



# Javascript Introduction

**Silver** - Chapter 2 - Topic 1

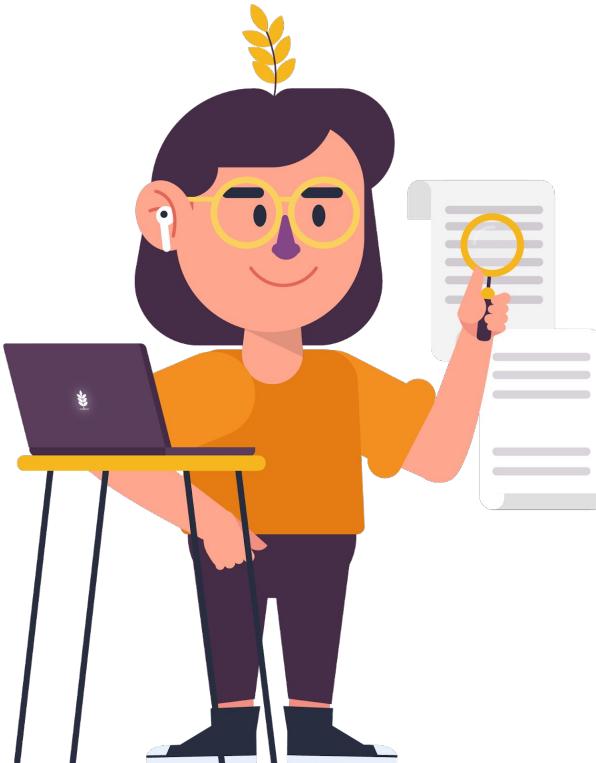
---

**Selamat datang di Chapter 2 Topic 1 online  
course Full-Stack Web dari Binar Academy!**



Setelah kamu selesai dijelaskan tentang dasar-dasar membuat halaman web, **Chapter 2 ini bakalan ngajak kamu buat lebih paham logika bahasa pemrograman Javascript untuk membuat halaman web.** Mulai dari penjelasan terkait struktur data, penggunaan operator, pembuatan expression sampai logika algoritma yang ada di dalamnya.

Pada topik pertama ini, kita bakal bahas tentang **apa itu Javascript, dan apa aja aturan yang ada dalam bahasa Javascript.** Cus langsung aja~





### Detailnya, kita bakal bahas hal-hal berikut ini :

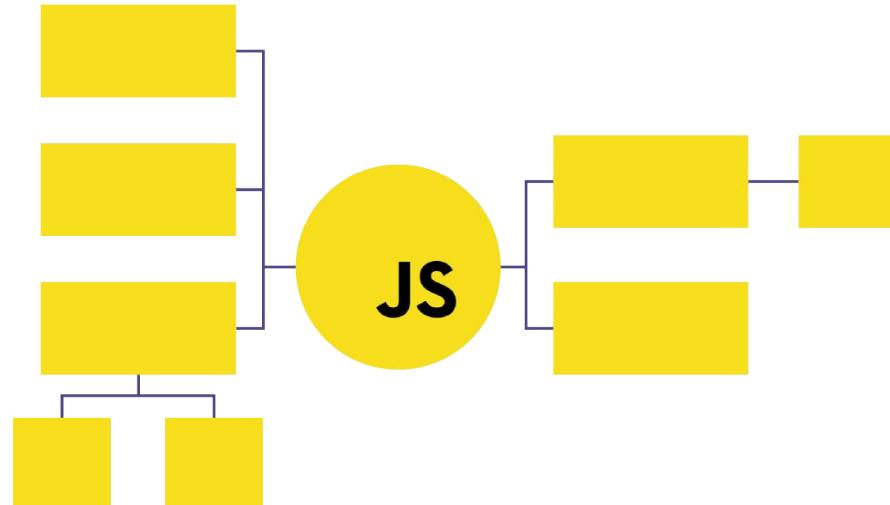
- Apa itu pemrograman dan bahasa pemrograman?
- Pengenalan bahasa pemrograman Javascript
- Aturan-aturan pada bahasa pemrograman Javascript



Dari chapter sebelumnya kita sudah bahas banyak tentang apa itu website dan komponen apa saja yang dibutuhkan untuk mengembangkan sebuah website. Ternyata menjadi 3 bahan utama :

- HTML
- CSS
- Javascript

2 poin pertama udah kita bahas di chapter 1. Yang mungkin sudah terjawab secara singkat, bahwa HTML, dan CSS ini adalah suatu teks yang berfungsi untuk mendefinisikan bagaimana tampilan website kita di dalam web browser. Dan tersisa 1 bahan lagi yang belum kebayang, yaitu **Javascript**.





## Apa itu Pemrograman?

Nah, untuk memahami apa itu Javascript, kita bisa perlu tau dulu nih **apa itu pemrograman**, dan **apa itu bahasa pemrograman**.

**Yuk kita bahas!**





### Bayangkan situasi berikut untuk lebih memahami apa itu pemrograman

Kamu tentu pernah merasa lapar tengah malam. Sayangnya, hari ini hujan jadi jasa pesan antar makanan pasti akan tiba terlambat.

Aha! Kamu ingat masih punya beberapa bahan makanan di kulkas yang bisa dimanfaatkan. Kamu pun pergi mengecek isi kulkas.





Di dalam kulkas ada nasi putih, telur, bawang, dan sawi. Berbekal resep nasi goreng hasil googling, kamu putuskan bahwa malam ini kamu akan masak nasi goreng!



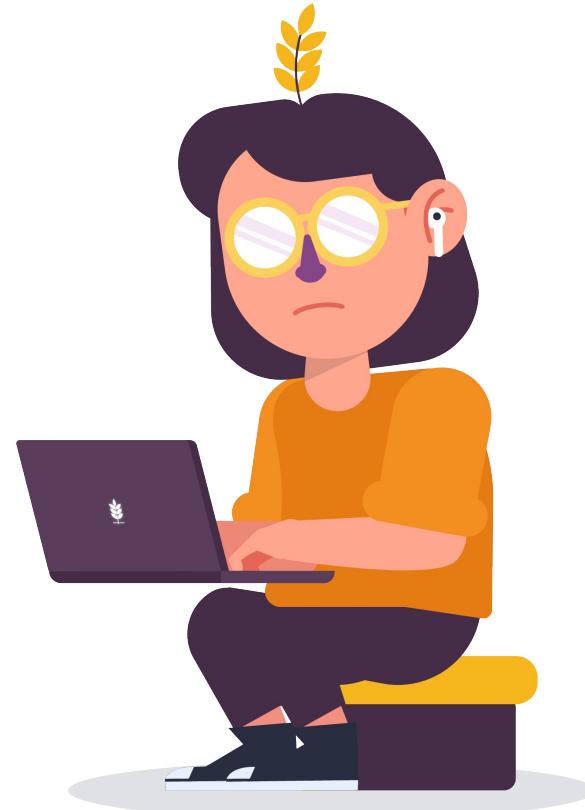


Nah, sekarang kalau objek-objek dari analogi tadi kita sambungkan dengan terminologi pemrograman, bisa kita petakan seperti tabel di bawah.

Objek di analogi	Objek di dalam pemrograman
Kamu	Komputer yang menjalankan program
Resep Nasgor	Program
Nasi, telur, bawang, sawi, minyak, penggorengan, pisau, kompor	Input
Nasi Goreng	Output

Berdasarkan terminologi singkat dari program diatas, kesimpulannya bahwa **pemrograman adalah sebuah aktivitas membuat sebuah program atau software.**

Nah, aktivitas ini bahasa kerennya ngoding, jelas dong, karena pemrograman tadi kan intinya kita ngedit teks, teks ini biasanya disebut sebagai code. Nah, teks atau kode ini biasanya ditulis dalam sebuah bahasa, yang biasa kita sebut sebagai bahasa pemrograman.



Nah, kita udah tau nih apa sih pemrograman itu, dan ternyata membuat program itu perlu yang namanya **bahasa pemrograman**, apa itu?



