BISA NGODING PAKAI JAVASCRIPT

part 1

Belajar ngoding menggunakan javascript dari dasar

Bisa Ngoding Pakai JavaScript - part1

“Belajar ngoding menggunakan javascript dari dasar”

Daftar Isi

Persiapan Sebelum Ngoding

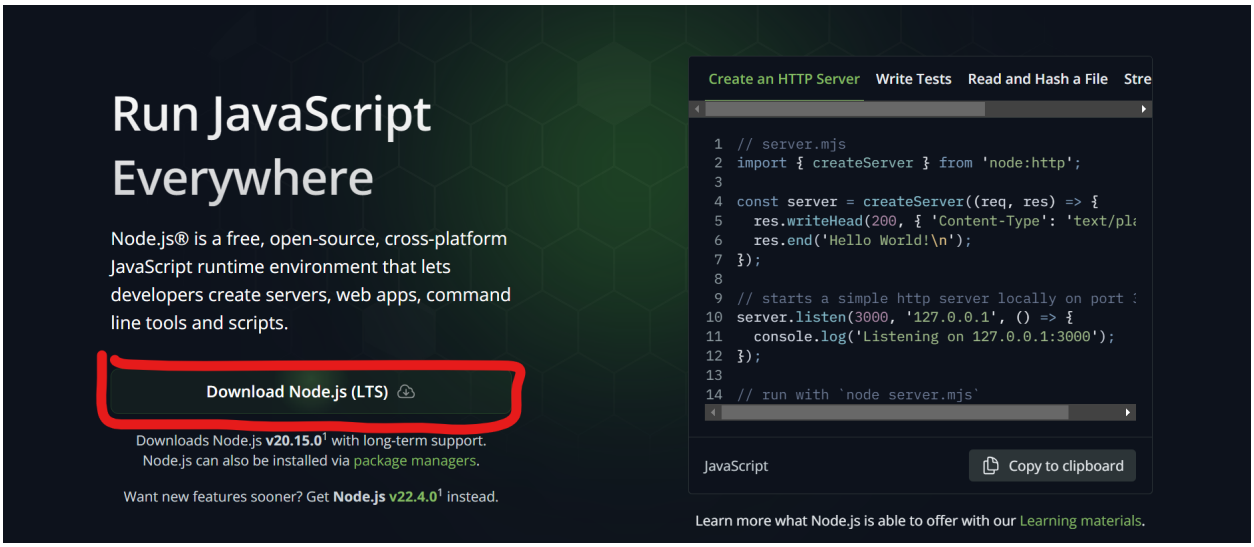
Download Code Editor Favoritmu

Pilih code editor favorit kalian, ada banyak sekali code editor yang populer seperti Sublime Text, Notepad++, Atom, Visual Studio Code. Kalau Saya menggunakan Visual Studio Code. Silahkan download disini.

Download Node Js

Untuk menjalankan code javascript, kita perlu node js. Silahkan download disini.

Download yang versi LTS ya 😁.



