

* Expression merupakan kode yang menghasilkan nilai.
* Statement menunjukkan aksi yang dilakukan.

**VARIABLE**

 atau 

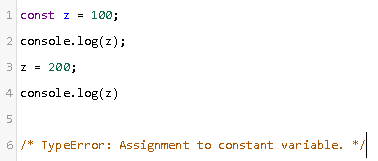
Kode untuk menginisialisasikan nilai ke dalam sebuah variabel dengan tanda sama dengan (=) ini disebut dengan assignment expression.

Tunggu sebentar. Kita kembali bertemu dengan istilah statement dan expression. Karena expression pasti menghasilkan nilai, sehingga mereka bisa muncul di mana pun dalam program JavaScript. Sementara itu, statement merujuk pada aksi. Sehingga, pada bagian kode tertentu yang membutuhkan nilai tidak bisa kita isi dengan sebuah statement. Contohnya seperti kode berikut:

1. let fullName = let lastName; // Error karena let lastName adalah sebuah statement untuk deklarasi variabel. Statement tidak bisa berada di posisi expression.
2. let fullName = (lastName = "Skywalker"); // (lastName = "Skywalker") merupakan expression, sehingga kode ini tidak error.
3. let fullName = "Luke" + "Skywalker"; // "Luke" + "Skywalker" juga merupakan expression, sehingga kode ini tidak error.

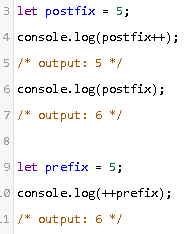
Lalu, bagaimana dengan const? Const merupakan kependekan dari constant. Artinya, kita akan mendeklarasikan sebuah variabel dengan const ketika ingin variabel bernilai konstan dan tidak bisa diubah setelah diinisialisasi nilainya. Bayangkan variabel bernilai const sebagai sebuah kotak yang ditutup dan disegel setelah diisi, sehingga nilainya tidak bisa diubah lagi.

**TIPE DATA**



Cek tipe data = 

* Jika dituliskan setelah variabel (x++), expression akan menghasilkan nilai variabel sebelum ditingkatkan nilainya.
* Jika dituliskan sebelum variabel (++x), expression akan menghasilkan nilai variabel setelah ditingkatkan nilainya.

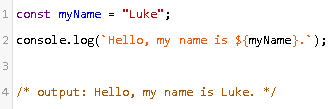


Data “Number” hanya mencakup nilai dari -(253 - 1) hingga (253 - 1). Untuk kebutuhan umum, Untuk nilai di luar Number, kita bisa menggunakan tipe BigInt. Untuk membedakan tipe BigInt dan Number, tambahkan karakter n di akhir angka. Contohnya adalah seperti kode di bawah ini. Bandingkan dengan yang bertipe Number



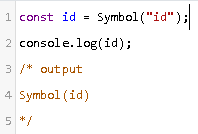
gunakan backslash(\) untuk mengurangi ambiguitas dalam tanda petik. Mekanisme ini juga dikenal dengan nama escape string. Sehingga kode di atas akan menjadi seperti berikut:

1. const answer = '"I think it\'s awesome!" he answered confidently';
2. console.log("Windows path: C:\\Program Files\\MyProject");

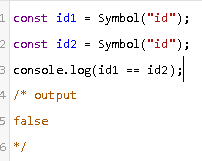


Perhatikan bahwa untuk mendefinisikan string template, Anda perlu menggunakan backticks (`), biasanya terletak di keyboard di bawah tombol Esc . Di dalam string letakkan variabel yang ingin dimasukkan ke dalam placeholder ${myName}.

