai~





Terus, cara bikin variabel dalam JavaScript gimana ya? 😕

Kalau tahu arti variabel aja pasti rasanya masih kurang kan?

Nah, biar pemahamanmu makin lengkap, ikut simak yuk penjelasan berikut~





masih ingat sama statement?

Nah, kode berikut adalah sebuah *statement* yang fungsinya untuk bikin suatu variabel yang bernama **pesan**.

Setelah variabel “pesan” dibuat, kita bisa masukin data-data ke dalamnya dengan cara seperti di samping ini~

```
» let pesan;
```

```
» let pesan;
```

```
← undefined
```

```
» pesan = "Hello World";
```

```
← "Hello World"
```



Saat kamu coba jalankan `console.log(pesan)`, hasil yang keluar adalah data yang kamu masukin tadi.

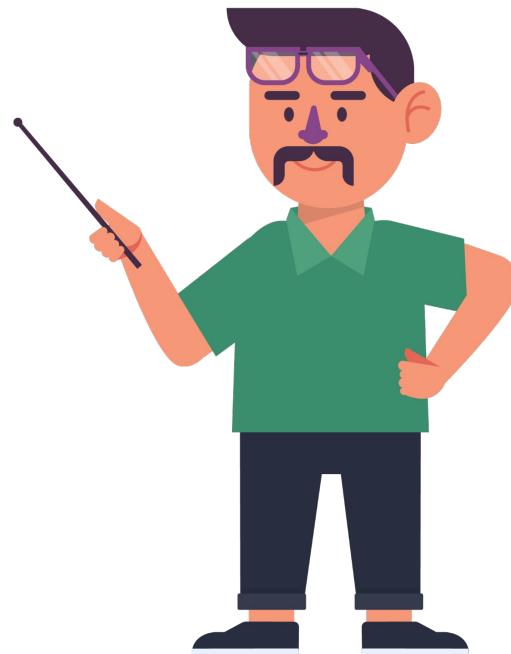
Nah, ini adalah bukti bahwa data sudah **tersimpan** di dalam variabel `pesan` 😊

```
» let pesan;  
← undefined  
  
» pesan = "Hello World";  
← "Hello World"  
  
» console.log(pesan);  
← undefined  
  
Hello World
```



Selain cara yang tadi, kita juga bisa nih bikin variabel pakai cara lain.

Wah, apa aja tuh? Mau tahu dong~





Cara lain bikin variabel dan masukin data:

```
» let pesan = "Hello World"; // Bikin variabel sekaligus memasukkan datanya (nilai)  
console.log(pesan);
```

Kamu juga bisa bikin beberapa variabel sekaligus nih:

```
» let pengguna = 'Didit', umur = 23, pesan = 'Hello';
```

```
» // Atau biar lebih enak dibaca  
let pengguna = 'Didit'  
, umur = 23  
, pesan = 'Hello';
```



Tahu nggak sih~

Ada cara gampang lho biar kamu bisa lebih paham tentang konsep "variabel".

Anggap aja variabel itu kayak **kotak** buat tempat datamu. Nah, kamu bisa tempelin stiker ke kotak itu biar mudah dicari lho~





Kalau kita ilustrasiin pakai JavaScript, kurang lebihnya jadi gini ya!

Pesan variabel adalah kotak **berlabel "pesan"** dengan **nilai "Halo!"** di dalamnya.

Kamu bisa ngasih nilai apapun ke dalam kotak itu dan kamu juga bisa mengubahnya sebanyak yang kamu mau.

```
let pesan;  
  
pesan = 'Hello';  
  
pesan = 'World'; // nilai diganti  
  
console.log(pesan);
```

Ngomongin soal variabel, nggak semua data variabelnya bisa diubah-ubah lho~

Selain variabel biasa, ada juga nih variabel konstan atau yang biasa disebut Variabel - Const.





**Sebenarnya, variabel konstan ini apa ya?
Terus, cara pakainya gimana dong?**





Yang pasti-pasti emang pasti, toh

Konstan itu tetap 🚚

Seperti namanya, variabel konstan itu adalah **variabel yang value-nya tetap**, nggak bisa diganti bagaimanapun kondisinya 😱

Contoh variabel konstan kayak berikut ini nih sob~





Kalau begini, dilarang keras untuk mengganti nilai dari variabel itu ya.

```
>> /* Kalo tadi kita bikin variabel pake kata "let"
   Nah, kalo kita pengen bikin variabel konstan,
   kita pake kata "const" */
const bumi = "bulat";
← undefined
>> bumi = "datar";
! ▶ TypeError: invalid assignment to const `bumi` [Learn More]
```



Kapan variabel konstan dipakai?

Ada beberapa kasus ketika kita **perlu banget nih pakai konstanta sebagai variabel~**

Kayak nama panggilan, pasti kita lebih mudah mengingatnya daripada mengingat nama lengkap orang lain kan?





Kapan variabel konstan dipakai?

Ada beberapa kasus ketika kita **perlu banget nih pakai konstanta sebagai variabel~**

Kayak nama panggilan, pasti kita lebih mudah mengingatnya daripada mengingat nama lengkap orang lain kan?



Nah, dalam JavaScript, **variabel konstan ini bisa dipakai untuk bikin semacam nama panggilan gitu**.

Jadi, ketika kita bikin suatu perintah dalam JavaScript, kita nggak perlu menulis ulang lagi 😊





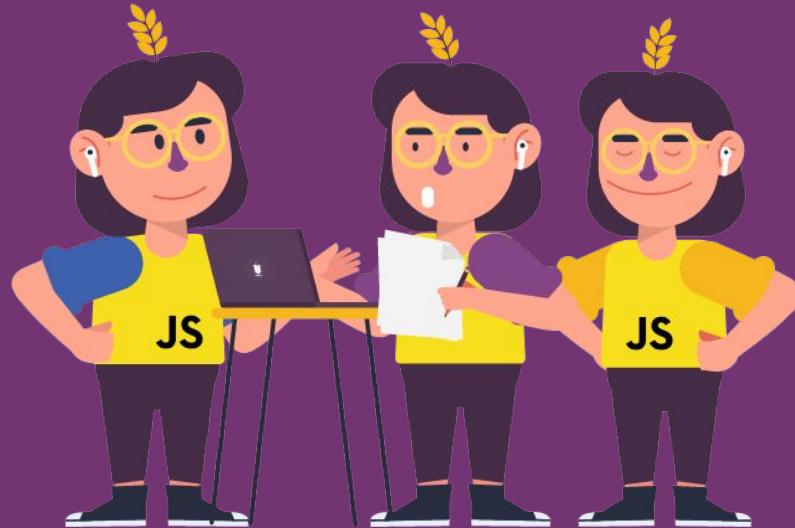
Biar lebih jelas, coba yuk perhatiin contoh **konstanta untuk warna dalam format yang disebut "web" (heksadesimal)** seperti berikut~

```
const WARNA_MERAH = "#F00";
const WARNA_HIJAU = "#0F0";
const WARNA_BIRU = "#00F";
const WARNA_ORANGE = "#FF7F00";

// ... ketika kita butuh mengambil warna maka
let warna = WARNA_BIRU;
console.log(warna); // #00F
```

Ketika bahas variabel, pasti kita bakal nemuin nih kalau data itu ada banyak tipenya kan?