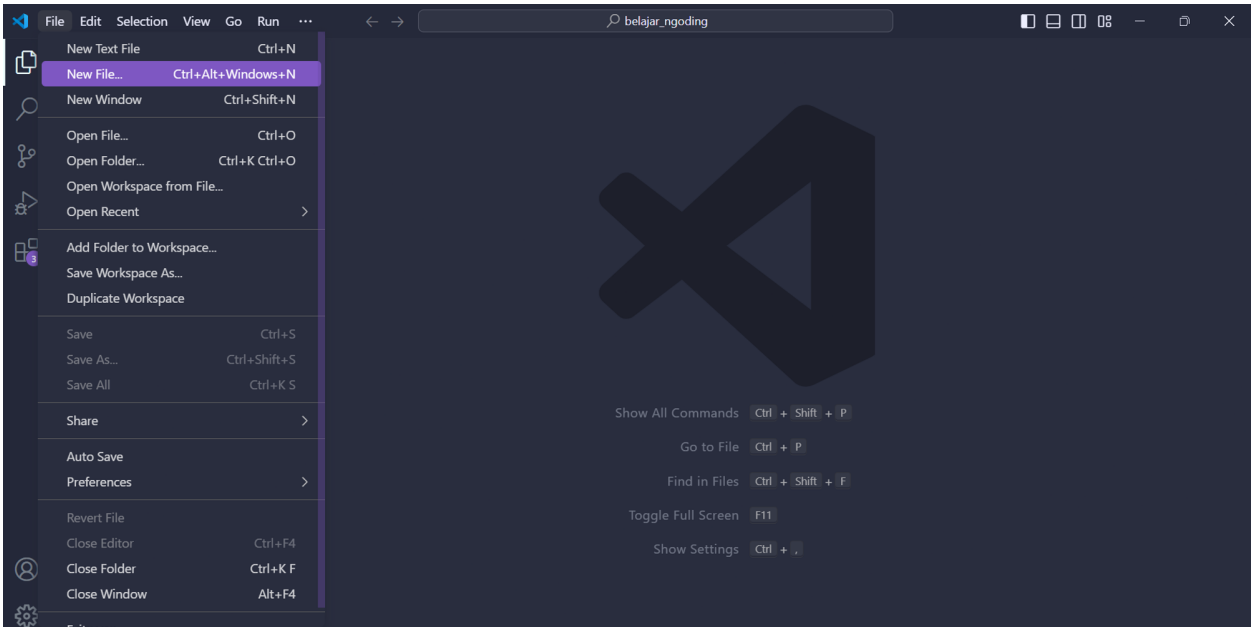
Terminal

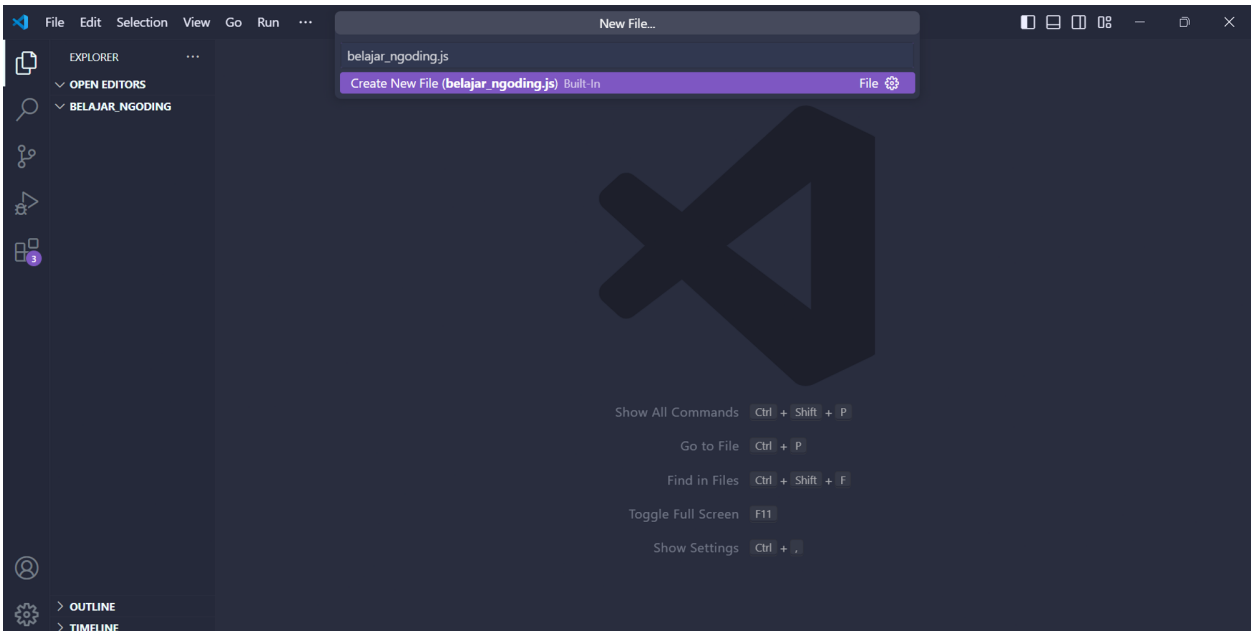
Ada banyak terminal bawaan dari PC masing masing yaitu Windows Terminal, CMD, Windows Power Shell, atau terminal bawaan dari Visual Studio Code. Saya merekomendasikan menggunakan terminal bawaan VS code agar lebih mudah dan tidak berpindah pindah aplikasi.