Di dalam bahasa ada yang namanya aktor (pengguna bahasa), yang mana aktor ini bertukar informasi melalui bahasa tersebut. Nah kalau kita ngomongin bahasa pemrograman, **aktor-aktornya adalah manusia dan komputer**. Yang mana aktor tersebut menggunakan bahasa pemrograman untuk **bertukar informasi**.





## Apa itu Bahasa Pemrograman?

Nah, terkhusus untuk bahasa pemrograman, transfer informasi antara manusia dengan komputer ini saat ini hanya tersedia dalam bentuk teks saja. Jadi, kita akan sering-sering ngoding hehe.



Kita udah tau kan sekarang bahasa pemrograman itu apa, nah sekarang cus lanjut bahas **Javascript** sebagai salah satu bahasa pemrograman!



## Apa itu Javascript?

Javascript adalah **sebuah bahasa pemrograman yang dipake untuk membuat sebuah halaman web, agar tampilan web lebih interaktif** (contoh: membuat animasi yang kompleks, menu popup dan sebagainya).

Dari definisi ini mungkin muncul pertanyaan

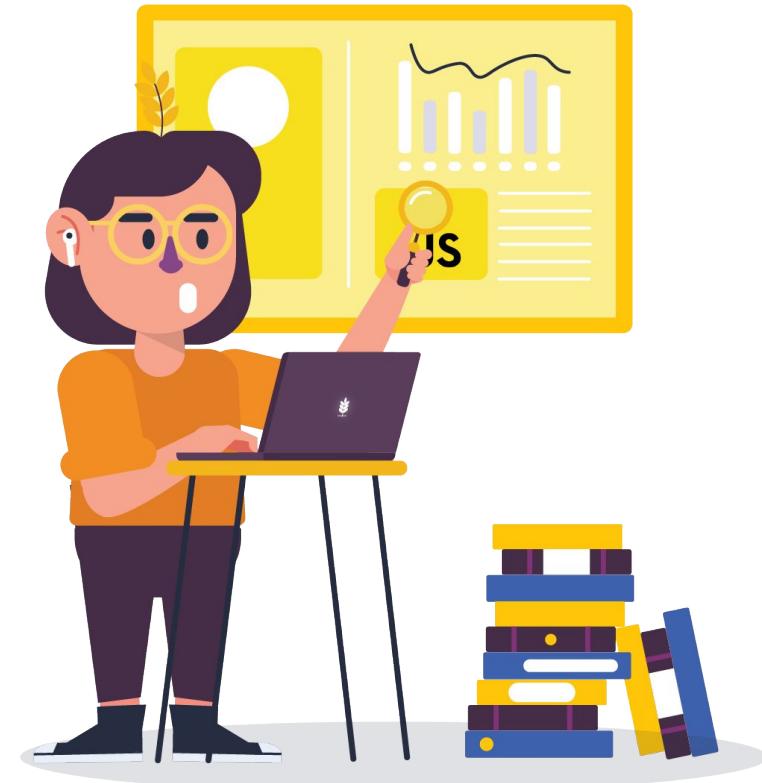
"Hmm... berarti javascript ini hanya bisa dipake di frontend aja ya?"

Coba kita cek next slide ya ~



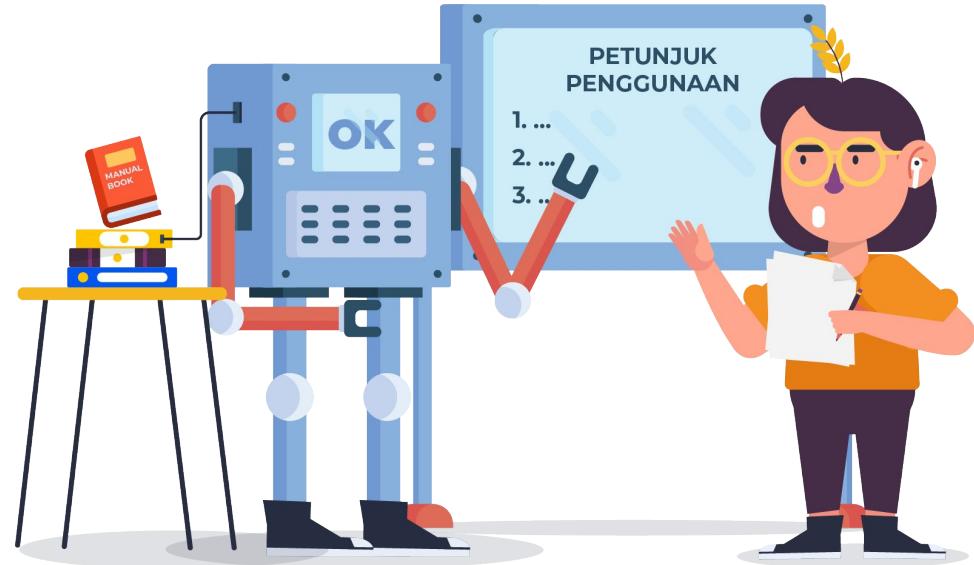
## Apa yang bisa Javascript lakukan?

Berkat software engineering dari Amerika Ryan Dahl, Javascript saat ini **ga cuma dipake untuk frontend aja, melainkan bisa dipake di backend** dengan menggunakan tools NodeJS.



Naahh~ Javascript ini adalah **interpreted language**, yang artinya ketika kita menjalankan sebuah aplikasi Javascript, kodennya akan diterjemahkan baris per baris.

Anggep aja, kita nerima prosedur dalam Bahasa Inggris, nah penerjemah bahasa Inggris kita akan menerjemahkan kalimat per kalimat (satu-satu)



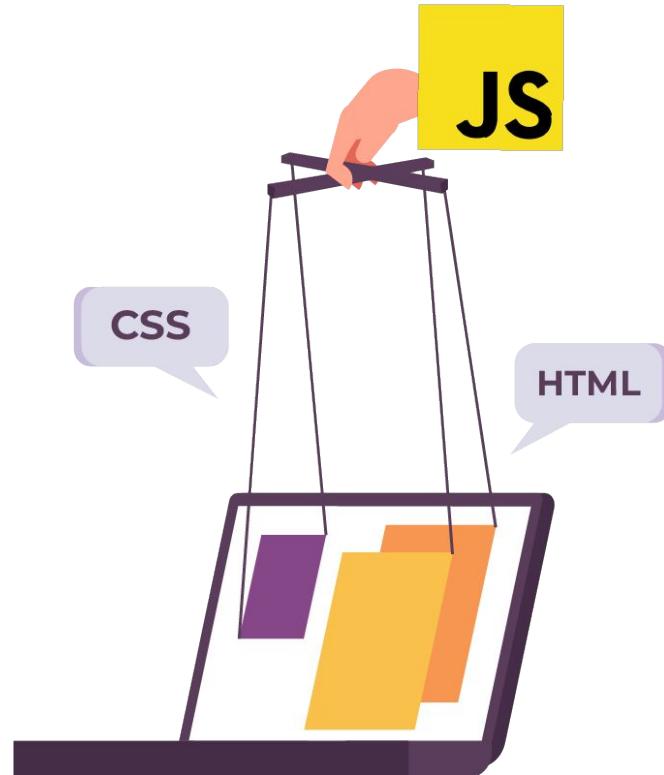
## Apa yang Javascript lakukan terhadap halaman web kita?

Javascript sering dipake untuk menampilkan suatu halaman secara dinamis.

Hmmm gimana ya maksudnya?