Symbol adalah tipe data baru yang dikenalkan pada ES6. Tipe data Symbol digunakan untuk menunjukkan identifier yang unik. Ketika membuat Symbol, kita bisa memberikan deskripsi atau nama symbol seperti ini:



Symbol disebut sebagai identifier yang unik karena meskipun kita membuat dua variabel symbol dengan nama atau deskripsi yang sama, kedua nilainya tetap dianggap berbeda. Contohnya lihat kode berikut:



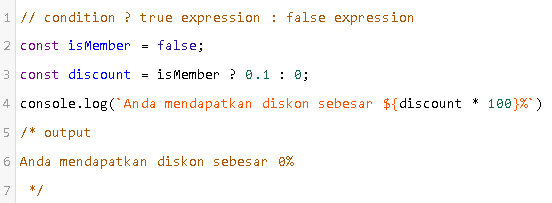
Symbol ini umumnya digunakan sebagai nama property dari Object. Object sendiri merupakan tipe data kompleks untuk menyimpan berbagai struktur data. Kita akan segera bertemu dan mempelajari tentang object pada modul Data Structure.

**OPERATOR**

1. let x = 10;
2. let y = 5;
3. x += y; // artinya -> x = x + y;
4. x -= y; // artinya -> x = x - y;
5. x \*= y; // artinya -> x = x \* y;
6. x /= y; // artinya -> x = x / y;
7. x %= y; // artinya -> x = x % y;

**IF/ELSE STATEMENT**

Selain if statement di atas, JavaScript juga mendukung ternary operator atau conditional expressions. Dengan ini, kita bisa menuliskan if-else statement hanya dalam satu baris.

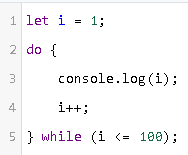
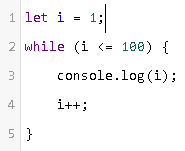


Tipe data atau nilai yang dianggap falsy, antara lain:

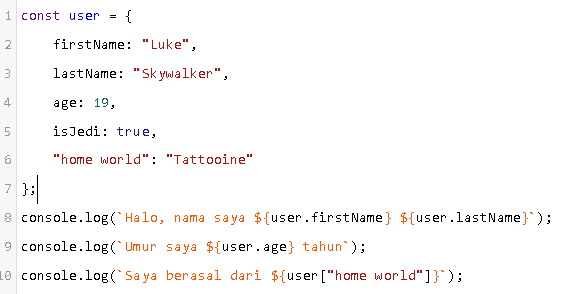
* Number 0 BigInt 0n
* String kosong seperti “” atau ‘’
* null
* undefined
* NaN, atau Not a Number

**LOOP**

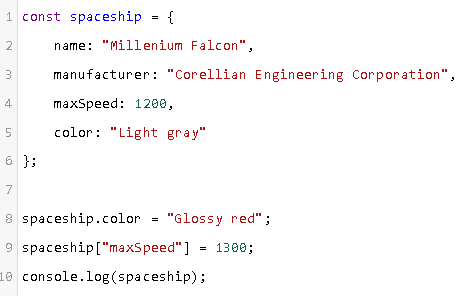
1. for(arrayItem of myArray) {
2. // do something
3. }

**OBJECT**



Setelah mempelajari bagaimana membuat object dan menampilkan property di dalamnya, selanjutnya kita akan memodifikasi sebuah object. Untuk mengubah nilai properti di dalam object kita gunakan assignment operator (=).

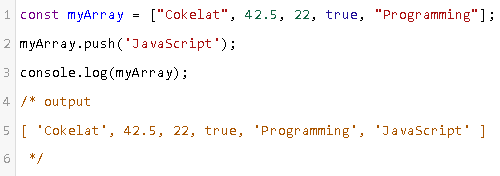


Tunggu dulu. Object spaceship dideklarasikan sebagai const, tetapi kenapa kita bisa mengubah nilainya?

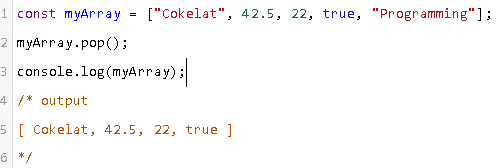
Yang perlu diperhatikan adalah mengubah nilai berbeda dengan menginisialisasi ulang nilai. Ketika membuat sebuah object, kita tidak terikat dengan properti di dalamnya sehingga kita masih bisa memodifikasi nilainya. Berbeda jika kita menginisialisasi ulang variabel dari object.

