



! **Pembaruan!**

Modul ini dibuat pada tanggal 6 December 2019. Pembaruan terakhir adalah: **Konversi potongan kode menjadi kode interaktif..**

[Lihat riwayat »](#)

## Operator Komparasi

Sekarang kita sudah mengetahui bagaimana cara menyimpan nilai pada sebuah variabel, array, ataupun objek. Nah, selanjutnya kita akan belajar mengenai operator komparasi sebagai logika dasar dalam membandingkan nilai pada JavaScript.

Terdapat serangkaian karakter khusus yang disebut dengan operator pembanding/komparasi yang dapat mengevaluasi dan membandingkan dua nilai. Berikut daftar operator dan fungsinya:

Operator	Fungsi
==	Membandingkan kedua nilai <b>apakah sama</b> . (Tidak Identik)
!=	Membandingkan kedua nilai <b>apakah tidak sama</b> . (Tidak Identik)
===	Membandingkan kedua nilai <b>apakah identik</b> .
!==	Membandingkan kedua nilai <b>apakah tidak identik</b> .
>	Membandingkan dua nilai <b>apakah nilai pertama lebih besar dari nilai kedua</b> .
>=	Membandingkan dua nilai <b>apakah nilai pertama lebih besar atau sama dengan dari nilai kedua</b> .
<	Membandingkan dua nilai apakah nilai pertama <b>lebih kecil dari nilai kedua</b> .
<=	Membandingkan dua nilai <b>apakah nilai pertama lebih kecil dari atau sama dengan nilai kedua</b> .

Ketika kita melakukan perbandingan antara dua nilai, JavaScript akan mengevaluasi kedua nilai tersebut dan akan mengembalikan boolean dengan nilai hasil perbandingan tersebut, baik **false** , atau **true** . Berikut contohnya:

`main.js` **+**

```
1 let a = 10;
2 let b = 12;
3
4 console.log(a < b);
5 console.log(a > b);
6
7 /* output
8 true
9 false
10 */
```





INPUT

RESET

JALANKAN

Output:

Perbedaan antara “sama” dan “Identik”

Dalam operator komparasi di JavaScript, hal yang menjadi sedikit “tricky” adalah membedakan antara “sama” ( == ) dan “identik” ( === ).

Kita sudah mengetahui bahwa setiap nilai pasti memiliki tipe data baik itu number, string atau boolean. Contohnya sebuah string “10” dan number 10 merupakan hal yang serupa, tetapi keduanya tidak benar-benar sama.

Hal inilah yang membedakan antara sama dan identik pada JavaScript. Jika kita ingin membandingkan hanya dari kesamaan nilainya kita bisa gunakan == tapi jika kita ingin membandingkan dengan memperhatikan tipe datanya kita gunakan === .

Contohnya sebagai berikut:

main.js +

```
1 const aString = '10';
2 const aNumber = 10
3
4 console.log(aString == aNumber) // true, karena nilainya sama-sama 10
5 console.log(aString === aNumber) // false, karena walaupun nilainya sama, tetapi tipe datanya berbeda
6
7 /* output
8 true
9 false
10 */
```

INPUT

RESET

JALANKAN

Output:

Logical Operators

Terdapat beberapa operator lain yang dapat kita gunakan untuk menetapkan logika yang lebih kompleks, yakni dengan logical operators. Dengan logical operator kita dapat menggunakan kombinasi dari dua nilai boolean atau bahkan lebih dalam menetapkan logika.

Pada JavaScript terdapat tiga buah karakter khusus yang berfungsi sebagai logical operator, berikut macam-macam logical operator dan fungsinya:

Operator	Deskripsi	DIBANTU
----------	-----------	---------



&&	Operator dan ( <i>and</i> ), logika akan menghasilkan <b>true apabila semua kondisi terpenuhi</b> (bernilai true).
	Operator atau ( <i>or</i> ), logika akan menghasilkan <b>true apabila ada salah satu kondisi terpenuhi</b> (bernilai true).
!	Operator tidak ( <i>not</i> ), digunakan untuk membalikan suatu kondisi.

Berikut contoh penerapannya pada JavaScript:

main.js +

```
1 let a = 10;
2 let b = 12;
3
4 /* AND operator */
5 console.log(a < 15 && b > 10); // (true && true) -> true
6 console.log(a > 15 && b > 10); // (false && true) -> false
7
8 /* OR operator */
9 console.log(a < 15 || b > 10); // (true || true) -> true
10 console.log(a > 15 || b > 10); // (false || true) -> true
11
12 /* NOT operator */
13 console.log(!(a < 15)); // !(true) -> false
```

INPUT

RESET

JALANKAN

Output:

Mungkin sebagian dari kita bertanya, sebenarnya apa kegunaan dari nilai boolean selain hanya menampilkan nilai **true** dan **false** saja? Pada pembahasan tipe data sudah pernah disebutkan bahwa boolean merupakan salah satu kunci dari logika pemrograman, karena boolean dapat mengontrol aliran pada program.

Lantas bagaimana cara boolean mengontrol sebuah aliran program? Pada bab selanjutnya, kita akan membahas mengenai *if/else statement* yang dapat mengontrol *flow* pada program, tentunya pada penggunaan *statement* ini boolean sangat berperan.

← KEMBALI KE MATERI SEBELUMNYA

LANJUTKAN KE MATERI BERIKUTNYA →



Dicoding Space  
Jl. Batik Kumeli No.50, Sukaluyu,  
Kec. Cibeunying Kaler, Kota Bandung  
Jawa Barat 40123



Decode Ideas  
Discover Potential

➤ [Tentang Kami](#)

[Blog](#)

[Reward](#)

[Showcase](#)

[Hubungi Kami](#)

[FAQ](#)



DIBANTU



click bila  
belum muncul

click bila  
belum muncul