Membuat Folder Pertama di VS Code

- Buat folder di desktop, klik kanan > New Folder.
- Klik kanan folder yang telah dibuat, lalu open with Vs Code.

- Setelah masuk VS Code, klik File > New File... > beri nama file sesuai keinginan
- *Jangan lupa tambahkan exstention .js* karena kita akan ngoding menggunakan javascript.



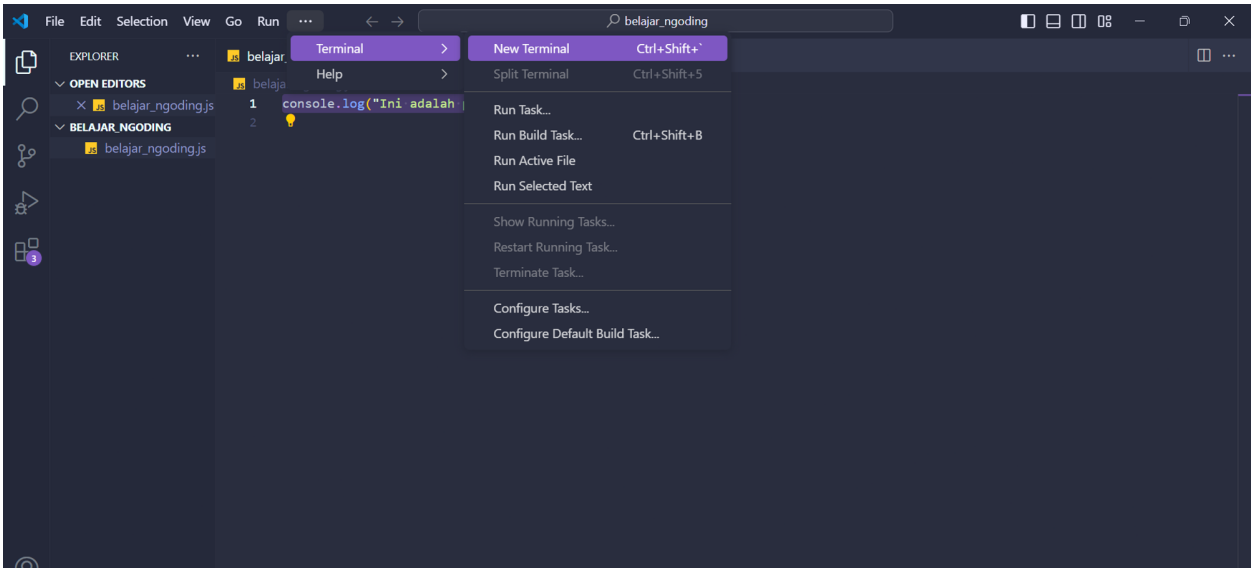


- Save file yang sudah dibuat

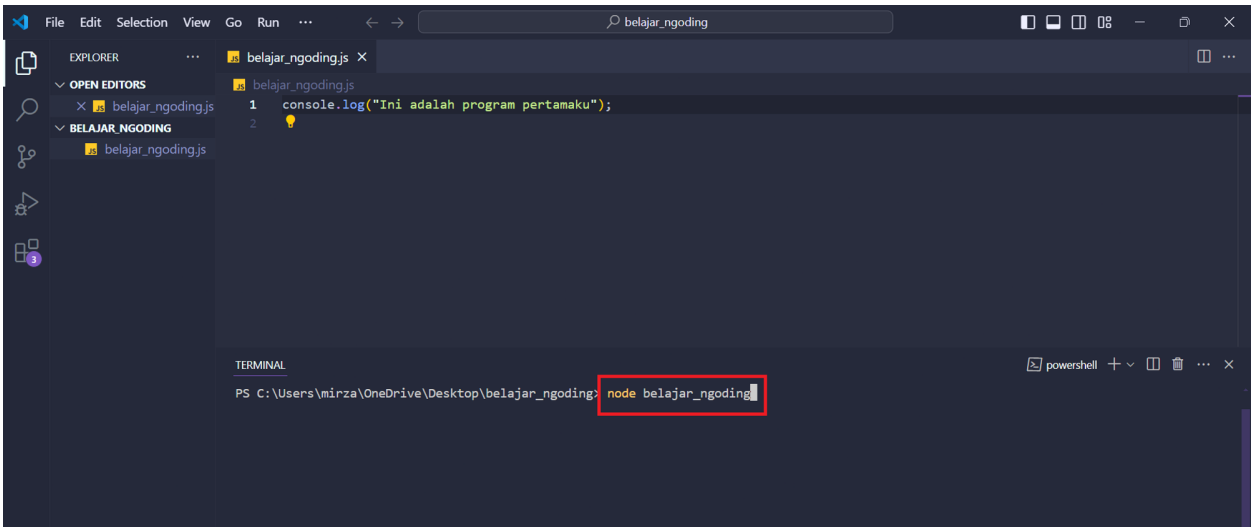
