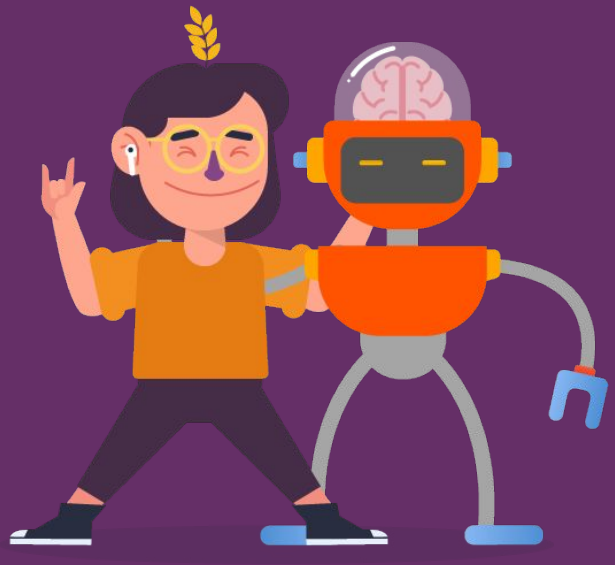
Expression dan Operators

Silver - Chapter 2 - Topic 2

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



BINAR

Kali ini kita akan membahas
Operator Javascript

Di **Chapter 2**, kita akan belajar banyak tentang **Javascript.** Mulai dari hal-hal dasar sampai ke algoritmanya.

Khusus di sesi ini, ada dua hal yang akan kamu pelajari, antara lain:

1. Menulis ekspresi di Javascript
2. Menulis operators di Javascript

Ketika sekolah, kita belajar banyak operator dalam pelajaran matematika. Mulai dari penjumlahan (+), perkalian (*), pengurangan (-), dan operator-operator lainnya.

Kali ini kita akan fokus untuk membahas aspek operator yang **tidak** kita dapatkan di bangku sekolah.

Sebelum kita bahas operator, mari kita pecah dulu apa saja yang ada di dalam sebuah operasi.

- Operan

  Adalah pelaku dari operasi itu. Contoh: `2, 3, 4, 10`

- Operator

  Memberi tahu operan apa yang harus dilakukan. Contoh: `+, -, *, %`

Lalu, ada juga yang kita sebut dengan operasi

Ada dua jenis operasi yang harus kamu tahu

- **Unary**

  Unary adalah suatu operasi yang hanya memiliki 1 operan.

```
let x = 1;
x = -x; // - (minus) di depan x kita sebut
dengan unary
console.log(x)
// Output: -1
```

-x adalah *unary*

- **Binary**

  Binary adalah suatu operasi yang memiliki lebih dari 1 operan..

```
let x = 50;
let y = 100;
console.log(x + y);
```

Pada perangkaian string, operator plus (+) memiliki dua bentuk: bentuk *binary* dan bentuk unary.

Binary + pada perangkaian string

Ternyata, operator plus **(+)** itu nggak cuma bisa dipakai dalam operasi aritmatika aja. Ia juga bisa digunakan untuk ***concatenation*** (perangkaian) sebuah string.

```
let a = "Hello";
let b = "World";
console.log(a + b);
// Output: HelloWorld
```

Pada string, operator plus (+) fungsinya bukan untuk menjumlahkan, tetapi menyatukan

```
/* Jika salah satu operan
adalah string
  Maka operan lain yang
berinteraksi dengannyan
  akan dianggap String juga
*/
console.log("1" + 2);
// Output: "12"
```

Jika salah satu operan adalah *string*, maka operan yang lain akan ikut dikonversi menjadi *string* juga

```
console.log(2 + 2 + "1");
// Output: "41"
```

Namun, perhatikan bahwa operasi berjalan dari kiri ke kanan.
Jika ada dua angka diikuti oleh *string*, angka-angka itu akan ditambahkan sebelum dikonversi menjadi *string*

Numeric conversion, unary +

Operator plus (+) dalam unary diterapkan pada nilai tunggal dan tampak tidak memberikan aksi apapun pada angka. Namun, jika operan bukan angka, ***plus unary*** mengubahnya menjadi angka.