Chapter 1 kemarin kita udah belajar website dinamis itu apa yaa. Nah ~ HTML dan CSS akan jadi kerangka website kita. Sedangkan **Javascript bertugas mengisi dan mengatur ulang tampilanya.**

Biar lebih gampang bayanginnya kita bisa ambil contoh platform Youtube, skuy~

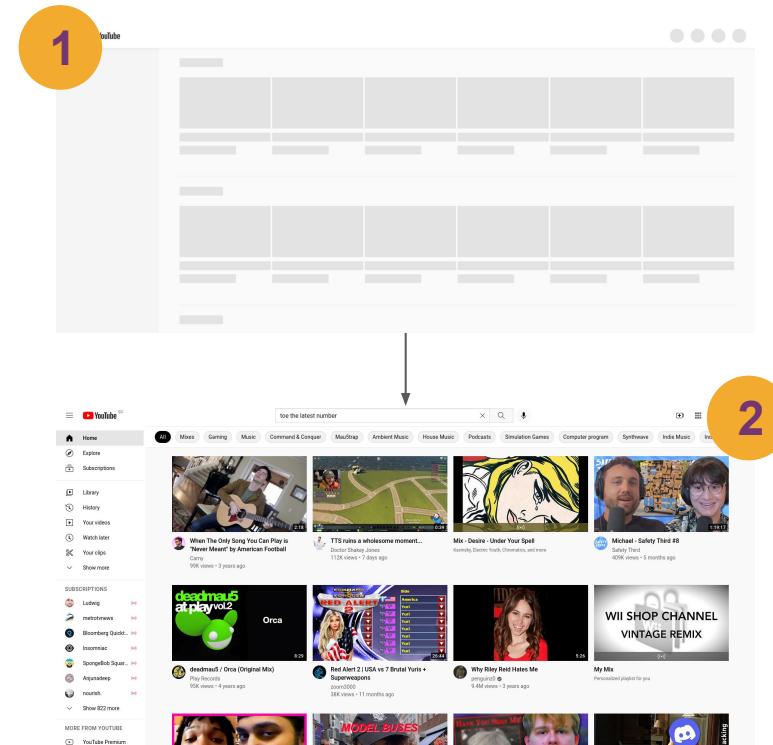


## Cek Youtube!

Ketika kita membuka Youtube untuk pertama kali, terlihat semacam kerangka dari situs Youtube (gambar 1). Nah ketika dalam fase tersebut browser hanya menampilkan HTML & CSS saja dan belum selesai menjalankan Javascript.

Nah, disini **Javascript berperan untuk mengambil data dari server Youtube**. Setelah diterima, data tersebut kemudian akan diolah menjadi tampilan menarik (gambar 2) yang sudah dapat dipake untuk melihat video dan sebagainya.

Javascript akan selalu dijalankan pada setiap halaman yang ada di Youtube, termasuk ketika kita mengetikkan sebuah keyword di dalam search bar. Jadi, lebih gampang kan bayanginnya?



## Terus cara nambahin Javascript di halaman web kita gimana?

Hmmm.. dari tadi kita ngomogin apa itu Javascript, gaada praktek-prakteknya, pasti pada bosen.

Yaudah skuy, langsung cobain aja gimana caranya nambahin Javascript di halaman web kita.

Ada **3 cara** nih untuk nambahin javascript di dalam halaman web kita, yaitu:

- Internal Javascript
- External Javascript
- Inline Javascript

Yuk, langsung bahas satu per satu~





```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Hello from Javascript</title>
  </head>
  <body>
    <script>
      console.log("Hello from Javascript!")
    </script>
  </body>
</html>
```

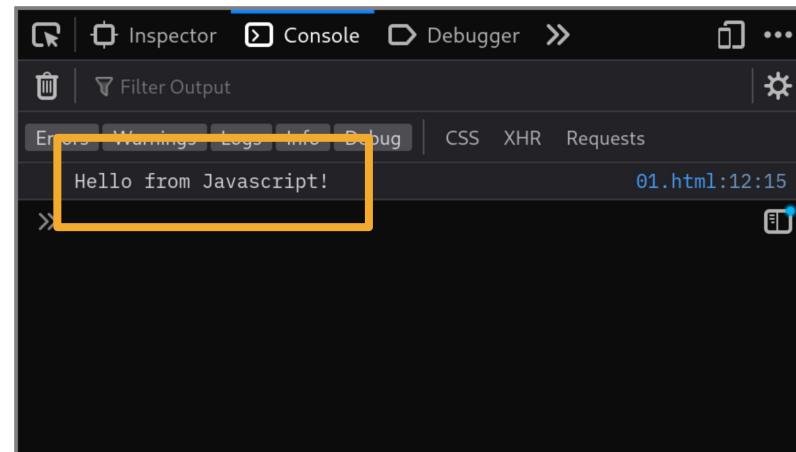
## Internal Javascript

Mirip dengan internal CSS, internal Javascript adalah **penambahan kode Javascript di dalam file HTML secara langsung**.

Javascript ini akan **ditulis di dalam tag bernama script**. Sebagian besar penambahan tag script ini ditulis di dalam tag body di dalam HTML.

Nahhh ~ kalau kamu udah buat internal Javascript di HTML, kamu bisa cek apakah coding Javascript kamu berjalan atau enggak di web yang dibuat. Caranya, kamu tinggal buka devtools pada browser, lalu lihat tab console. Kalau berhasil, maka tampilannya akan kaya gambar disamping ini!

Kamu bisa juga liat contoh tampilan dari code yang tadi kamu pelajari, [disini](#) ya



The screenshot shows the 'Console' tab of a browser's developer tools. The output area displays the message 'Hello from Javascript!' which is highlighted with a yellow rectangular box. The timestamp '01.html:12:15' is visible to the right of the message. The top navigation bar includes tabs for Inspector, Console, Debugger, and other developer tools.



## index.html

```
● ● ●  
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <title>Hello from Javascript</title>  
  </head>  
  <body>  
    <script src="../external.js"></script>  
  </body>  
</html>
```

## external.js

```
● ● ●  
  
console.log("Hello from external Javascript!");
```

## External Javascript

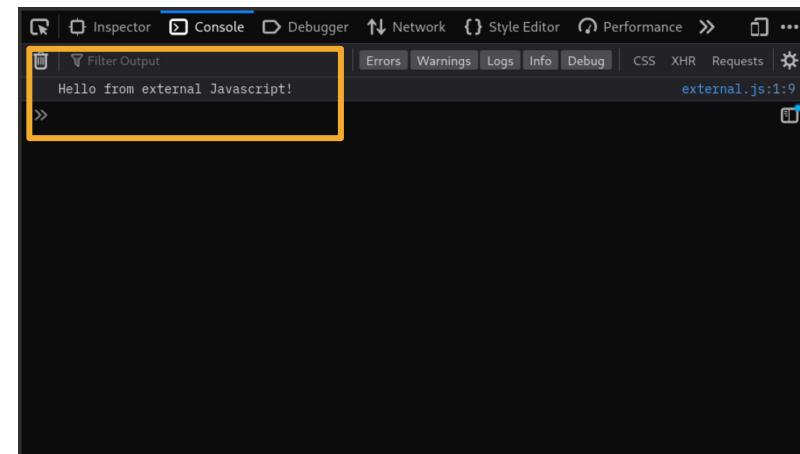
Eksternal Javascript adalah **file yang berisikan kode Javascript, kemudian kita impor melalui dokumen HTML.**

Berbeda dengan CSS yang menggunakan link untuk mengimpor. Dalam mengimpor file Javascript kita hanya perlu **menggunakan script dengan atribut src** yang mana nilai dari atribut tersebut berisi alamat dari file tersebut.

Yuk kita simak contoh disamping ini, pastikan semua file disimpan di folder yang sama yak!

Nahhh coding external Javascript juga bisa kita cek di console, nanti akan muncul kaya gambar disamping ya ~

Kamu bisa juga liat contoh tampilan dari code yang tadi kamu pelajari, [disini](#) ya



The screenshot shows a browser's developer tools interface with the 'Console' tab selected. The output window displays the message 'Hello from external Javascript!' on a single line. This line is highlighted with a yellow rectangular box. Above the output, there are tabs for 'Inspector', 'Console' (which is active), 'Debugger', 'Network', 'Style Editor', 'Performance', and others. Below the output, there are buttons for 'Errors', 'Warnings', 'Logs', 'Info', 'Debug', and file filters for 'CSS', 'XHR', and 'Requests'. The status bar at the bottom right shows 'external.js:1:9'.