Mulai Ngoding

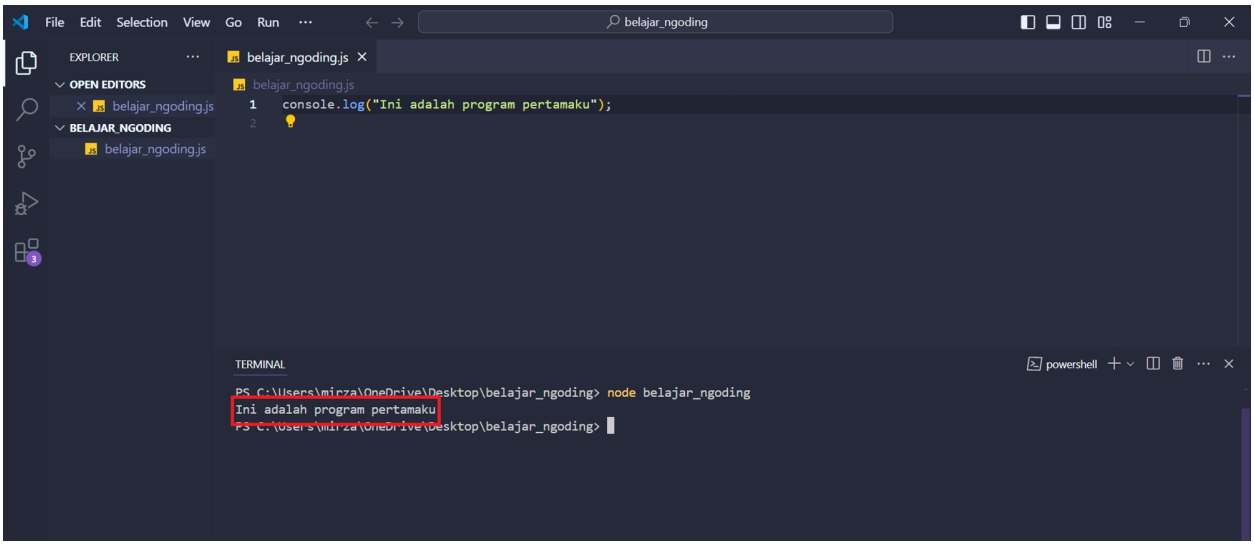
Membuat Program Pertamamu

- Kita akan membuat program sederhana pertama dengan menampilkan output kode kita di terminal.
- Ketik `console.log("Ini adalah program pertamaku");`
- Buka terminal, klik tanda ... > Terminal > New Terminal (Ctrl + Shift + `)



- Ketik node nama file kalian, misal node belajar_ngoding
- Hasilnya bisa kita lihat di terminal, *ini adalah program pertamaku*.