Mengonversi *string* menjadi *number*, sebenarnya bisa kita lakukan dengan menggunakan *function* **`Number("12345").`** Tetapi, cara di atas lebih singkat dan mudah untuk dipraktikkan

```
/* Gak memiliki efek apapun
  Kalo diterapin ke angka */
let x = 1;
console.log(+x); // Output: 1
let y = -2;
console.log(+y); // Output: -2

/* Namun kalo diterapkan di tipe data
selain angka */
let z = true;
console.log(+z); // Output: 1
let w = "";
console.log(+w); // Output: 0
```

Konversi dari string menjadi *number* akan sangat sering dijumpai ketika kalian membuat **form HTML**. Mengapa begitu? Karena akan sangat memungkin bahwa data hasil input dari form HTML akan berupa string.

```
let a = "1";
let b = "2";

console.log(a + b); // Output: 12
```

Contoh yang salah

Jika kita mau menggunakannya sebagai angka, perlu dikonversi dan dijumlahkan terlebih dahulu

```
let a = "1";
let b = "2";

// a dan b kita konversi dulu menjadi angka
console.log(+a + +b); // Output: 12
```

Contoh yang benar

BINAR

Tingkatan Prioritas Operasi

Sama seperti matematika yang diajarkan di sekolah, Setiap operasi memiliki tingkat prioritas mana yang harus dikerjakan terlebih dahulu.

```
2 + 2 * 10
```

Dari operasi di atas, dapat dipastikan bahwa operasi yang akan dijalankan terlebih dahulu adalah `2 * 10`

Tapi,

Kalo dalam hal
Programming gimana?

Tabel dari tingkatan prioritas dari tiap-tiap operasi

Ada banyak operator di bahasa pemrograman Javascript. Setiap operator memiliki urutan prioritas yang sesuai. Angka yang tertinggi akan dieksekusi terlebih dahulu. Dan, jika prioritas pengurutannya adalah 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	=

Unary plus memiliki prioritas 17, yang lebih tinggi dari 13 "penambahan" (binary plus), itulah sebabnya dalam ekspresi seperti "`+apel + +mangga`", *unary plus* berfungsi sebelum penambahan.

Assignment (Sama Dengan)

Assignment adalah penentuan nilai dari sebuah data. Ketika kita bilang **`let a = 1`**, maka kita sudah melakukan sebuah *assignment* terhadap variabel **a**. Kita memberi tahu bahwa variabel a bernilai 1.

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

Memungkinkan juga kalau kita ingin melakukan *assignment* berantai.

```
let a, b, c;
a = b = c = 2 + 2;

/* 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. */
console.log(a); // Output: 4
console.log(b); // Output: 4
console.log(c); // Output: 4
```

Eksponensial (Perpangkatan)

Operator ini berfungsi untuk membuat suatu perpangkatan seperti di matematika.

2^4 ⟶ **Kalau di matematika, ketika kita ingin menulis 2 pangkat 2, kita tulis dengan cara seperti itu.**

Tapi, kalau kita pengen nulis perpangkatan/eksponen di dalam Javascript, kita nulisnya kayak gini.

```
2**4 // Ini cara menulis perpangkatan di Javascript
console.log(2**4) // Output: 16
```

Increment dan *Decrement*

(Peningkatan atau Penurunan)

Menaikkan atau menurunkan satu angka dari suatu variabel.

PENTING: Operator ini hanya berlaku untuk variabel saja, kalo kalian menambahkan operator ini pada sebuah angka secara langsung, maka akan menyebabkan error

Increment

```
let a = 2;
a++; // Nilai a akan menjadi a + 1
console.log(a); // Output: 3
```

Decrement

```
let b = 3;
b--; // Nilai b akan menjadi nilai b - 1
console.log(b); // Output: 2
```

Operator *increment* dan *decrement*, mereka bisa ditaruh sebelum (**prefix**) ataupun sesudah (**postfix**) variabel.