Nah, apa aja ya tipe-tipenya?

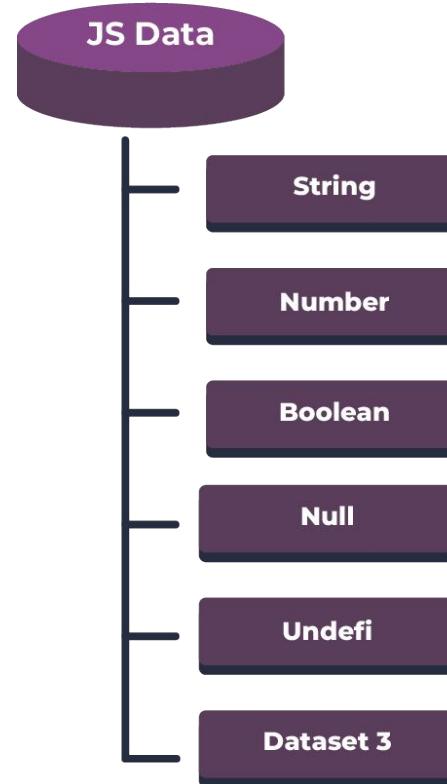




Tipe-tipe data

Tipe data adalah jenis-jenis data yang bisa disimpan di dalam variabel. Nah, dalam JavaScript, tipe data ini ada 6 macam lho~

Wah, apa aja tuh?





String

String adalah tipe data yang menyimpan nilai berupa **karakter**, mulai dari **huruf, angka, simbol dan sebagainya**, di mana nilainya akan dikelilingi oleh tanda kutip.





contoh String kayak gini nih sob~

```
// Pake tanda kutip dua
"Ini string";
"Ini string tapi pake karakter aneh-aneh ()*!)(^&^%^#*";
"Ini string pake angka 123456";

// Pake tanda kutip satu
'Ini string tapi aku bisa nambahin kutip dua di string ini, liat nih ', tuh kan!';
'Gapapa kan kalo pake kutip satu?';

// Pake backtick
`Ini string tapi petiknya kebalik`;
`${10 + 11}`;
```



FYI, di dalam string, kamu bisa manggil variabel yang sudah kamu deklarasikan dan juga bisa bikin suatu ekspresi di dalamnya.

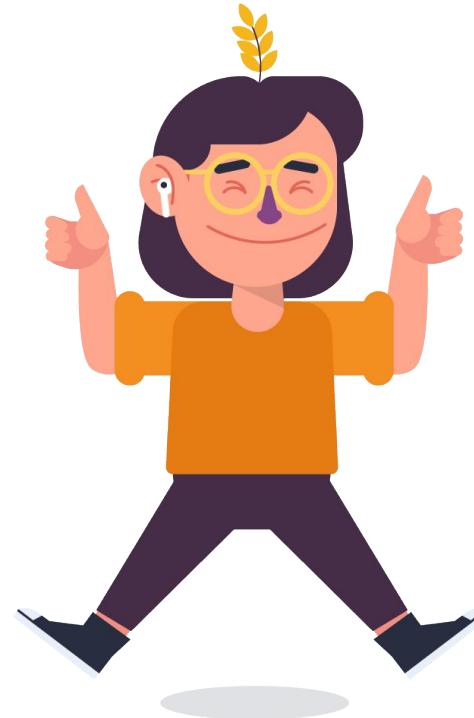
String semacam ini disebut dengan **template literal**.





Template literal wajib pakai **backtick** (`) ya. Lalu untuk nambahin suatu kode yang akan dieksekusi di dalam string, kamu juga perlu nambahin karakter seperti ini \${...}.

Nah kode yang ada di dalam karakter itu akan dievaluasi oleh JavaScript (dijalankan).





kalau dieksekusi dalam JavaScript, hasilnya bakal kayak gini nih sob!

```
» let nama = "Sabrina";
  let pekerjaan = "Fasilitator Binar";

  console.log(`Hi, nama aku ${nama} dan pekerjaanku sekarang adalah ${pekerjaan}`);
← undefined
Hi, nama aku Sabrina dan pekerjaanku sekarang adalah Fasilitator Binar
```



Number

Sesuai dengan namanya, tipe data ini berbentuk **angka** berupa **bilangan bulat atau bilangan koma**.

Tipe data number nggak punya tanda ("").



```
// Ini Number  
const umur = 22;  
const variabelNumber = 22.22;  
const variabelNumber2 = 0.244;
```



Dengan **tipe data number**, kamu bisa melakukan **operasi matematika**, termasuk penjumlahan, perkalian, pembagian serta pengurangan lho~

Oh iya, bagian ini akan dibahas lebih detail nantinya ya 😊





Contoh tipe data number dalam JavaScript~

```
● ● ●  
  
const umurKamu = 22;  
const umurKakek = 40;  
const umurNenek = 43;  
  
const totalUmur = umurKamu + umurKakek + umurNenek;  
console.log(totalUmur) // Output => 105
```



Boolean

Tipe data ini cuma **punya dua nilai saja**, true atau false. Tipe data boolean akan sangat sering dipakai pada pengecekan suatu data.





Contoh tipe data boolean dalam JavaScript~

```
let apakahSedangTerjadiPandemi = true;  
let apakahAkuHarusKeluarRumah = false;
```



Oh iya, perlu diketahui nih sob kalau **nilai boolean itu bisa aja datang dari suatu perbandingan~**

Perbandingan yang dimaksud disini adalah **Logical Operator** yang akan dibahas lebih lanjut nantinya 😊

```
» let apakahLebihDari2 = 1 > 2;
  console.log(apakahLebihDari2);
← undefined
false

```

```
» let a = 1;
let b = 2;
let apakahASamaDenganB = a == b;
  console.log(apakahASamaDenganB);
← undefined
false

```



Null

Null adalah **tipe data yang nilainya kosong**. Kalau nilainya kosong, null ini mewakili apa dong?