Membuat Komentar di JavaScript

- ketik `//` di awal kalimat.
- Atau tekan `Ctrl + /` pada baris kode tersebut.

```
//Ini adalah sebuah komentar
```

- untuk multi line komentar menggunakan tanda `/* ... */`.

```
/*Ini adalah
multi line
komentar*/
```

Variabel

- Variabel adalah tempat menyimpan data atau nilai.

- fungsinya ketika akan menampilkan code nya, hanya perlu menuliskan nama variabelnya. Code akan lebih rapi dan mudah terbaca.

```
var nama = "Budi Kusuma";
console.log(nama); //Hasilnya: Budi Kusuma
```

- Analogi sederhananya seperti kita memasukkan baju di dalam loker/lemari. Baju sebagai data dan loker sebagai variabel.
- Di Javascript, membuat variabel dapat menggunakan kata kunci `var`, `let`, dan `const`

```
var nama = "Budi Kusuma";
```

- Dari code di atas, pembuatan variabel menggunakan kata kunci `var`. Nama variabel adalah ***nama***. Budi Kusuma adalah ***data atau nilai*** dari variabel tersebut.

Perbedaan var, let, dan const

Fitur	`var`	`let`	`const`
Scope	Function scope	Block scope	Block scope
Hoisting	Yes (declaration only)	Yes (declaration only)	Yes (declaration only)
Reassign	Yes	Yes	No (immutable binding)
Redeclare	Yes	No	No
Initialization	Can be declared without initialization	Needs to be initialized before use	Needs to be initialized before use
Global object property	Yes	No	No

Penjelasan:

1. **Scope:**
    - `var`: Memiliki scope pada level function, artinya variabel hanya terbatas pada fungsi tempat ia dideklarasikan.
    - `let`: Memiliki scope pada level blok (blok `{ ... }`), seperti `if`, `for`, atau `while` (nanti kita akan belajar), yang membuat variabel hanya terlihat di dalam blok tersebut.
    - `const`: Seperti `let`, memiliki scope pada level blok, tetapi variabel yang dideklarasikan dengan `const` tidak bisa diubah nilai referensinya.
2. **Hoisting:**
    - `var`, `let`, dan `const` semuanya hoisted (ditarik ke atas), tetapi yang ditarik hanya deklarasinya, bukan inisialisasinya. Misalnya, `console.log(x); var x = 5;` akan menghasilkan `undefined`, tetapi tidak error.
3. **Reassign:**
    - `var` dan `let` memungkinkan untuk direassign (diubah nilainya setelah dideklarasikan).

```
var x = 3;
var x = 5;

console.log(x); //hasilnya: 5
```

```
let x = 3;
x = 5;

console.log(x); //hasilnya: 5
```

    - `const` tidak dapat direassign setelah inisialisasi. Namun, untuk objek dan array, isi dari objek atau array tersebut bisa diubah (mutable).
4. **Redeclare:**
    - `var` memungkinkan untuk diredeclare di dalam scope yang sama.
    - `let` tidak memungkinkan redeclaration di dalam scope yang sama.

- `const` juga tidak memungkinkan redeclaration di dalam scope yang sama.

5. **Initialization:**
   - `var` bisa dideklarasikan tanpa diinisialisasi dan akan memiliki nilai `undefined`.

```
console.log(x);
var x = 5;

//Hasilnya: undefined
```

   - `let` dan `const` harus diinisialisasi dengan nilai sebelum digunakan.

```
let x = 5;
console.log(x); //Hasilnya: 5

const y = 5;
console.log(y); //Hasilnya: 5
```

6. **Global object property:**
   - Ketika dideklarasikan di dalam lingkup global (di luar fungsi atau blok), `var` akan menambahkan properti pada objek global (`window` di browser).
   - `let` dan `const` tidak menambahkan properti pada objek global.