Seperti yang kalian coba praktekkan tadi, setiap operasi pasti mengembalikan suatu nilai, nah, akan ada **perbedaan** antara menggunakan **prefix** dengan **postfix** dalam **pengembalian nilai** dari operasi tersebut.

Yuk kita simak perbedaannya →

Prefix

```
let a = 2;
console.log(a++); // Operasi a++ akan mengembalikan nilai a yang asli
// Output: 2
console.log(a);
// Output: 3
```

Postfix

```
a = 2; // Mengatur ulang nilai a
console.log(++a); // Operasi ++a akan mengembalikan hasil dari operasi
// Output: 3
console.log(a);
// Output: 3
```

Bitwise

Operator Bitwise memperlakukan argumen sebagai angka integer 32-bit dan bekerja pada level representasi binernya. Operator ini tidak hanya untuk Javascript saja, tetapi juga mendukung sebagian besar bahasa pemrograman.

Operator ini sangat jarang digunakan dan tidak akan kita bahas lebih lanjut di materi ini.

Jika anda penasaran, anda dapat membaca artikel dari Mozilla. Akan lebih praktis untuk melakukannya ketika ada kebutuhan nyata.

Beberapa contoh operator yang tersedia

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

Modify-in-place

Terkadang kita pengen menyunting nilai dari suatu variabel (Kita butuh nilai awal dari variabel itu), Buat ngelakuin itu, kita bisa pake cara ***modify in place***

```
let n = 2;
console.log(n) // Output: 2

/* Kita pengen rubah nilai n,
 tapi kita butuh nilai n yang sebelumnya */
n = n + 5; // n = 2 + 5
console.log(n); // Output: 7

n = n ** 2; // n = 7 ** 2
console.log(n); // Output: 49
```

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

```
/* Atau bisa ditulis lebih simpel kayak gini */
let n = 2;
n += 5; // n = 7 (n = n + 5)
n **= 2; // n = 49 (n = n ** 2)
```

Operator ***"modify-and-assign"*** berlaku juga untuk semua operator aritmatika dan bitwise: **/=**, **-=**, dan lain-lain.

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

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

Comma

Operator koma (,) adalah salah satu operator yang paling langka dan paling tidak biasa. Terkadang digunakan untuk menulis kode yang lebih pendek. Operator koma memungkinkan kita untuk mengevaluasi beberapa ekspresi, membaginya dengan koma. Masing-masing dievaluasi tetapi hanya hasil terakhir yang akan dikembalikan

```
let a = (1 + 2, 3 + 4);
console.log(a);
// Output: 7, hasil dari 3 + 4
```

LOGICAL OPERATOR

Javascript

Ada tiga logical operator dalam javascript: **|| (OR), && (AND), ! (NOT).**

Meskipun disebut *"logic"*, operator-operator ini tidak hanya untuk *boolean*, tapi bisa juga diterapkan pada nilai-nilai jenis lainnya. Dan, hasilnya juga bisa dari jenis apapun.

|| (OR)

Operator "OR" diwakili dengan dua simbol garis vertikal.

Dalam pemrograman klasik, logika OR dimaksudkan hanya untuk memanipulasi nilai boolean. Jika salah satu argumennya benar maka akan mengembalikan nilai benar dan jika salah maka akan mengembalikan nilai salah.

```
let hasil = false || 10;
console.log(hasil); // Output: 10;

console.log(true || 10); // Output: true
console.log(null || 5); // Output: 5
console.log(undefined || null) // Output:
null
console.log(0 || "Hello World") // Output:
Hello World
```

&& (AND)

Operator ini dievaluasi dari kiri ke kanan, jika salah satu operan bernilai ***false*** maka semuanya akan bernilai dianggap **false**

```
let hasil = true && 10;
console.log(hasil); // Output: 10

console.log(false && "Hello World"); //
Output: false
console.log(null && "Hello World"); //
Output: null
```

! (NOT)