## Inline Javascript

Yang terakhir adalah Inline Javascript. **Cara ini hanya berlaku jika dipake untuk memetakan fungsi yang akan dipanggil ketika suatu event dari suatu elemen dipanggil.**

Hmmm.. Kenapa ya?

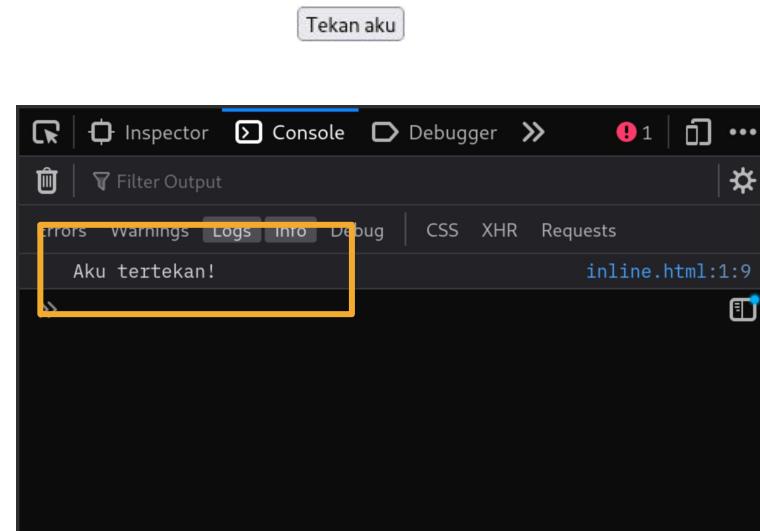
Karena jika kita menuliskan kode yang kompleks dengan menggunakan Inline Javascript, ini akan sulit untuk dibaca.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Hello from Javascript</title>
  </head>
  <body>
    <button onclick="console.log('Aku tertekan!')">Tekan aku</button>
  </body>
</html>
```



Tampilan dari coding inline Javascript tadi, bisa kamu lihat [disini](#) ya ~

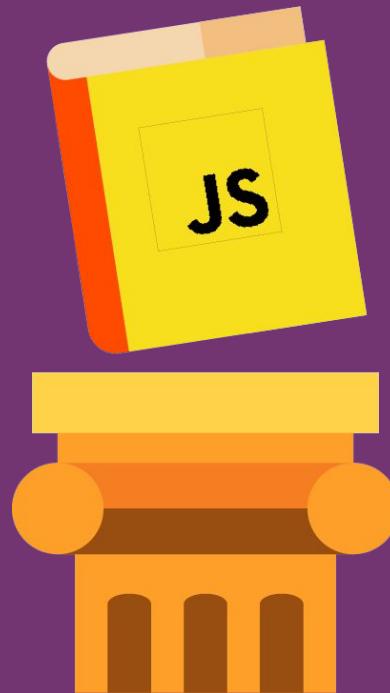
Dan kita juga bisa cek apakah Javascript sudah berjalan atau belum melalui console



The screenshot shows a browser's developer tools interface with the 'Console' tab selected. The output area displays the message "Aku tertekan!" followed by the file path "inline.html:1:9". The entire message line is highlighted with a yellow box. At the bottom right of the output area, there is a small blue edit icon.



Berhubung kita udah tau nih cara nambahin Javascript di halaman web kita, sekarang kita coba cari tahu nih, **kaidah bahasa Javascript** itu kayak gimana. Ya dong, karena dia sebuah bahasa pastinya ada grammar-nya hehe.



## Sintax dasar

JavaScript itu termasuk bahasa pemrograman dengan model data loosely typed. Artinya, **penentuan tipe datanya dinamis**.

Dinamis seperti apa, sih? Maksudnya, tipe data di dalam JavaScript bisa kita ubah-ubah, atau biasa dikenal dengan sebutan dynamic typing.





## Sintax dasar

Ini beda banget dengan Java. Kenapa? Karena, Java menerapkan static typing. Maksudnya, ketika membuat suatu data atau mendeklarasikan sebuah variabel, kita tidak boleh mengubahnya dengan tipe data yang berbeda.

### Java

#### Static typing:

```
String name;  
name = "John"  
name = 34;
```

Variables have types  
Values have types  
Variables cannot change type

### JavaScript

#### Dynamic typing:

```
var name;  
name = "John"  
name = 34;
```

Variables have no types  
Values have types  
Variables change type dynamically



Javascript itu case sensitive dan menggunakan Unicode character set. Jadi kita **bisa gunain emoji, atau mungkin aksara jawa untuk membuat sebuah variabel atau sebuah fungsi**. Kaya gambar disamping inii ~



```
const name = "Sabrina";
const name = "Sabrina";
console.log(name);
console.log(name);
```

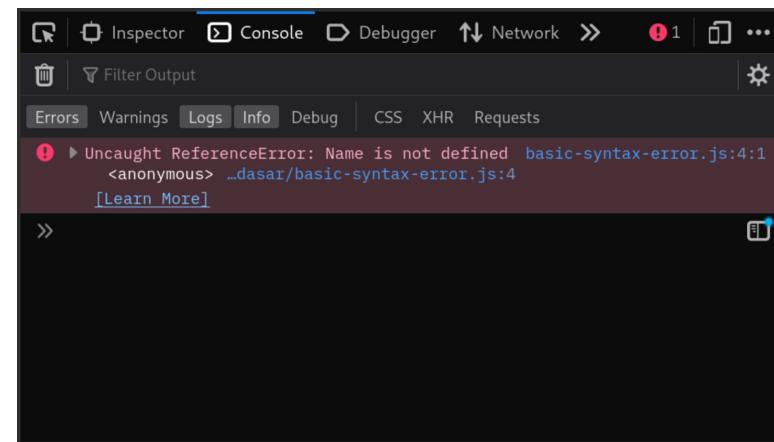


Tapi, karena case sensitive itu maka **name** itu tidak sama dengan **Name**.

Kode disamping memanggil variabel **Name** yang sebelumnya tidak didefinisikan, maka pasti output dari kode tersebut adalah error.



```
const name = "Sabrina";
const Name = "Sabrina";
console.log(Name); // Ini pasti error!
console.log(name);
```



Screenshot of a browser's developer tools Console tab. The tab bar includes Inspector, Console, Debugger, Network, and other developer tools. The Console tab is active. The output area shows an error message:

```
Uncaught ReferenceError: Name is not defined    basic-syntactical-error.js:4:1
<anonymous> ...dasar/basic-syntactical-error.js:4
[Learn More]
```



Kode-kode tadi adalah contoh perintah yang bisa kita tuliskan di dalam Javascript.

Nah, perintah di dalam Javascript itu disebut sebagai **statements dan dipisah oleh titik koma (;**). Titik koma tidak wajib ditulis jika setiap statement ditulis di masing-masing baris. Jika 1 baris terdapat 2 statemen, maka wajib dipisah dengan titik koma.



```
const name = "Sabrina";
const umang = "Sabrina";
console.log(name); console.log(umang);
```



## Comments

Kode itu bisa saja sangat rumit, bisa saja sangat sederhana, namun sebagai developer yang baik, kita dianjurkan untuk menulis comment di dalam kode kita.

Hmm.. Apa sih comment itu?

Comment adalah **teks yang kita tulis di dalam kode, namun tidak akan dianggap sebagai kode**, karena comment secara intensional akan dianggap sebagai catatan saja di dalam kode.