Tipe Data dalam JavaScript

- Ada banyak tipe data di Javascript, namun kali ini Saya hanya akan menyebutkan 3 terlebih dahulu yang paling sering digunakan.
- Untuk mengecek tipe data menggunakan kata kunci `typeof`
- 3 tipe data yaitu:
  1. Number

```
let angka = 5;

console.log(typeof angka); //Hasilnya: Number
```

2. String

   Untuk membuat string harus menggunakan tanda single quote `'…'` , double quote `"…"` atau backticks `` `...` `` . Selengkapnya akan dijelaskan di topik string.

```
let nama = "Budi Kusuma";

console.log(typeof nama); //Hasilnya: String
```

3. Boolean

   Hanya ada dua yaitu, `True` atau `False`

```
let benar = True;

console.log(typeof benar); //Hasilnya: Boolean
```

Gaya Penulisan Code di JavaScript

- Case sensitivity: JavaScript adalah bahasa yang case-sensitive, artinya huruf besar dan kecil dianggap berbeda. Misalnya, variabel bernama `data` berbeda dari variabel bernama `Data`.

```
var data = "Hello";
var Data = "World";

console.log(data); // Hasilnya: Hello
console.log(Data); // Hasilnya: World
```

- **Camel case**: Untuk penamaan variabel dan fungsi, JavaScript biasanya menggunakan gaya camel case, di mana huruf pertama adalah huruf kecil dan setiap kata berikutnya dimulai dengan huruf besar.

```
var firstName = "Budi";
function getUserData() {
    // kode fungsi
}
```

- **Snake case:** adalah gaya penulisan di mana setiap kata dipisahkan oleh tanda garis bawah (_) dan semua huruf ditulis dalam huruf kecil. Bisa digunakan untuk penamaan variabel.

  Contohnya: `nama_depan`, `nama_belakang`.

- **Capital case:** gaya penulisan di mana setiap kata dimulai dengan huruf kapital. Bisa digunakan untuk penamaan variabel.

  Contohnya: `NamaDepan`, `NamaBelakang`.

- **Kebab case:** gaya penulisan di mana setiap kata dipisahkan oleh tanda hubung (-) dan semua huruf ditulis dalam huruf kecil. Biasanya digunakan untuk menulis nama file `test-1.js`. Untuk penamaan variabel tidak bisa menggunakan kebab case.

  Contohnya: `nama-depan`, `nama-belakang`.

Operasi Aritmatika/Perhitungan di JavaScript

JavaScript mendukung berbagai operasi aritmatika dasar seperti penjumlahan, pengurangan, perkalian, pembagian, dan modulus. Berikut adalah beberapa contohnya:

1. **Penjumlahan (+)**

```
let a = 5;
let b = 3;
let hasil = a + b;

console.log(hasil); // Hasilnya: 8
```

2. **Pengurangan (-)**

```
let a = 5;
let b = 3;
let hasil = a - b;

console.log(hasil); // Hasilnya: 2
```

3. **Perkalian (*)**

```
let a = 5;
let b = 3;
let hasil = a * b;

console.log(hasil); // Hasilnya: 15
```

4. **Pembagian (/)**

```
let a = 6;
let b = 3;
let hasil = a / b;

console.log(hasil); // Hasilnya: 2
```

5. **Modulus (%)**

Operasi modulus menghasilkan sisa dari pembagian dua angka.

```
let a = 5;
let b = 3;
let hasil = a % b;

console.log(hasil); // Hasilnya: 2
```

6. **Eksponen ()**

Operator eksponen digunakan untuk memangkatkan angka ke pangkat tertentu.

```
let a = 2;
let b = 3;
let hasil = a ** b;

console.log(hasil); // Hasilnya: 8
```

7. **Increment (++) dan Decrement (--)**

Operator inkrement (`++`) menambah nilai variabel sebesar 1, sedangkan operator dekrement (`--`) mengurangi nilai variabel sebesar 1.

```
let a = 5;
a++;
console.log(a); // Hasilnya: 6

let b = 5;
b--;
console.log(b); // Hasilnya: 4
```

Dengan memahami operasi aritmatika dasar ini, kita dapat melakukan berbagai jenis perhitungan dalam JavaScript.