**ARRAY**

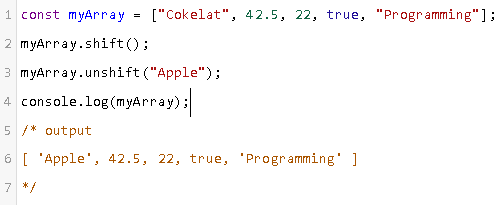
untuk menambahkan data ke dalam array, kita bisa menggunakan metode push(). Fungsi push ini akan menambahkan data di akhir array.



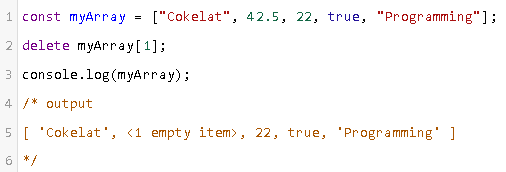
Sedangkan untuk mengeluarkan data atau elemen terakhir dari array, kita bisa gunakan metode pop().



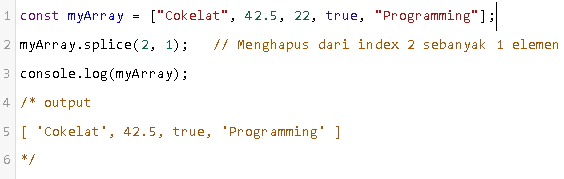
Metode lain yang bisa kita gunakan untuk memanipulasi data pada array adalah shift() dan unshift(). Metode shift() digunakan untuk mengeluarkan elemen pertama dari array, sementara unshift() digunakan untuk menambahkan elemen di awal array.



Lalu bagaimana jika kita ingin menghapus data dari array? Sama seperti object, kita bisa menggunakan keyword delete.



Namun, perhatikan di sini bahwa keyword delete hanya menghapus data pada index yang ditentukan lalu membiarkan posisi tersebut kosong. Untuk menghapus elemen, gunakan metode splice() seperti ini:

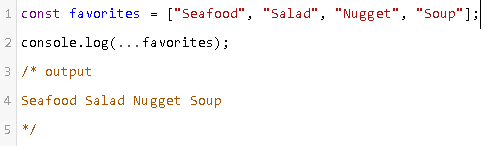


**SPREAD OPERATOR**

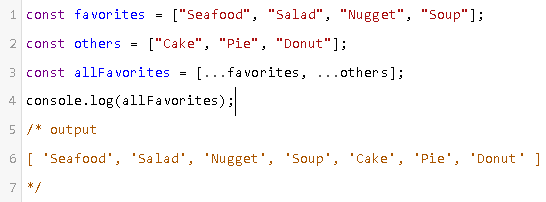
Sesuai namanya “spread”, fitur ini digunakan untuk menyebarkan nilai array atau lebih tepatnya iterable object menjadi beberapa elemen. Spread operator dituliskan dengan tiga titik (...). Mari kita lihat contoh kode berikut:

1. const favorites = ["Seafood", "Salad", "Nugget", "Soup"];
2. console.log(favorites);
3. /\* output
4. [ 'Seafood', 'Salad', 'Nugget', 'Soup' ]
5. \*/

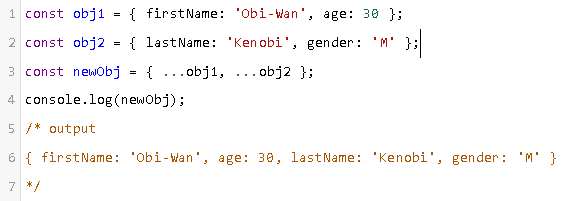
Pada kode tersebut hasil yang dicetak adalah sebuah array (ditunjukkan dengan tanda [ ]), karena memang kita mencetak nilai favorites itu sendiri. Nah, dengan menggunakan spread operator kita dapat menyebarkan nilai-nilai dalam array tersebut.



Bisa digunakan untuk menggabungkan 2 Array pada Array baru seperti berikut:



Selain array, spread operator juga bisa digunakan untuk object literals. Hal ini memungkinkan kita dapat menggabungkan beberapa object dengan kode yang lebih ringkas.