Tahu nggak sob, dalam JavaScript, null ini **bukan sebuah referensi ke objek yang nggak ada** atau "null pointer" seperti pada beberapa bahasa pemrograman lain lho~

Dalam kondisi tertentu (conditional), data ini akan dianggap **false**.

```
>> let kosong = null;  
      console.log(kosong);
```



Undefined

Adalah tipe data yang **nggak terdefinisi**, artinya nggak ada sumber nilai pada data tersebut.

Nilai ini biasanya muncul saat kamu mendefinisikan variabel tetapi kamu nggak masukin nilai ke dalamnya sama sekali.

Dalam *conditional* data ini akan dianggap *false*.

```
» let tidakTerdefinisi;  
← undefined  
» console.log(tidakTerdefinisi);  
← undefined  
undefined
```



Oh iya, tipe ini juga sering disebut tipe data primitif lho~

Kenapa ya?

Itu karena karena tipe ini secara langsung merepresentasikan suatu nilai tertentu nih sob 😎





Object

Jenis ini merupakan suatu tipe data yang **menyimpan koleksi tipe data yang lain**. Itu artinya, suatu object bisa memiliki string, number, boolean sekaligus lho~

Karena object adalah koleksi dari berbagai tipe data, kita perlu punya kunci nih buat manggil salah satu koleksi data dalam object tersebut~





Nah, kunci ini nih yang biasa disebut sebagai **key**. Di dalam key ini, kita bakal nemuin **nilai atau value ya~**

Ketika **key dan value** ini dikombinasikan, hal ini diistilahkan sebagai **property!**





Oh iya sob, object ini bisa menyimpan berbagai jenis tipe data yang ada lho.

Doi juga bisa memetakan data-data di dalamnya menggunakan key kayak contoh di samping ini nih~

Dari contoh di samping, yang dimaksud dengan key adalah **nama, umur, jenisKelamin, sudahMenikah**

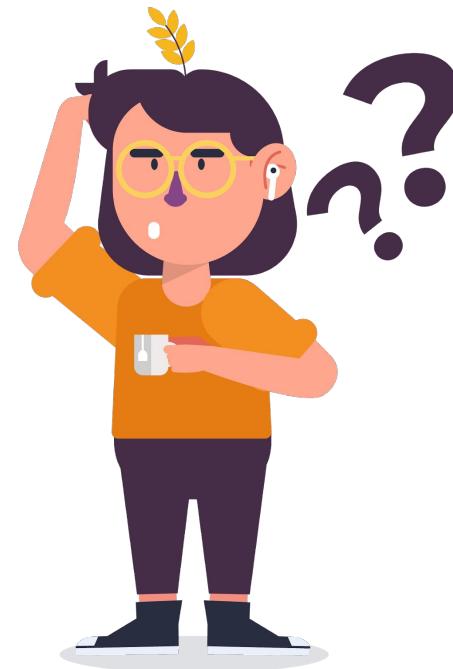


```
const contohObjek = {  
  nama: "Katharina",  
  umur: 23,  
  jenisKelamin: "Perempuan",  
  apakahSudahMenukah: false  
}
```



kalau tipe datanya banyak, biar gampang cari tipe data tertentu gimana ya?

Kamu bisa ngecek tipe data yang ada dengan menggunakan **operator typeof**.





Kalau tipe datanya banyak banget, cari tipe tertentu gimana ya?

It's okay sob! Kita bisa ngecek dulu tipe data yang ada dengan menggunakan **operator typeof**.

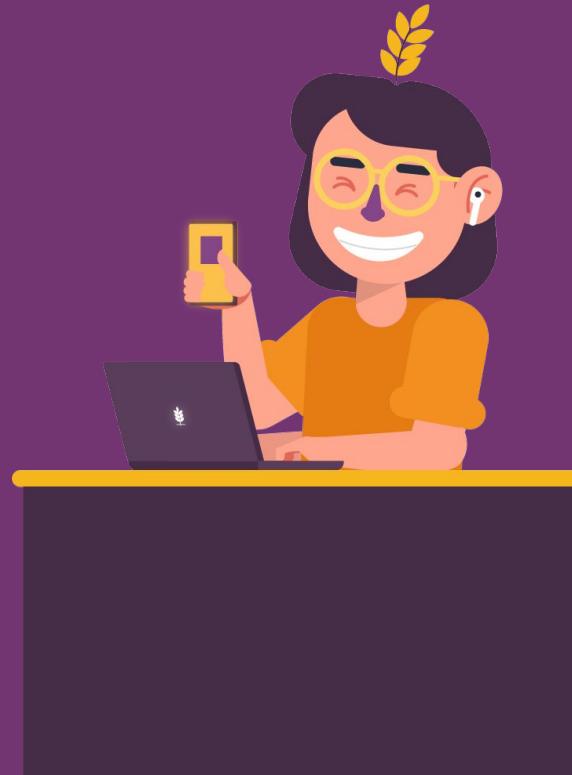
Penulisan syntax-nya sendiri ada dua cara ya seperti ini:

- Sebagai operator → `typeof x`
- Sebagai function → `typeof(x)`

```
>> let x = "Hello World";
← undefined
>> typeof x;
← "string"
>> typeof 123
← "number"
>> typeof true;
← "boolean"
>> typeof {
    name: "Sabrina"
};
← "object"
>> typeof null;
← "object"
>> typeof undefined;
← "undefined"
```



Setelah paham tentang jenis-jenis variabel, kita bakal lanjut nih belajar tentang cara mengoperasikannya~





Bicara soal operasi, kamu ingat nggak tentang operasi matematika waktu sekolah dulu?