Operasi Komparasi di JavaScript

Operasi komparasi digunakan untuk membandingkan dua nilai. Hasil dari operasi komparasi adalah nilai boolean (`true` atau `false`). Berikut adalah beberapa operasi komparasi yang paling umum digunakan di JavaScript:

1. **Sama dengan (==)**

Membandingkan dua nilai untuk kesetaraan tanpa memperhatikan tipe data.

```
let a = 5;
let b = "5";

console.log(a == b); // Hasilnya: true
```

2. **Sama persis (===)**

Membandingkan dua nilai untuk kesetaraan dengan memperhatikan tipe data.

```
let a = 5;
let b = "5";

console.log(a === b); // Hasilnya: false
```

3. **Tidak sama dengan (!=)**

Membandingkan dua nilai untuk ketidaksamaan tanpa memperhatikan tipe data.

```
let a = 5;
let b = "5";

console.log(a != b); // Hasilnya: false
```

4. **Tidak sama persis (!==)**

Membandingkan dua nilai untuk ketidaksamaan dengan memperhatikan tipe data.

```
let a = 5;
let b = "5";
```

```
console.log(a !== b); // Hasilnya: true
```

5. **Lebih besar dari (>)**

Membandingkan apakah nilai di sebelah kiri lebih besar dari nilai di sebelah kanan.

```
let a = 7;
let b = 5;

console.log(a > b); // Hasilnya: true
```

6. **Lebih besar atau sama dengan (>=)**

Membandingkan apakah nilai di sebelah kiri lebih besar atau sama dengan nilai di sebelah kanan.

```
let a = 5;
let b = 5;

console.log(a >= b); // Hasilnya: true
```

7. **Lebih kecil dari (<)**

Membandingkan apakah nilai di sebelah kiri lebih kecil dari nilai di sebelah kanan.

```
let a = 3;
let b = 5;

console.log(a < b); // Hasilnya: true
```

8. **Lebih kecil atau sama dengan (<=)**

Membandingkan apakah nilai di sebelah kiri lebih kecil atau sama dengan nilai di sebelah kanan.

```
let a = 5;
let b = 5;
```

```
console.log(a <= b); // Hasilnya: true
```

Dengan memahami operasi komparasi ini, kita dapat membuat logika yang lebih kompleks dalam kode JavaScript kita.

Contoh Aplikasi Operasi Aritmatika

Berikut adalah beberapa contoh soal aplikasi operasi aritmatika beserta jawabannya:

1. Sebuah restoran menawarkan diskon 15% untuk pembelian di atas Rp200.000. Jika seseorang memesan makanan seharga Rp250.000, berapakah yang harus dibayar setelah diskon?

```
// Jawaban Soal 1:
let hargaMakanan = 250000;
let persentaseDiskon = 15%;

let diskon = 250000 * 15%
let totalBayar = hargaMakanan - diskon

console.log("Total bayar setelah diskon: Rp" + totalBayar);
// Hasilnya: Total bayar setelah diskon: Rp212500
```

2. Sebuah mobil melaju dengan kecepatan rata-rata 60 km/jam. Jika mobil tersebut menempuh perjalanan selama 2,5 jam, berapa kilometer jarak yang ditempuh?

```
// Jawaban Soal 2:
let kecepatan = 60; // km/jam
let waktu = 2.5; // jam
```

```
let jarak = kecepatan * waktu;
console.log("Jarak yang ditempuh: " + jarak + " km"); // Hasilny
```

Operasi Logika di JavaScript

Operasi logika digunakan untuk menggabungkan beberapa kondisi atau membandingkan nilai boolean. Berikut adalah beberapa operasi logika yang umum digunakan di JavaScript:

1. **AND Logika (&&)**
   - Operator AND (`&&`) mengembalikan `true` jika kedua operandnya `true`.
   - `false` jika salah satu operandnya `false`.

```
let a = true;
let b = false;

console.log(a && b); // Hasilnya: false
```