**DESTRUCTURING OBJECT**



**Destructuring Assignment**

Pada contoh sebelumnya, kita telah melakukan destructuring object pada deklarasi variabel. Namun, pada kasus tertentu mungkin kita perlu melakukannya pada variabel yang sudah dideklarasikan.

Saat melakukan destructuring assignment, kita perlu menuliskan destructuring object di dalam tanda kurung. Jika tidak menuliskan tanda kurung, tanda kurung kurawal akan membuat JavaScript mengira kita membuat block statement, sementara block statement tidak bisa berada pada sisi kiri assignment.

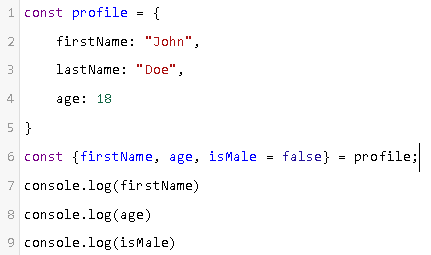
1. // tidak bisa karena JavaScript mengira kita membuat block statement
2. // block statement tidak bisa berada pada sisi kiri assignment
3. {firstName, age} = profile;

Nah, inilah fungsinya tanda kurung. Ia akan memberi tahu JavaScript bahwa tanda kurawal di dalamnya bukan sebuah block statement melainkan sebuah expression, sehingga assignment dapat dilakukan.



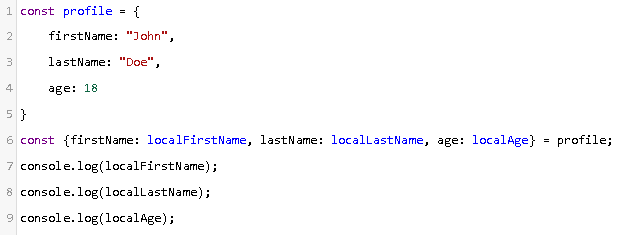
**Default Values**

Alternatifnya, kita bisa secara opsional mendefinisikan nilai default pada properti tertentu jika tidak ditemukan. Untuk melakukanya, tambahkan tanda assignment (=) setelah nama variabel dan tentukan nilai default-nya seperti ini:

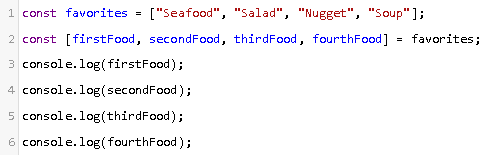


**Assigning to Different Local Variable Names**

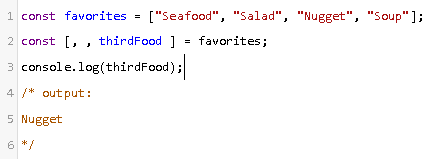
Sampai saat ini kita tahu bahwa untuk melakukan destrukturisasi object pada variabel lokal, kita perlu menyeragamkan penamaan variabel lokal dengan properti object-nya. Namun, sebenarnya dalam proses destrukturisasi object kita bisa menggunakan penamaan variabel lokal yang berbeda. ES6 menyediakan sintaksis tambahan yang membuat kita dapat melakukan hal tersebut. Penulisannya mirip seperti ketika kita membuat properti beserta nilainya pada object.



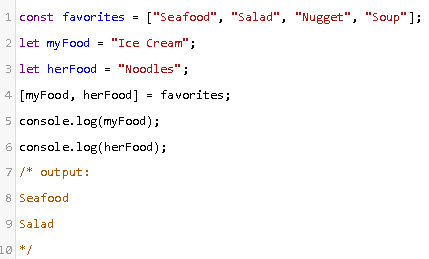
**Destructuring Array**



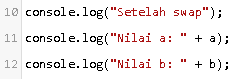
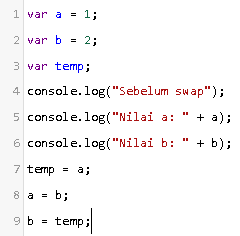
Kode di atas merupakan contoh proses destructuring array. Di dalam array favorites terdapat 4 (empat) nilai string yang masing-masing nilainya dimasukkan ke variabel lokal firstFood, secondFood, thirdFood, dan fourthFood. Nilai dari array yang dimasukkan ke variabel lokal dipilih berdasarkan posisi di mana ia dideklarasikan pada array.