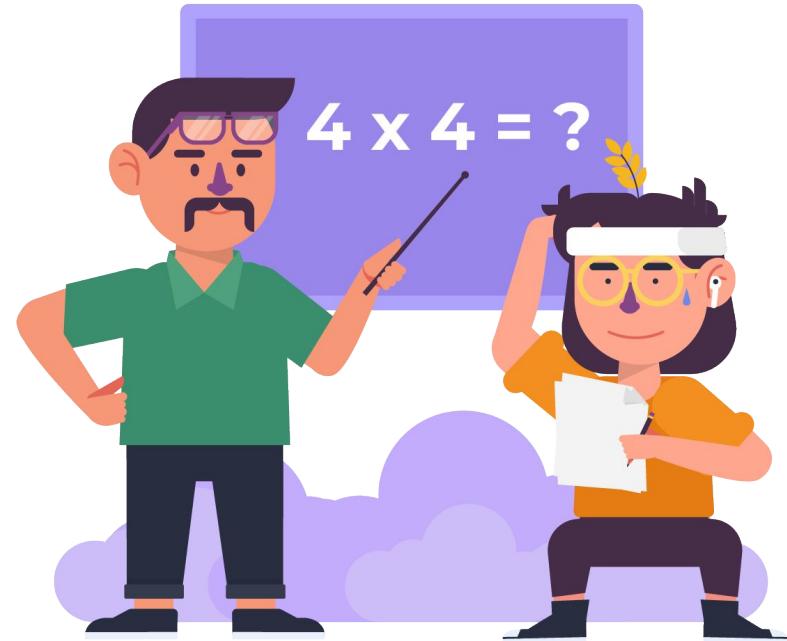




4 x 4 = 16, sempat nggak sempat harus dibalas~

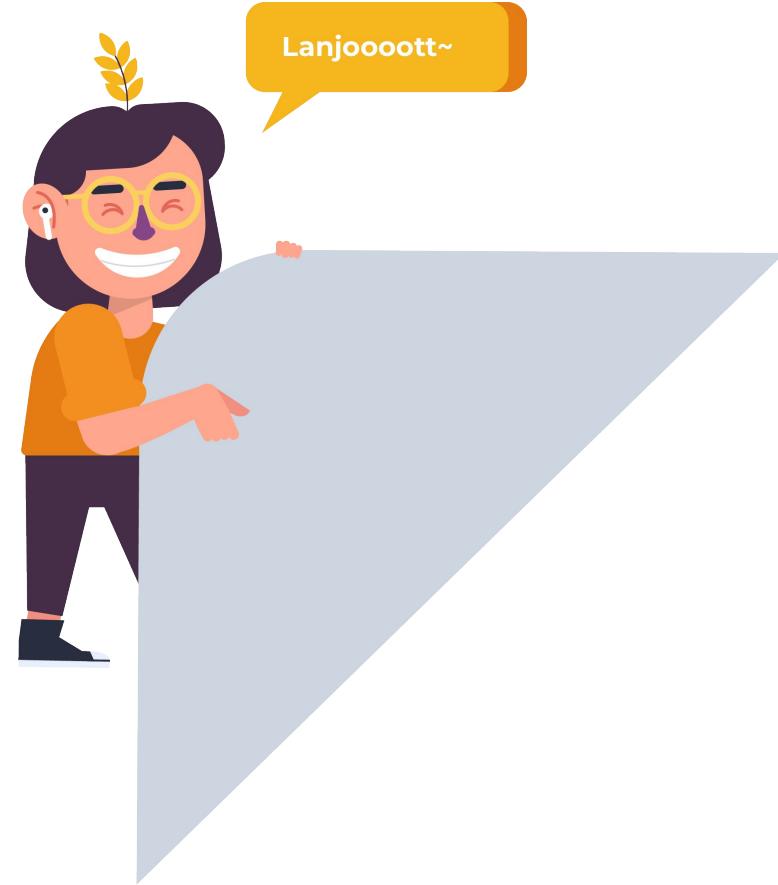
Ingat nggak, waktu sekolah dulu, pasti kita pernah belajar kan tentang operasi sederhana matematika seperti penjumlahan (+), perkalian (*), pengurangan (-), dan operator-operator lainnya?

Nah, kali ini kita nggak akan bahas itu lagi kok sob. Tenang aja 😊





Tapi, sebelum lanjut, kamu perlu tahu nih ada beberapa istilah penting yang bakal kepakai. Wah, apa aja ya istilahnya?





Istilah yang perlu diketahui 📖

Beberapa istilah berikut bakal sering banget kamu temuin ketika mengoperasikan JavaScript nih Sob. Apa aja ya istilahnya?

- Operan
- Unary
- Binary





Operan

Operan ini adalah pelaku dari suatu operasi. Maksudnya?

Coba kamu perhatiin contoh di samping. Dalam operasi di samping, ada dua operan nih sob!

pertama adalah operan sisi kiri yang bernilai 1 dan kedua adalah operan sisi kanan yang bernilai 4 ✨





Unary

berbeda dengan operan, unary ini hanya punya 1 operan saja
lho~

Contohnya kayak di samping ini nih! Kalau diperhatikan,
terdapat **-x** dalam operasinya.

Nah, **-x** inilah yang disebut sebagai unary~

```
>> let x = 1;
    x = -x;
    console.log(x) // Menggunakan unary negation
```



Binary

Kalau unary hanya punya 1 operan saja, maka binary ini operan-nya lebih dari satu lho~

Contohnya kayak di samping ini nih sob!

```
» let x = 10;  
let y = 5;  
console.log(x - y); /*  
Terjadi operasi di dalam console.log()  
Dimana x - y, x adalah operan dan y adalah operan  
Operasi ini kita sebut dengan binary  
*/
```



Binary + pada rangkaian string

Ternyata, operator plus nggak cuma bisa dipakai dalam operasi aritmatika aja lho sob!

Selain itu, operator plus ini juga bisa dipakai untuk melakukan concatenation (perangkaian) sebuah string.

Contohnya kayak berikut ini nih. Let's go 🚀





Kalau biasanya operator plus (+) dipakai untuk menjumlahkan angka, biner plus (+) yang diterapkan pada String punya fungsi lain nih~

Fungsinya itu bukan untuk menjumlahkan, melainkan untuk menyatukan seperti hubunganmu dengan doi. Ea~

```
» let a = "Hello";
let b = "World";
console.log(a + b);
← undefined
HelloWorld
```



Coba deh perhatiin ☺

Dari gambar di samping, kita bisa lihat nih kalau salah satu operannya adalah string. Nah, hal ini bakal berdampak pada operan yang lain. Mereka juga bakal dikonversi jadi string juga lho~

```
» /* Jika salah satu
operan adalah string,
maka operan yang
lain akan dianggap string juga */
console.log("1" + 2);
← undefined
```

12



Coba perhaiiin ini juga~

Operasi berjalan dari kiri ke kanan. Nah, kalau ada dua angka yang diikuti oleh string, angka-angka itu akan ditambahkan sebelum dikonversi ke string.

Menurutmu, kalau angka-angkanya dikombinasikan dengan huruf, bakal dikonversi juga nggak ya?

```
» /* Namun berbeda juga
kalau di dalam operasi itu
terdapat operan yang bisa melakukan
operasi aritmatika.
Javascript akan menjalankan
operasi tersebut terlebih dahulu baru
menambahkannya ke string */
console.log(4 + 4 + "2"); // Hasilnya bukan "442"
← undefined
```



Numeric conversion, unary +

Tahu nggak kalau plus (+) itu terdiri dari dua bentuk lho~

Yang pertama kan binary dan sudah kita bahas sebelumnya.
Nah, yang kedua ini namanya unary.