2. **OR Logika (||)**
   - Operator OR (`||`) mengembalikan `true` jika salah satu dari operandnya `true`.
   - `false` jika kedua operandnya `false`.

```
let a = true;
let b = false;

console.log(a || b); // Hasilnya: true
```

3. **NOT Logika (!)**
   - Operator NOT (`!`) mengembalikan `true` jika operandnya `false`
   - `false` jika operandnya `true`.

```
let a = true;

console.log(!a); // Hasilnya: false
```

4. **Contoh Gabungan Operasi Logika**

   Kita dapat menggabungkan beberapa operasi logika dalam satu ekspresi untuk membuat kondisi yang lebih kompleks.

```
let a = true;
let b = false;
let c = true;

console.log((a && b) || c); // Hasilnya: true
```

Dengan memahami operasi logika ini, kita bisa membuat kondisi yang lebih dinamis dan kompleks dalam kode JavaScript kita.

Contoh Aplikasi Operasi Logika

Berikut adalah dua contoh soal aplikasi operasi logika beserta jawabannya:

1. Sebuah toko online memberikan diskon 10% jika pelanggan membeli lebih dari 5 item atau total pembelian melebihi Rp500.000. Buatlah program untuk menentukan apakah seorang pelanggan berhak mendapatkan diskon.

```
// Jawaban Soal 1:
let jumlahItem = 4;
let totalPembelian = 600000;

let berhakDiskon = (jumlahItem > 5) || (totalPembelian > 500000
```

```
console.log("Pelanggan berhak mendapatkan diskon: " + berhakDisk
// Hasilnya: Pelanggan berhak mendapatkan diskon: true
```

2. Sebuah situs web hanya mengizinkan akses ke halaman tertentu jika pengguna sudah login dan memiliki usia di atas 18 tahun. Buatlah program untuk menentukan apakah seorang pengguna dapat mengakses halaman tersebut.

```
// Jawaban Soal 2:
let sudahLogin = true;
let usia = 20;

let dapatMengakses = sudahLogin && (usia > 18);

console.log("Pengguna dapat mengakses halaman: " + dapatMengakse
// Hasilnya: Pengguna dapat mengakses halaman: true
```

Kedua contoh ini mendemonstrasikan penggunaan operasi logika AND (&&) dan OR (||) dalam situasi praktis.

Operator Assignment di JavaScript

Operator assignment digunakan untuk menetapkan nilai ke variabel. Berikut adalah beberapa operator assignment yang umum digunakan di JavaScript:

1. **Assignment (=)**

Operator assignment dasar yang digunakan untuk menetapkan nilai ke variabel.

```
let x = 10;
```

2. **Addition Assignment (+=)**

Menambahkan nilai ke variabel dan menetapkan hasilnya kembali ke variabel tersebut.

```
let x = 10;
```

```
x += 5; // x sekarang adalah 15
```

3. **Subtraction Assignment (-=)**

Mengurangi nilai dari variabel dan menetapkan hasilnya kembali ke variabel tersebut.

```
let x = 10;
x -= 5; // x sekarang adalah 5
```

4. **Multiplication Assignment (*=)**

Mengalikan nilai variabel dengan nilai lain dan menetapkan hasilnya kembali ke variabel tersebut.

```
let x = 10;
x *= 2; // x sekarang adalah 20
```

5. **Division Assignment (/=)**

Membagi nilai variabel dengan nilai lain dan menetapkan hasilnya kembali ke variabel tersebut.

```
let x = 10;
x /= 2; // x sekarang adalah 5
```

6. **Modulus Assignment (%=)**

Mengambil sisa pembagian dari variabel dengan nilai lain dan menetapkan hasilnya kembali ke variabel tersebut.

```
let x = 10;
x %= 3; // x sekarang adalah 1
```

7. **Exponentiation Assignment (**=)****

Memangkatkan nilai variabel ke pangkat tertentu dan menetapkan hasilnya kembali ke variabel tersebut.

```
let x = 2;
x **= 3; // x sekarang adalah 8
```