Operator ini berfungsi untuk membalikkan nilai boolean.

```
let hujan = false;
// ! akan merubah nilai hujan dari false, ke true
if (!hujan) console.log("Gausah pake payung");
```

BINAR

Binary +

Digunakan untuk menjumlahkan angka atau merangkai string

&&

Logical Operator dimana operasi ini dieksekusi dari kiri ke kanan, jika dia menemui operan yang bernilai ***false*** maka operan tersebut akan menjadi hasil dari operasi tersebut, dan apabila tidak ada operan yang bernilai ***false***, maka hasil dari operasi tersebut adalah operan terakhir

||

Logical Operator dimana operasi ini dieksekusi dari kiri ke kanan, jika dia menemui operan yang bernilai ***true*** maka operan tersebut akan menjadi hasil dari operasi tersebut, dan apabila tidak ada operan yang bernilai ***true***, maka hasil dari operasi tersebut adalah operan terakhir

Tugas

Buatlah sebuah operasi-operasi ini dalam Javascript:

- Perhitungan Luas LIngkaran
- Perhitungan Keliling Lingkaran
- Perhitungan Luas Segitiga
- Perhitungan Volume Balok

Kirim hasil pekerjaan kalian dalam bentuk zip, ke email **fnurhidayat@binar.co.id**

Maks hari ini pukul 23:59 WIB.

BINAR

Saatnya

QUIZ

BINAR

1

Berapa nilai akhir dari semua variabel a, b, c dan d pada kode dibawah ini?

```
let a = 1, b = 1;

let c = ++a; // ?
let d = b++; // ?
```

A. a = 2, b = 1, c = 3, d = 2

B. a = 2, b = 2, c = 2, d = 1

C. a = 1, b = 1, c = 2, d = 1

BINAR

2

Berapa nilai dari a dan x pada kode dibawah ini?

```
let a = 2;

let x = 1 + (a *= 2);
```

A. a = 4, x = 5

B. a = 2, x = 5

C. a = 4, x = 3

BINAR

3

Apa hasil dari penggunaan operator **OR** jika kodenya seperti dibawah ini?

```
alert( null || 2 || undefined );
```

A. 2
B. undefined
C. null

BINAR

4

Apa hasil dari penggunaan operator AND jika kodenya seperti dibawah ini?

```
alert( 1 && null && 2 );
```

A. 2

B. null

C. 1

BINAR

Pembahasan

Quiz

BINAR

1

Berapa nilai akhir dari semua variabel a, b, c dan d pada kode dibawah ini?

```
let a = 1, b = 1;

let c = ++a; // ?
let d = b++; // ?
```

B. a = 2, b = 2, c = 2, d = 1

++a akan mengembalikan nilai a yang baru, namun b++ akan mengembalikan nilai b yang lama. Maka dari itu c = nilai a yang baru, dan d = nilai b yang lama. a dan b sama2 melakukan pernambahan

2

Berapa nilai dari a dan x pada kode dibawah ini?

```
let a = 2;

let x = 1 + (a *= 2);
```

A. a = 4, x = 5

a *= 2 akan mengembalikan nilai a yang baru, maka dari itu nilai a yang baru adalah 4, dan karena nilai a yang baru demikian, maka nilai x akan menjadi 5.

BINAR

3

Apa hasil dari penggunaan operator **OR** jika kodenya seperti dibawah ini?

```
alert( null || 2 || undefined );
```

A. 2

Ketika operasi dengan operator OR bertemu dengan operan yang bernilai ***true*** maka dia akan langsung menganggap operan tersebut sebagai hasil dari operasi tersebut

BINAR

4

Apa hasil dari penggunaan operator AND jika kodenya seperti dibawah ini?

```
alert( 1 && null && 2 );
```

B. null

Ketika operasi dengan operator AND bertemu dengan operan yang bernilai ***false*** maka dia akan langsung menganggap operan tersebut sebagai hasil dari operasi tersebut

BINAR

Referensi

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_Operators

BINAR

Terima kasih