Numeric conversion, unary +

Tahu nggak kalau plus (+) itu terdiri dari dua bentuk lho~

Yang pertama kan binary dan sudah kita bahas sebelumnya.
Nah, yang kedua ini namanya unary.





Unary plus atau dengan kata lain operator plus (+) diterapkan pada nilai yang tunggal dan tidak mengubah apapun pada angka.

Tetapi jika operan bukan angka, plus unary mengubahnya menjadi angka.





Oh iya, ketika mengonversi string jadi number sebenarnya bisa kamu lakukan dengan function **Number("12345")**

Tapi ada cara kilat lho kayak di samping. Kamu udah coba belum?

```
// Tidak berpengaruh pada angka
let x = 1;
console.log( +x ); // 1

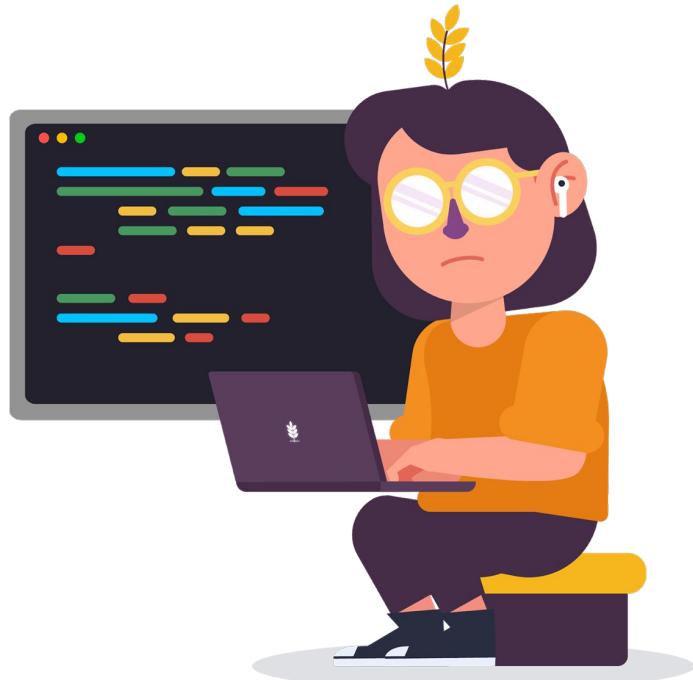
let y = -2;
console.log( +y ); // -2

// Mengkonversi yang bukan angka
console.log( +true ); // 1
console.log( +" " ); // 0
```



FYI, konversi string menjadi number akan sering kamu jumpai saat bikin form HTML. Soalnya, hasil input dari form HTM mungkin banget berupa string nih sob!

Nah, biar makin paham seputar string, coba deh perhatiin cara melakukan operasi aritmatika dengan data string berikut ini 😊





Contoh yang salah

```
let apel = "2";
let mangga = "3";

console.log( apel + mangga ); // "23", string binar plus digabungkan
```

Jika kita mau menggunakannya sebagai angka, kita perlu mengonversi dan menjumlahkannya.



Contoh yang benar ✓

```
let apel = "2";
let mangga = "3";

// kedua nilai dikonversi menjadi angka sebelum buler plus
console.log( +apel + +mangga ); // 5

// variasi yang panjang
// console.log( Number(apel) + Number(mangga) ); // 5
```



Tingkatan prioritas operasi

Sama seperti matematika yang diajarkan di sekolah, setiap operasi punya tingkat prioritas, **mana yang harus dikerjakan duluan.**





Contohnya begini~

$$2 + 2 * 10$$

Dari operasi di atas, maka yang akan dijalankan lebih dahulu adalah **2 * 10**



Tapi, kalau dalam Programming
gimana ya?





Tabel tingkatan prioritas dari tiap-tiap operasi

Ada banyak operator di bahasa pemrograman JavaScript, dan setiap operator punya urutan prioritas yang sesuai.

Angka yang tertinggi akan dieksekusi lebih dahulu dan kalau prioritas pengurutannya sama, maka urutan eksekusi dilakukan dari kiri ke kanan.

Prioritas	Nama	Tanda
17	unary plus	+
17	unary negation	-
15	perkalian	*
15	pembagian	/
13	penambahan	+
13	pengurangan	-
3	sama dengan	=



Seperti yang kamu lihat pada tabel, "unary plus" punya prioritas 17 yang lebih tinggi dari 13 "penambahan" (binary plus).

Itulah kenapa dalam ekspresi seperti "+apel + +mangga", plus unary berfungsi sebelum penambahan.





Gimana? Gimana? Materinya tambah seru atau tambah bikin pusing?

Jangan pusing, karena pada dasarnya kalau kamu pahami alias nggak cuma dihafal, materi ini ngasih banyak banget ilmu yang bermanfaat dalam pengembangan aplikasi.

Semangat cinta! Kamu pasti bisa! 😊





Jenis-jenis operator~

Ngomongin soal operator, ini bukan operator telepon seluler ya sob!

Untuk menggunakan JavaScript, kita perlu tahu beberapa operator yang sering dipakai lho. Nah, Operatornya ada 6 jenis nih.

Apa aja ya?





Assignment (Sama Dengan)

Jenis ini merupakan penentuan nilai dari sebuah data.

Ketika kamu bilang `let a = 1`, maka kamu sudah melakukan sebuah *assignment* terhadap variabel a.

Hal ini berarti, kamu ngasih tau bahwa variabel a bernilai 1.

```
let x = 2 * 2 + 1;  
  
console.log( x ); // 5
```



Kamu juga bisa nih kalau mau melakukan assignment berantai seperti berikut.

Assignment berantai mengevaluasi dari kanan ke kiri. Pertama, ekspresi paling kanan $2 + 2$ dievaluasi kemudian ditetapkan ke variabel di sebelah kiri: c, b dan a.

Pada akhirnya, semua variabel berbagi nilai tunggal.

```
let a, b, c;  
  
a = b = c = 2 + 2;  
  
console.log( a ); // 4  
console.log( b ); // 4  
console.log( c ); // 4
```



Eksponensial (Perpangkatan)

Operator eksponensial `**` adalah tambahan terbaru untuk bahasa pemrograman.

Untuk bilangan asli b , hasil dari `**` adalah dikalikan dengan sendirinya b kali.

```
console.log( 2 ** 2 ); // 4 (2 * 2)
console.log( 2 ** 3 ); // 8 (2 * 2 * 2)
console.log( 2 ** 4 ); // 16 (2 * 2 * 2 * 2)
```



Increment dan Decrement (Peningkatan atau Penurunan)

Penting dipahami, operator ini hanya berlaku untuk variabel saja.