```
// deklarasiin kalo ada variabel bernama name yang nilainya sabrina
const name = "Sabrina";
// Aku suka mengomentari
const umang = "Sabrina";

/*
    Aku komentar yang lebih dari 1 baris
    Disini aku bebas mau speak up
    Pokoknya gitu deh
*/

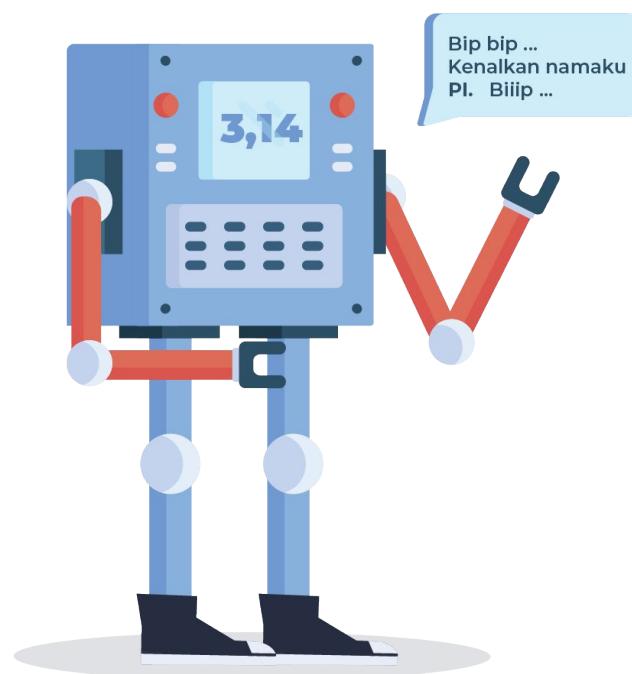
console.log(name); console.log(umang);
```

## Deklarasi

Deklarasi ini dipake buat **mendefinisikan sebuah variabel**. Variabel di dalam Javascript adalah sebuah nama yang kita berikan untuk mereferensikan sebuah nilai.

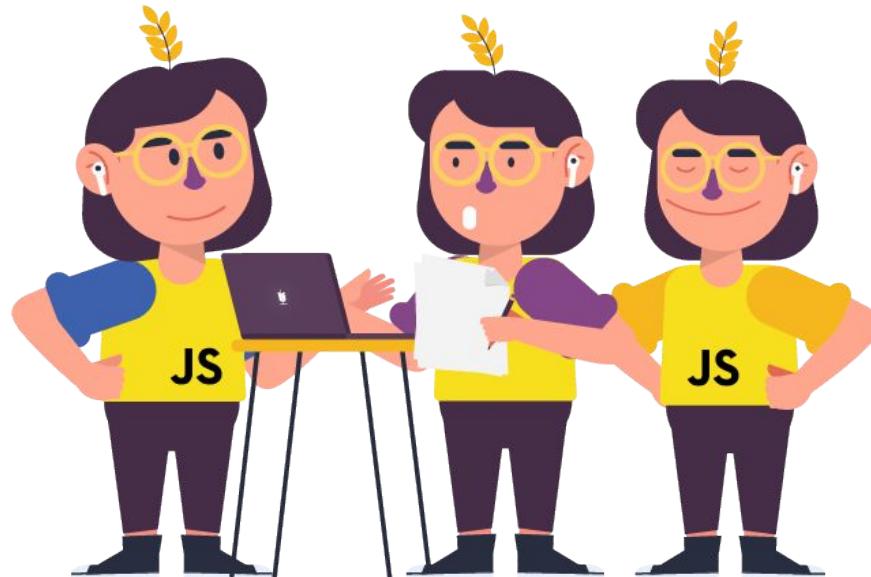
Contohnya gini, kita punya variabel bernama PI, yang mana kita pasti tau nilai dari PI itu adalah 3.14.

Jadi, variabel itu bagian dari struktur data yang digunakan untuk menyimpan informasi-informasi tertentu.



Terdapat 3 jenis variabel dalam mendeklarasikan Javascript. masing-masing cara memiliki sifat sendiri-sendiri yang direpresentasikan dengan menggunakan keyword sebelum nama variabel. Yaitu, **var**, **let**, dan **const**

Yuk, langsung aja lanjut buat melihat cara-caranya~





## Variabel Var

Bisa dibilang, variabel var merupakan variabel yang lawas. Tapi, penggunaannya masih dipertahankan oleh banyak programmer untuk menjaga kompatibilitas ke versi sebelumnya.

Variabel var ini punya fungsi dan cara penggunaan yang simpel. Tapi, kita perlu berhati-hati dalam menggunakan variabel ini karena ia memiliki beberapa problem.



```
/* Buat variabel var dengan cara deklarasi dulu */
var harga; // Declaration
harga = 1000; // Assignment

/* Buat variabel var yang langsung kita kasih nilai */
var harga = 1000
```



```
● ● ●

var diskon = 500 // Global scope
if (true){
  var diskon = 300 // Global scope
}
console.log(diskon)
// Output: 300
// karena var adalah global scope

/* Sebelum ada ES6, solusinya membuat function scope -> local scope */
var diskon = 500 // Global scope
function diskonScope(){
  var diskon = 300 // Local scope
  console.log(diskon) // Output: 300
}
diskonScope()
console.log(diskon) // Output: 500
```

## 1. Scope

Scope di dalam JavaScript bisa dikatakan sebagai cakupan atau jangkauan program yang ditandai dengan tanda kurung kurawal atau curly brackets `{...}`.

Umumnya, variabel dalam scope akan dianggap sebagai **local scope** agar tidak bisa dibaca oleh scope lain.

Tapi, jika kita menggunakan `var`, maka variabel akan berubah menjadi **global scope**. Artinya, ia masih bisa diakses meski berada di dalam scope.

Nah, biar nggak bingung dan repot, sebaiknya kita gunakan variabel **let** untuk fungsi scope.



```
var name; // Declaration
console.log(name) // Output: undefined

name ='Bot' // Assignment
console.log(name) // Output: Bot

var name ='Bot Sabrina' // Redeclared and Reassigned
console.log(name) // Output: Bot Sabrina
```

## 2. Reassigned dan Redeclared

Variabel var **dapat di-update nilainya (reassigned)** dan **dapat di deklarasi ulang namanya (redeclared)**.

Masalahnya, tidak akan ada pesan error sama sekali ketika terjadi duplikasi variabel. Hal ini cukup riskan jika kita melakukannya secara tidak sengaja. Bisa-bisa hasilnya berbeda dengan yang kita inginkan~



```
name = 'Mentor Sabrina' // Variabel di-assign duluan
var name; // Kemudian baru dideklarasikan
console.log(name) // Output: Mentor Sabrina

/* Di belakang layar terjadi hoisting */
var name;
name = 'Mentor Sabrina'
console.log(name) // Output: Mentor Sabrina
```

### 3. Hoisting

Hoisting ini ibarat **variabelnya “diangkat” atau “dipindahkan” ke atas**.

Maksudnya, gimana? Jika kita assign sebuah variabel var lebih dulu, JavaScript akan mendeklarasikan variabel tersebut ke atas atau mengangkatnya ke posisi atas di dalam scope. Hasilnya nggak akan error kok, tapi bakalan membuat kita bingung.

Jadi, **sebaiknya variabel dideklarasikan di awal sebelum di-assign**. Untuk lebih jelasnya, coba perhatikan contoh di samping!



## Variabel Let

Sejak kehadiran ES6/ES2015 (EcmaScript versi 6 atau EcmaScript yang rilis tahun 2015), JavaScript memperkenalkan variabel **let** dan **const**.

Kode di samping ini adalah contoh penulisan variabel let.

```
/* Buat variabel yang langsung kita kasih nilai */
let pesan = "Hello World";
console.log(pesan);

/* Buat banyak variabel sekaligus */
let nama = "Sabrina", // dipisahkan dengan koma
umur = 25, // dipisahkan dengan koma
jenisKelamin = "Perempuan";

console.log(nama); // Output: Sabrina
console.log(umur); // Output: 25
console.log(jenisKelamin); // Output: Perempuan/
```

*\*Catatan: EcmaScript adalah sebuah standardisasi scripting language (Javascript) yang dibuat oleh sebuah organisasi bernama European Computer Manufacturers Association (ECMA).*



```
let diskon = 500
if (true) { // Tanda awal scope
  let diskon = 300 // Hanya bisa diakses di dalam scope
  console.log(diskon) // Output: 300
} // Tanda akhir scope
console.log(diskon) // Output: 500
```

## 1. Scope

Jika kita menggunakan `let`, maka variabel akan bertindak menjadi **block scope**. Artinya, **variabel hanya bisa diakses di dalam scope, yang ditandai dalam sebuah kurung kurawal**.

Dengan menggunakan `let`, kita tidak perlu berurusan lagi dengan `function` untuk membuat local scope.