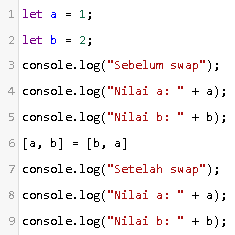
**Destructuring Assignment**



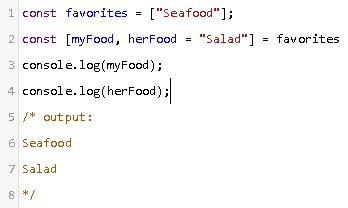
Array destructuring assignment sangat berguna ketika kita hendak menukar nilai antara dua variabel. Sebelum ES6, untuk melakukan hal ini kita menggunakan cara manual menggunakan algoritma sorting seperti ini:



Untuk melakukan pertukaran nilai, kita membutuhkan variabel penengah. Pada contoh kode di atas menggunakan variabel temp. Variabel penengah dibutuhkan untuk menyimpan data sementara pada variabel yang akan ditukar. Hal ini menjadi kurang efektif karena kita harus membuat variabel baru yang sebenarnya hanya bersifat sementara. Dengan array destructuring assignment, kita bisa menukar nilai variabel dengan mudah tanpa membuat variabel tambahan.



Sama seperti object, pada destructuring array kita juga dapat memberikan nilai default pada variabel yang tidak dapat terjangkau oleh array, sehingga nilai pada variabel tidak akan menjadi undefined.



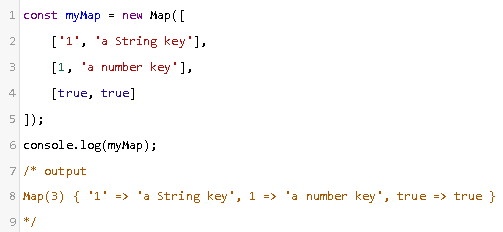
**MAP**

Map adalah tipe data yang menyimpan koleksi data dengan format key-value layaknya Object. Yang membedakan adalah Map memperbolehkan key dengan tipe data apa pun, dibandingkan Object yang hanya mengizinkan key bertipe String atau Symbol.

Untuk mendefinisikan Map gunakan constructor seperti di bawah ini:

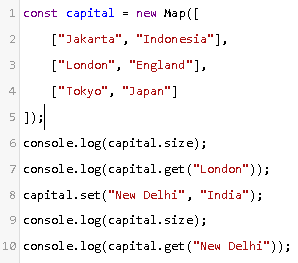
1. const myMap = new Map();

Apabila ingin menetapkan nilai dari Map secara langsung, gunakan array multi dimensi seperti ini:



Array pertama (yang berada di luar) berfungsi untuk menyimpan masing-masing elemen atau pasangan key-value dari Map. Kemudian array di dalamnya, memiliki dua elemen, di mana elemen pertama adalah key dan array keduanya merupakan value.

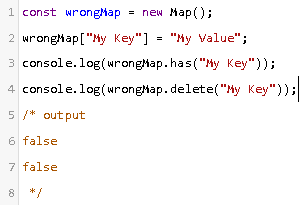
Ketika sudah membuat objek Map, kita bisa mendapatkan nilainya berdasarkan key tertentu dengan metode get(). Lalu, untuk menambahkan pasangan key-value baru gunakan metode set().



Note: Meskipun kita bisa menetapkan dan mendapatkan data pada map seperti ini:

1. const wrongMap = new Map();
2. wrongMap["My Key"] = "My Value";
3. console.log(wrongMap["My Key"]);

Namun, kode di atas bukanlah cara yang sesuai untuk mengoperasikan Map. Ketika kita menetapkan nilai map seperti di atas, data akan disimpan sebagai generic object. Ini akan mengakibatkan data tidak tersimpan dalam Map query dan tidak bisa menggunakan fitur dari Map seperti .has atau .delete.



Jadi, pastikan untuk menggunakan metode .set() dan .get() untuk Map.