Kalau kamu menambahkan operator ini pada sebuah angka secara langsung, maka akan menyebabkan error

Error 404





Increment

```
let hitung = 2;  
hitung++;           // bekerja sama dengan penghitung = penghitung + 1, tetapi lebih pendek  
console.log( hitung ); // 3
```

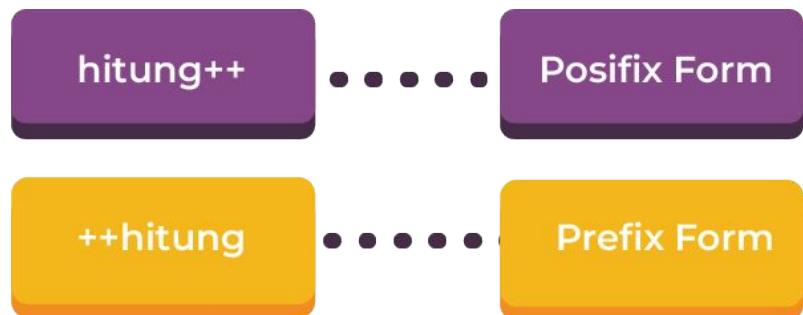
Decrement

```
let hitung = 2;  
hitung--;           // bekerja sama dengan penghitung = penghitung - 1, tetapi lebih pendek  
console.log( hitung ); // 1
```



Operator ++ dan -- dapat ditempatkan sebelum atau sesudah variabel.

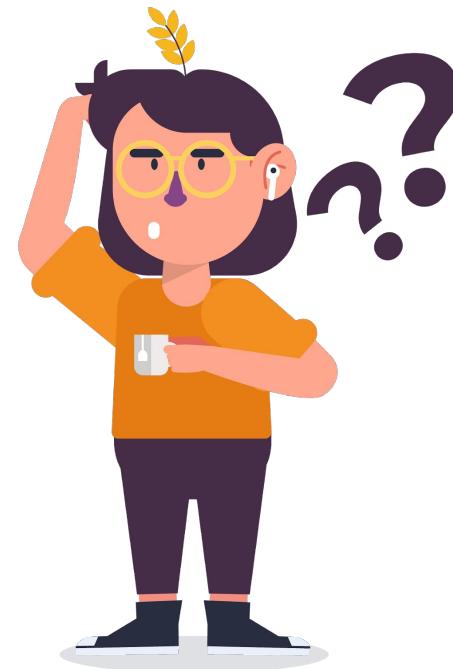
Ketika operator berada setelah variabel maka operator akan berbentuk "posifix form": hitung++, sebaliknya kalau operator berada di awal variabel maka akan berbentuk "prefix form": ++hitung.





Kedua pernyataan tadi melakukan hal yang sama yaitu meningkatkan (increase) hitung dengan 1.

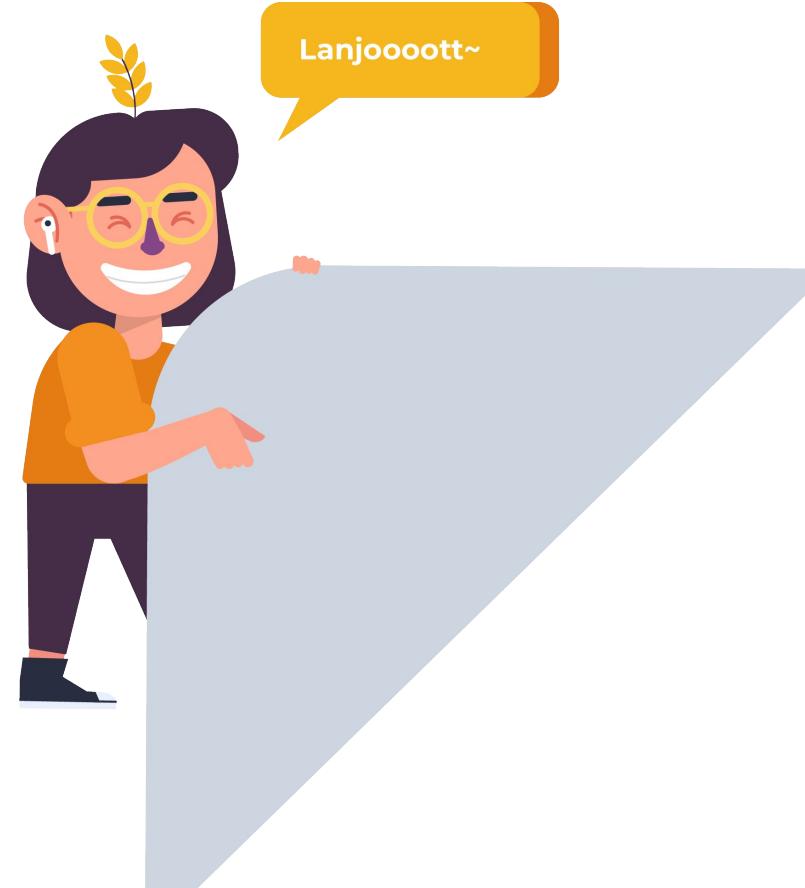
Apakah ada bedanya? Ada, tapi kamu hanya akan melihat perbedaan itu saat kamu menggunakan nilai yang dikembalikan dari `++` / `-`. Mari kita perjelas.





Semua operator mengembalikan nilai. Prefix form mengembalikan nilai baru, sementara bentuk postfix form akan mengembalikan nilai lama sebelum dilakukan increasing/decreasing.

Untuk melihat perbedaannya, yuk kita lihat contoh berikut!





Contoh perbedaannya~

```
let hitung = 1;  
let a = ++hitung; // (*)  
  
console.log(a); // 2
```

```
let hitung = 1;  
let a = hitung++; // (*) mengubah ++hitung ke hitung++  
  
console.log(a); // 1
```



Bitwise

Operator Bitwise memperlakukan argumen sebagai angka integer 32-bit dan bekerja pada level representasi binernya.

Operator ini nggak spesifik hanya untuk JavaScript, tapi juga mendukung sebagian besar bahasa pemrograman.

Biar lebih paham, simak contoh berikut yuk!