```
● ● ●  
let name; // Declaration  
console.log(name) // Output: undefined  
name ='Bot' // Assignment  
console.log(name) // Output: Bot  
name ='Bot Sabrina' // Reassigned  
console.log(name) // Output: Bot Sabrina  
let name ='Mentor Sabrina' // Can not redeclared  
console.log(name)  
  
// Output: SyntaxError: Identifier 'name' has already been declared
```

## 2. Reassigned dan Redeclared

Variabel let **dapat di-update nilainya (reassigned)**, tapi **tidak bisa di deklarasi ulang namanya (redeclared)**. Jadi, akan ada pesan error ketika terjadi duplikasi variabel.

Tentu hal ini akan sangat membantu jika kita nggak sengaja melakukan deklarasi ulang variabel.



```
● ● ●  
  
// Contoh pertama di codesandbox.io  
name = "Bot"; // Assignment  
let name; // Declaration  
console.log(name); // Output: Bot  
// Contoh pertama di console web browser  
name = "Bot"; // Assignment  
let name; // Declaration  
console.log(name);  
// Output: Cannot access 'name' before initialization  
  
/* Setelah revisi kode */  
let name = "Bot"; // Initialization  
console.log(name); // Output: Bot
```

### 3. Hoisting

Sebenarnya, variabel let di JavaScript juga bisa hoisted.

Menariknya, jika dijalankan pada console di web browser, ia akan menghasilkan error. Kenapa? Karena engine JavaScript nggak bisa mengakses nama variabel sebelum dideklarasikan atau diinisialisasi.

Kondisi ini biasanya disebut dengan istilah **Temporal Dead Zone (TDZ)**.



```
● ● ●

// Contoh Kedua
let message = "Hello"
function greetings(){
  console.log(message)
  let message = "Hello World!"
}
greetings()
// Output: Uncaught ReferenceError: Cannot access 'message' before initialization

/* Setelah kode direvisi */
let message = "Hello"
function greetings(){
  let message = "Hello World!"
  console.log(message)
}
greetings()
// Output: Hello World!
```

Nah, **untuk contoh kedua**, penerapan hoisting variabel let di dalam function juga akan menghasilkan error.

Karena, sebenarnya di belakang layar JavaScript mendeklarasikan variabel tersebut ke atas tanpa value, sehingga akan menghasilkan **undefined**.

Namun, ini justru berguna bagi kita. Agar kita merevisi kode sesuai kaidah yang baik.

Oleh karena itu, sejak awal variabel sudah harus dideklarasikan atau diinisialisasi dengan sebuah value.



## Variabel Const

Variabel const berarti, **nilai atau datanya tidak akan berubah dalam kondisi apapun**. Makanya dinamakan const (**konstan**).

Dalam mendefinisikan variabel konstan, kita diwajibkan untuk langsung memasukkan nilai dari variabel tersebut.



```
const bumi = "bulat";
const aku = "tamvan";
const pi = 3.14;
```



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

/* Ketika kita ingin memanggil warna
   Kita cuma perlu panggil variabelnya saja */
let warnaBaju = WARNA_MERAH;
console.log(warnaBaju);
```

## Uppercase Constants

Kita dianjurkan untuk menggunakan variabel konstan dalam beberapa kasus. Contohnya, penggunaan **alias**, agar kita lebih mudah mengingatnya.

Variabel konstan seperti itu biasanya diberi nama dengan menggunakan **garis bawah ( \_ )** untuk spasi.

Mari kita buat konstanta untuk warna dalam format yang disebut "web" (heksadesimal).

Kelebihan menggunakan variabel konstan adalah:

- **WARNA\_MERAH** lebih gampang untuk diingat, lebih bermakna, dan mudah dibayangkan daripada kode warnanya yaitu **#F00**
- Menghindari terjadinya typo.



```
const diskon = 500
if (true) { // Tanda awal scope
  const diskon = 300 // Hanya bisa diakses di dalam scope
  console.log(diskon) // Output: 300
} // Tanda akhir scope
console.log(diskon) // Output: 500
```

## 1. Scope

Seperti halnya variabel **let**, jika kita menggunakan **const**, maka variabel yang ada akan bertindak menjadi block scope. Artinya, variabel hanya bisa diakses di dalam scope, yang ditandai dalam sebuah kurung kurawal `{...}`.

Dengan menggunakan `const`, kita tidak perlu berurusan lagi dengan `function` untuk membuat local scope.



```
/* Kita gak bisa ubah nilai bumi */
const bumi = "bulat";
bumi = "datar";
// Output: TypeError: invalid assignment to const `bumi`

/* Kita juga gak bisa deklarasi ulang */
const bumi = "datar";
// Output: SyntaxError:Identifier 'bumi' has already been declared
```

## 2. Reassigned dan Redeclared

Karena ini konstanta, maka kita nggak boleh mengubah nilai dari variabel tersebut. Dengan kata lain, kita gak bisa me-reassigned. Kenapa? karena variabel ini sifatnya immutable. Berarti, value-nya nggak bisa diubah.

Kita juga nggak bisa me-redeclared atau mendeklarasikan ulang, lho! Jangan sampai lupa ya~



```
/* Object dengan variabel const masih bisa kita ubah property-nya */
cc const obj = { id:1, name:'Sabrina'}
ok obj.location="Jakarta"
cc console.log(obj) // Output: { id:1, name:'Sabrina', location:'Jakarta'}
// Tapi, kita gak bisa reassigned
ok obj={} // Output: TypeError: Assignment to constant variable.

/* Array dengan variabel const masih bisa kita ubah property-nya */
cc const arr = [1,2,3,4]
arr.push(5)
cc console.log(arr) // Output: [1,2,3,4,5]
// Tapi, kita gak bisa reassigned
ar arr=[] // Output: TypeError: Assignment to constant variable.
```

**Eitss, ini nggak berlaku untuk object dan array, ya!**

Variabel const memang nggak bisa kita reassigned dan redeclared.

Tapi, property dalam object masih bisa diubah. Ini karena object dan array dalam JavaScript bersifat mutable, yang berarti value-nya bisa kita ubah.



```
name = "Mentor Sabrina"; // Assignment
const name; // Declaration
console.log(name);
// Output: Uncaught SyntaxError: Missing initializer in const declaration
```

### 3. Hoisting

Hoisting di variabel const juga sama uniknya seperti variabel let. Karena, engine JavaScript nggak bisa mengakses nama variabel sebelum dideklarasikan atau diinisialisasi. Meski begitu, variabel const akan langsung menunjukkan error.

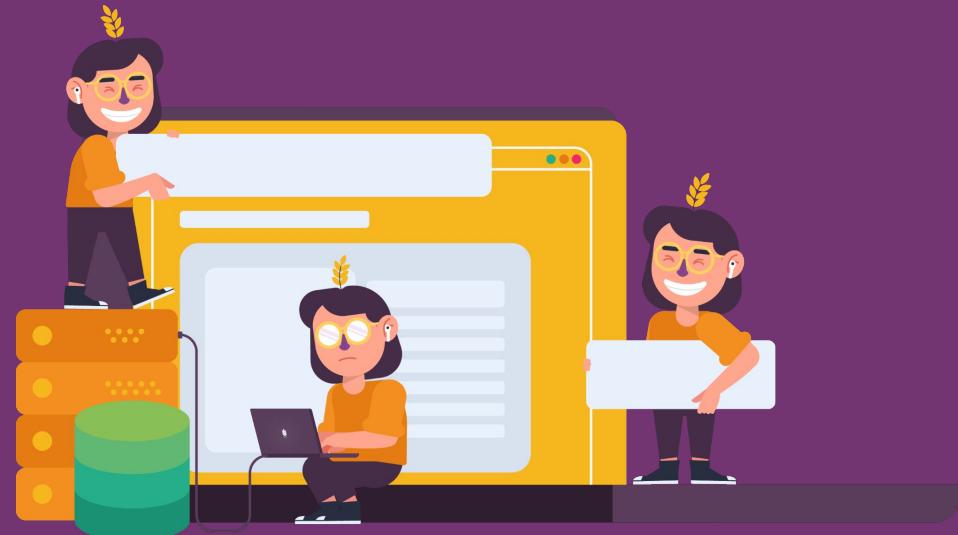
Jadi, kita harus menginisialisasi nilai dari sebuah variabel di awal ketika mau menggunakan variabel const.