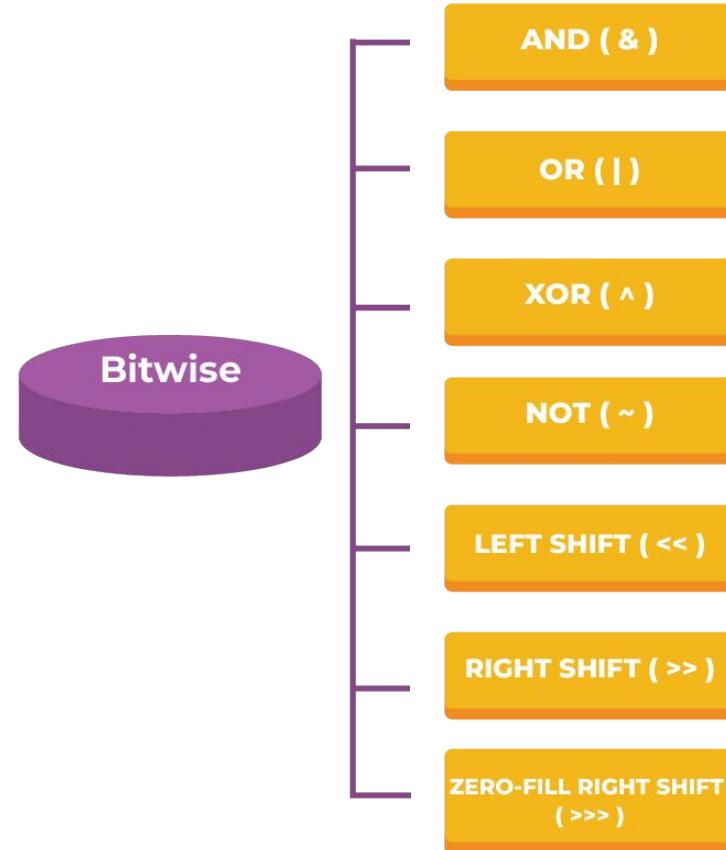




Contoh bitwise

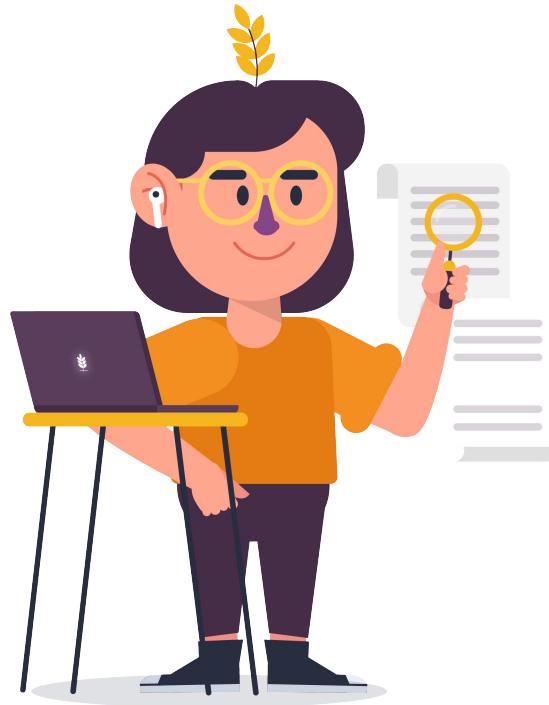
Berikut adalah beberapa contoh operator yang tersedia:

- AND (&)
- OR (|)
- XOR (^)
- NOT (~)
- LEFT SHIFT (<<)
- RIGHT SHIFT (>>)
- ZERO-FILL RIGHT SHIFT (>>>)





Operator ini sangat jarang digunakan sih. Untuk bisa lebih paham, kamu perlu belajar representasi *low-level number*. Untuk referensi kamu bisa baca artikel [Operator Bitwise MDN](#).





Modify-in-place

Kita sering membutuhkan penerapan operator ke variabel dan menyimpan hasil baru dalam variabel yang sama juga.

Kita mempunyai variabel `n` yang ingin kita tambahkan dan kalikan dengan beberapa angka, di baris kedua berarti $2 + 5$, dimana hasil dari baris kedua adalah 7, dan disimpan kembali ke variabel `n`





Notasi tadi bisa disingkat jadi seperti di samping nih sob~

Oh iya, "modify-and-assign" operator berlaku juga untuk semua operator aritmatika dan bitwise: /=, -= dll.

Operator itu memiliki prioritas yang sama dengan prioritas normal, sehingga akan dijalankan sebagian perhitungan lainnya.

```
let n = 2;  
n = n + 5;  
n = n * 2;
```

```
let n = 2;  
n += 5; // sekarang n = 7 (sama seperti n = n + 5)  
n *= 2; // sekarang n = 14 (sama seperti n = n * 2)  
  
console.log( n ); // 14
```

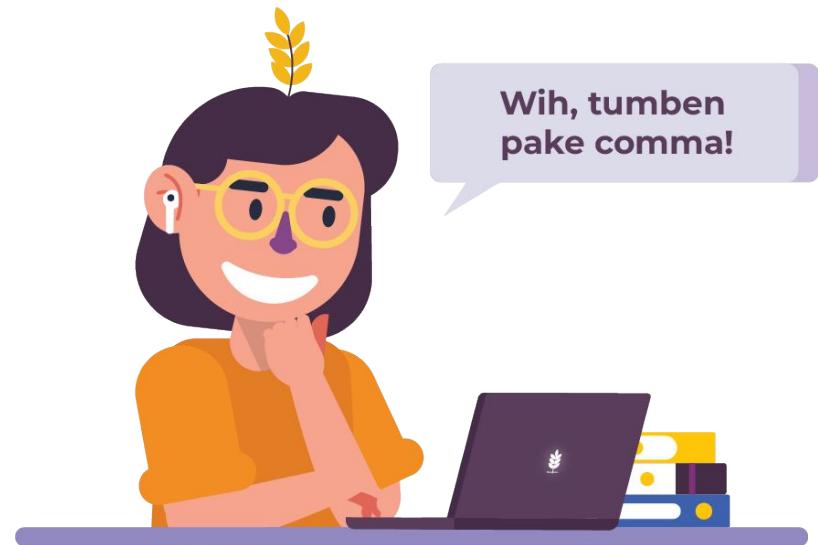
```
let n = 2;  
  
n *= 3 + 5;  
  
console.log( n ); // 16 (rbagian kanan akan dievaluasi terlebih dahulu dengan
```



Comma

Operator koma adalah salah satu operator yang paling langka dan paling nggak biasa.

Kadang dipakai untuk menulis kode yang lebih pendek, jadi kamu perlu tahu untuk paham apa yang akan terjadi.





Operator koma memungkinkanmu untuk mengevaluasi beberapa ekspresi dan membaginya dengan koma.

Masing-masing dievaluasi, tapi hanya hasil yang terakhir yang akan dikembalikan.

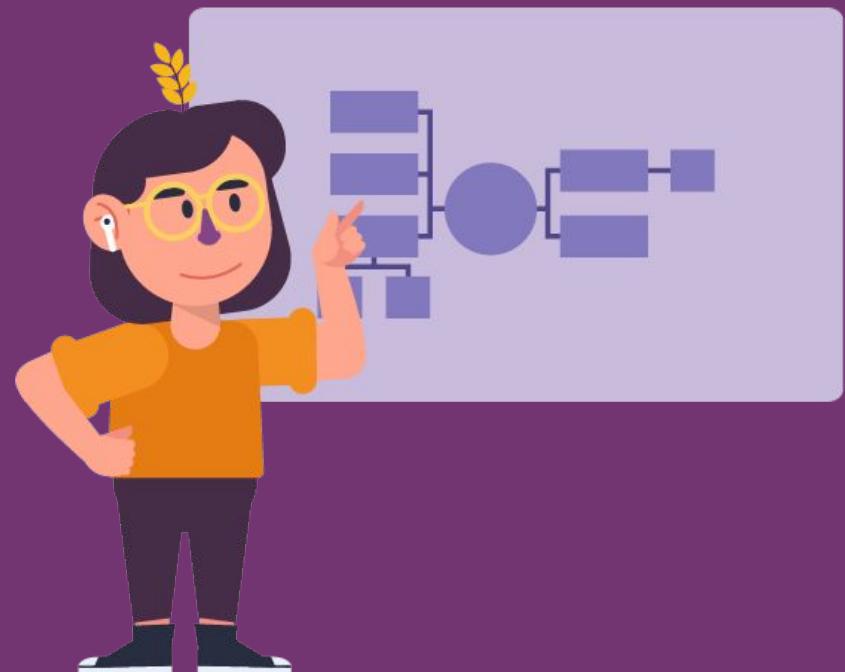
```
let a = (1 + 2, 3 + 4);
```

```
console.log( a ); // 7 (Hasil dari 3 + 4)
```

Akhirnya, sampai di bagian akhir nih sob!

Dalam bagian ini, kita bakal kenalan juga dengan tiga jenis logical operator dalam JavaScript~

Wah, apa aja tuh?





Jenis-jenis operator~

Selanjutnya, kita kenalan juga dengan tiga jenis logical operator dalam JavaScript yuk. Mereka adalah:

|| (OR)

&& (AND)

! (NOT)





Meski disebut "logic", operator-operator ini juga bisa diterapkan pada nilai-nilai jenis apapun, nggak cuma boolean. Hasilnya juga bisa dari jenis apapun.





|| (OR)

Operator "OR" diwakili oleh dua simbol garis vertikal. Kayak di samping ini nih~

Dalam pemrograman klasik, logika OR hanya bertujuan untuk memanipulasi nilai boolean.

Kalau salah satu argumennya benar maka akan mengembalikan nilai benar dan kalau salah maka akan mengembalikan nilai salah.

```
hasil = a || b;
```

```
alert( true || true ); // true
alert( false || true ); // true
alert( true || false ); // true
alert( false || false ); // false
```



```
hasil = a && b;
```

&& (AND)

Operator AND ini dievaluasi dari kiri ke kanan. Kalau salah satu operan bernilai false maka semuanya akan dianggap bernilai false.

```
let jam = 12;
let menit = 30;

if (jam == 12 && menit == 30) {
    alert( 'Sekarang jam 12:30' );
}
```



! (NOT)

Nah, kalau operator ini berfungsi untuk membalikkan nilai boolean.

```
>> let hujan = false;  
     if (!hujan) console.log("Gausah pakai payung");  
     // Kalau tidak hujan gausah pakai payung
```

```
← undefined
```

```
Gausah pakai payung
```

Saatnya kita Quiz!





1. Berikut ini output yang akan muncul kalau kode di atas akan dieksekusi



```
console.log(+true);
console.log(!'Lydia');
```

- A. 1 dan false
- B. False dan NaN
- C. False dan false



1. Berikut ini output yang akan muncul kalau kode diatas akan dieksekusi



```
console.log(+true);
console.log(!'Lydia');
```

- A. 1 dan false
- B. False dan NaN
- C. False dan false

+true, karena ada operator (+) maka value true akan dikonversi jadi number. !'Lydia' karena 'Lydia' sendiri bernilai truthy maka ketika diberikan negasi (!), nilainya akan berubah jadi lawan dari true, yaitu false



2. Berikut ini output yang akan muncul kalau kode di atas akan dieksekusi



```
let greeting;  
greetign = {};// Typo!  
console.log(greetign);
```

- A. {}
- B. ReferenceError: greeting is not defined
- C. undefined



2. Berikut ini output yang akan muncul kalau kode di atas akan dieksekusi



```
let greeting;  
greetign = {};// Typo!  
console.log(greetign);
```

- A. {}
- B. **ReferenceError: greeting is not defined**
- C. undefined

Variabel 'greeting' nggak terdeklarasi



3. Apa nilai/value yang disimpan pada variable sum



```
const sum = eval('10*10+5');
```

- A. 105
- B. "105"
- C. TypeError



3. Apa nilai/value yang disimpan pada variable sum



```
const sum = eval('10*10+5');
```

- A. 105
- B. "105"
- C. TypeError

eval mengevaluasi kode yang dimasukan sebagai parameter adalah string. Dalam kasus ini eval akan mengevaluasi ekspresi, yaitu menjadi kan '10*10+5' menjadi sebuah operasi $10 * 10 + 5$



4. Manakah output yang akan muncul kalau kode di atas akan dieksekusi?



```
var bar = true;  
console.log(bar + 0, bar + "xyz", bar + true, bar + false );
```

- A. 1, “xyz”, 0, 1
- B. 1, “truexyz”, 2, 1
- C. 0, “truexyz”, 1, 1



4. Manakah output yang akan muncul kalau kode di atas akan dieksekusi?



```
var bar = true;  
console.log(bar + 0, bar + "xyz", bar + true, bar + false );
```

- A. 1, “xyz”, 0, 1
- B. 1, “truexyz”, 2, 1
- C. 0, “truexyz”, 1, 1

bar + 0 = 1, bar + xyz = ‘truexyz’, bar + true = 2, bar + false = 1



Yeayyy, selesai sudah pembahasan kita di Chapter 1 Topic 2 ini.

Selanjutnya, di Topic 3 kita bakal bahas...

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~



Terima Kasih!



Next Topic

loading...