Gimana, udah tau kan perbedaan dari tiga jenis variabel di JavaScript dan kapan sebaiknya digunakan? Kalo masih bingung, coba cek tabel di bawah!

	<b>var</b>	<b>const</b>	<b>let</b>
scope	global or local	block	block
redeclare?	<b>yes</b>	<b>no</b>	<b>no</b>
reassign?	<b>yes</b>	<b>no</b>	<b>yes</b>
hoisted?	<b>yes</b>	<b>no</b>	<b>no</b>

Setelah kamu tau variabel di Javascript, sekarang saatnya kita bahas bagaimana informasi direpresentasikan di Javascript, cekidot ~



Masih ingat analogi masak nasgor tadi? Di dalam prosedur tadi, terdapat beberapa data, yaitu : resep nasgor, nasi putih, telur, bawang, dan sawi.

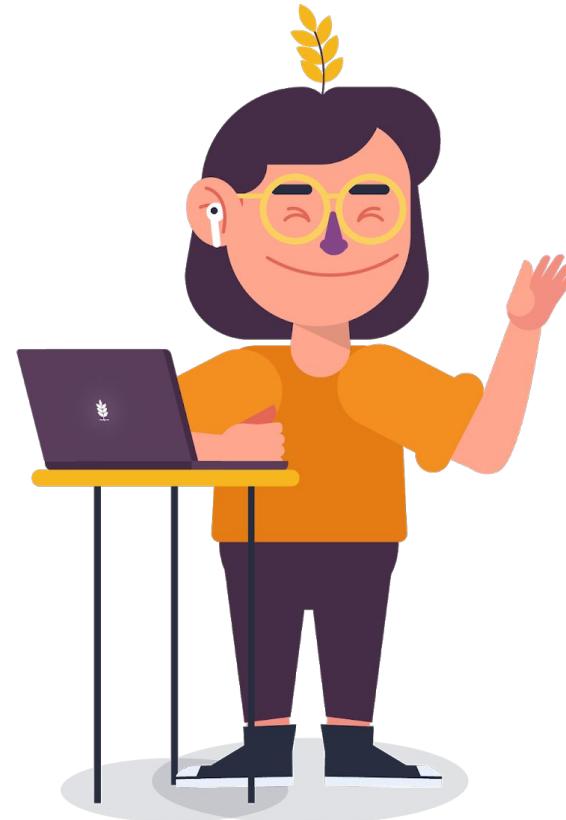
Nah, data-data itu bisa kita kelompokkan nih jadi peralatan dan bahan. Penggorengan dan pisau itu adalah peralatan, lalu nasi putih dan telur adalah bahan. Nah, bahan dan peralatan kalo kita sambungin ke bahasa pemrograman itulah yang kita sebut sebagai **tipe data**.





### Terus, tipe data ini gunanya untuk apa sih?

Setiap tipe data punya sifatnya masing-masing, dan akan dipake tergantung dengan kebutuhan. Contoh, jika kita punya data berupa angka, otomatis kita dapat menggunakan angka tersebut dalam operasi aritmatik. Kalau kita punya data berupa teks atau string, data itu akan kita gunakan untuk membentuk teks yang lain dan sebagainya.



Nah di dalam Javascript itu ada **beberapa tipe data** yang wajib kita ketahui, yaitu:

- **String**
- **Number**
- **Boolean**
- **Null**
- **Undefined**
- **Array**
- **Object**

Yuk kita bahas satu per satu~



## String

String adalah **teks**. Dalam penulisan string, teks tersebut akan diapit oleh tanda kutip, baik itu kutip satu maupun kutip dua.

Nah, string ini biasanya dipake buat apa aja sih?

Ya, cuma dipake **buat nyimpen teks aja**, contohnya nama, alamat, nomor hp, username, url, dan sebagainya.

```
const name = "Sabrina"; // Ini adalah string
let address = "Bumi Serpong Damai"; // Ini juga string

console.log(name);
console.log(address);
console.log('The Breeze') // Ini juga string pake kutip satu
```

## Number

Kalo kita terjemahkan secara harfiah, number itu adalah angka. Angka kan ada banyak tuh jenisnya, ada desimal, ada cacah, dan sebagainya.

Tapi kalo di Javascript, hanya ada dua jenis angka, yaitu **cacah (Integer)**, dan **desimal (Float)**.

Number biasa dipake dalam operasi aritmatik, dan bisa dipake sebagai pembanding juga.



```
const x = 10; // Integer
let y = 100; // Integer
let z = 1.1; // Float

console.log(x + y); // 110
```

## Boolean

Tipe data ini adalah tipe data paling simpel, karena cuma ada dua nilai.

Yaitu, **BENAR** atau **SALAH**, tipe data ini biasanya dipake untuk menandai suatu fakta itu bener atau enggak.



```
const isRaining = false;  
const isExhausted = true;  
let shouldRunFromReality = true;
```

## Null

Null adalah **sebuah tipe data yang kosong**. Null ini biasanya dipake untuk menandai bahwa sesuatu itu sedang kosong, dan nantinya akan dapat diisi dengan tipe data yang lain.

Maka dari itu jarang sekali ada orang menggunakan const untuk mendeklarasikan variabel yang bernilai null karena pastinya tidak berguna sama sekali ya ~



```
let seseorangDiHatiku = null;  
seseorangDiHatiku = "Kamu";
```

## Undefined

Undefined ini adalah **sebuah nilai yang menandakan bahwa sesuatu belum didefinisikan**. Sama dengan null, tidak ada developer yang menggunakan const untuk mendeklarasikan variabel yang bernilai undefined.

Secara default, ketika kita mendeklarasikan sebuah variabel dengan menggunakan var dan let tanpa memberikannya nilai, variabel tersebut akan bernilai undefined..



```
let tujuanHidupku;  
let artiKebahagiaan = undefined;  
  
console.log(tujuanHidupku); // undefined  
console.log(artiKebahagiaan); // undefined
```



## Array

Tipe data ini dipake untuk **menyimpan kumpulan dari data**.

Contoh, kumpulan angka, nama-nama kucing satu kelurahan, dan sebagainya. Dan dalam penamaan variabel yang menyimpan array, sangat dianjurkan untuk menggunakan kata jamak, contoh: names.

Setiap data di dalam array diidentifikasi dengan yang namanya indeks. Indeks ini dimulai dari 0, jadi data paling depan di dalam array itu pasti indeksnya 0, dan data selanjutnya indeksnya adalah 1 dan seterusnya. Yuk simak contoh kode disamping ini biar makin kebayang.

```
const catNames = ["Oyen", "Bob"];  
  
// Kalo aku mau memanggil "Oyen" maka aku akan menggunakan indeks 0,  
// karena "Oyen" adalah data terdepan dari array catNames  
console.log(catNames[0]); // Oyen  
  
// Kalo aku mau memanggil "Bob" maka aku akan menggunakan indeks 1,  
// karena "Bob" adalah data kedua dari array catNames  
console.log(catNames[1]); // Bob
```



```
let favouriteNumbers = [1, 2, 10];
favouriteNumbers[0] = 100;
console.log(favouriteNumbers) // [100, 2, 10];
```

Kita juga bisa nih merubah nilai data dari dalam sebuah array, dengan memanggil indeks dan melakukan assignment.



## Object

Object mirip dengan array, kalau array mengidentifikasi data di dalamnya menggunakan indeks, **object menggunakan key sebagai identifer-nya.**

```
const person = {  
  name: "Sabrina",  
  age: 21,  
  isMarried: false,  
  pets: [  
    {  
      name: "Oyen",  
      speciesName: "Cat"  
    },  
    {  
      name: "Bob",  
      speciesName: "Cat"  
    },  
  ]  
}
```



```
console.log(person.name); // Sabrina  
console.log(person["age"]); // 21;  
console.log(person.pets[0].name) // Oyen
```

Dari data tadi, yang merupakan key adalah **name**, **age**, **isMarried**, dan **pets**. Yang mana masing-masing key memiliki nilai yang merupakan sebuah tipe data juga.

Nah untuk memanggil nilai dari sebuah object, kita bisa gunakan **square bracket seperti array**, atau kita bisa menggunakan **dot notation (.)**

# Saatnya kita Quiz!





## 1. Manakah pernyataan yang tidak benar tentang pemrograman?

- A. Pemrograman adalah sebuah aktifitas menulis sebuah program.
- B. Inti dari pemrograman adalah menulis teks atau kode dalam suatu bahasa pemrograman tertentu.
- C. Mendesain suatu halaman web merupakan bagian dari aktifitas pemrograman.



### 1. Manakah pernyataan yang tidak benar tentang pemrograman?

- A. Pemrograman adalah sebuah aktifitas menulis sebuah program.
- B. Inti dari pemrograman adalah menulis teks atau kode dalam suatu bahasa pemrograman tertentu.
- C. Mendesain suatu halaman web merupakan bagian dari aktifitas pemrograman.

Mendesain suatu halaman web adalah aktifitas yang tidak termasuk aktifitas pemrograman. Kenapa, karena mendesain suatu halaman web itu masuk ke aktifitas desain.



## 2. Pernyataan manakah yang menggambarkan keterkaitan antara Pemrograman dan Bahasa Pemrograman?

- A. Untuk membuat sebuah program, bahasa pemrograman dibutuhkan dalam mendefinisikan program tersebut.
- B. Bahasa pemrograman merupakan produk dari sebuah program itu sendiri.
- C. Program dan bahasa pemrograman tidak terkait sama sekali



## 2. Pernyataan manakah yang menggambarkan keterkaitan antara Pemrograman dan Bahasa Pemrograman?

- A. Untuk membuat sebuah program, bahasa pemrograman dibutuhkan dalam mendefinisikan program tersebut.
- B. Bahasa pemrograman merupakan produk dari sebuah program itu sendiri.
- C. Program dan bahasa pemrograman tidak terkait sama sekali

Untuk membuat sebuah program, diperlukan bahasa pemrograman entah seabstrak apapun itu. Meskipun saat ini udah banyak tools, tapi under the hood mereka tetap menulis program dalam sebuah bahasa pemrograman.



- 3. Tipe data apa yang cocok untuk menyimpan sebuah dokumen yang berbentuk teks (HTML)?**
- A. String
  - B. Number
  - C. Boolean



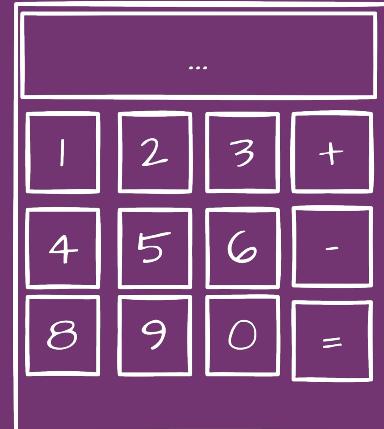
### 3. Tipe data apa yang cocok untuk menyimpan sebuah dokumen yang berbentuk teks (HTML)?

- A. String
- B. Number
- C. Boolean

Meskipun HTML ini biasanya berbentuk file, tapi isi file tersebut adalah teks yang mana apabila kita ingin menyimpan teks tersebut di dalam Javascript, kita hanya bisa menggunakan string saja.

#### 4. Tipe data yang digunakan untuk inputan dari kalkulator adalah

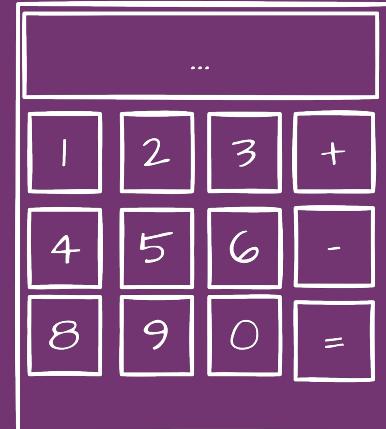
- A. String
- B. Number
- C. Boolean



#### 4. Tipe data yang digunakan untuk inputan dari kalkulator adalah

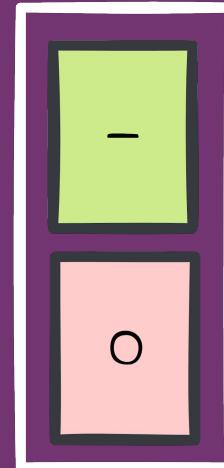
- A. String
- B. Number
- C. Boolean

Karena input dari akan digunakan untuk operasi aritmatik maka dari itu input dari kalkulator itu harus dibuat menjadi number.



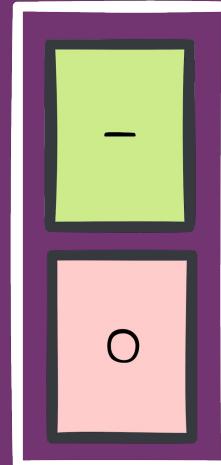
**5. Tipe data yang menggambarkan status "ya" dan "tidak" (seperti saklar) adalah data**

- A. String
- B. Number
- C. Boolean



## 5. Tipe data yang menggambarkan status "ya" dan "tidak" (seperti saklar) adalah data

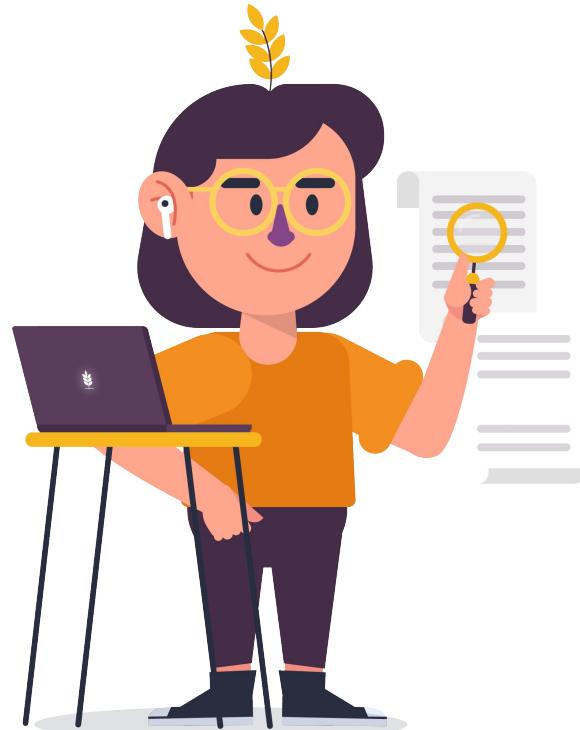
- A. String
- B. Number
- C. Boolean



Nilai dari saklar itu cuma ada dua, nyala atau tidak, yang mana bila kita sambungkan ke tipe data javascript, maka yang punya dua nilai itu cuma Boolean aja.

## Referensi, buat kamu yang masih kepo~

- [MDN - Javascript Guide \(Sumber Utama\)](#)
- [FikriRNurhidayat - Javascript Dasar](#)





Nah itu tadi penjabaran singkat terkait dasar-dasar dari Javascript, jangan lupa eksplorasi lebih jauh lagi dengan perbanyak latihan.

Mirip-mirip lah kayak kalian belajar Bahasa Inggris dulu, kalo dibiasain pasti lama-lama lancar kok.

Selanjutnya, kita bakal bahas tentang **Operator & Expression** di Topik 2 .



# Terima Kasih!



Next Topic

loading...