**Glosarium**

**B**

**BOM**  
Singkatan dari Browser Object Model. Objek JavaScript yang tersedia di browser yang dapat digunakan untuk mengontrol browser.

**Browser**  
Aplikasi yang digunakan untuk mengakses sebuah website.

**C**

**CSS**  
Style sheet language yang digunakan untuk mendeskripsikan visual yang harus diterapkan pada markup language, salah satunya HTML.

**D**

**DOM**  
Singkatan dari Document Object Model, Objek JavaScript yang tersedia di browser yang dapat digunakan untuk mengontrol dan memanipulasi konten yang ditampilkan pada website.

**H**

**HTML**  
Markup language yang digunakan untuk menyusun struktur website.

**J**

**JavaScript**  
Bahasa pemrograman yang dapat digunakan oleh browser untuk membuat website tampak interaktif.

**L**

**Local Storage**  
Salah satu Browser API yang digunakan untuk menyimpan data tanpa ada batasan waktu. Data yang disimpan tidak akan hilang bila browser atau tabs browser ditutup kecuali jika kita menghapusnya.

**S**

**Server**  
Komputer yang dapat melayani permintaan dari komputer lainnya.

**Session Storage**  
Salah satu Browser API yang digunakan untuk menyimpan data sementara pada browser. Data akan hilang ketika browser atau tab browser ditutup.

## Daftar Referensi

[1] Dicoding, “Developer Academy: Belajar Dasar pemrograman web.” [Daring]. Tersedia: <https://www.dicoding.com/academies/123>. [Diakses: 29-Apr-2021].

[2] D. Flanagan, "The JavaScript Standard Library," dalam JavaScript: The Definitive Guide, 7th Edition. O'Reilly Media, Inc., 2020. Available: O'Reilly

[3] D. Flanagan, "Introduction to JavaScript," dalam JavaScript: The Definitive Guide, 7th Edition. O'Reilly Media, Inc., 2020. Available: O'Reilly

## Pendahuluan Dasar JavaScript

Di kelas [Belajar Dasar Pemrograman Web](https://www.dicoding.com/academies/123), Anda sudah mampu membuat website sederhana menggunakan HTML dan CSS. Bila Anda paham dan menguasai kedua skill tersebut, sebenarnya Anda sudah mampu membuat website yang sangat amat cantik. Wah!

Kini timbul pertanyaan, apakah dengan HTML dan CSS saja sudah cukup dalam membuat website? Memang benar dengan HTML dan CSS saja Anda dapat membuat sebuah website, namun website hanya sebatas halaman informasi yang statis (seperti website yang Anda buat di kelas [Belajar Dasar Pemrograman Web](https://www.dicoding.com/academies/123)). Tidak akan ada interaktivitas dengan pengguna, kaku, dan sulit dalam menampilkan konten yang dinamis.

Lalu bagaimana agar kita dapat mengembangkan website yang interaktif dan dinamis? Perkenalkan, JavaScript!

JavaScript merupakan satu-satunya bahasa pemrograman yang dapat dipahami oleh browser untuk membantu website lebih interaktif dan dinamis. Dengan JavaScript, kita dapat memberikan fungsionalitas lebih pada website. Kita bisa membuat game di website, membuat website layaknya aplikasi, bahkan mengedit foto sekalipun. Luar biasa! JavaScript mampu mengubah fungsi dari website yang sebelumnya hanya sebatas portal untuk menampilkan informasi statis.

Sebelum kita melakukan hal-hal keren menggunakan JavaScript, tentu kita perlu mendalami dulu dasar dari bahasa pemrograman JavaScript. Kita akan belajar JavaScript untuk pengembangan web secara runut agar Anda tidak bingung.

Nah, di modul ini kita akan membahas secara detail dari dasar-dasar JavaScript hingga Anda mampu:

* Menuliskan sintaks JavaScript
* Mengetahui Statement dan Expressions
* Mengetahui cara menuliskan komentar di JavaScript
* Mengetahui cara membuat variabel
* Mengetahui tipe data yang ada di JavaScript
* Mengetahui berbagai macam operator yang ada di JavaScript
* Mengetahui cara melakukan percabangan di JavaScript
* Mengetahui cara melakukan perulangan di JavaScript
* Mengetahui cara mengelola data yang kompleks menggunakan objek dan array.
* Mengetahui mendeklarasikan atau membuat fungsi dan menggunakannya.

Supaya proses belajar menjadi lebih interaktif, apabila Anda memiliki pertanyaan perihal materi disajikan, jangan segan untuk bertanya via [Forum Diskusi](https://www.dicoding.com/academies/315/discussions). Mentor kami akan sigap untuk berusaha menjawab pertanyaan Anda. Selain itu, Anda juga dipersilahkan untuk membantu menjawab pertanyaan yang diajukan oleh peserta belajar lainnya.

## Apa itu JavaScript

JavaScript adalah bahasa pemrograman tingkat tinggi yang pada awalnya dikembangkan untuk membuat website menjadi lebih “hidup”. Bersama dengan HTML dan CSS, JavaScript menjadi bahasa pemrograman paling populer untuk mengembangkan aplikasi berbasis web. Bahasa ini mampu memberikan logic ke dalam website, sehingga website tersebut memiliki fungsionalitas tambahan dan lebih interaktif.

JavaScript termasuk ke dalam kategori scripting language. Apa maksudnya? Salah satu ciri-ciri utama dari bahasa scripting adalah kode tidak perlu dikompilasi agar bisa dijalankan. Scripting language menggunakan interpreter untuk menerjemahkan kode atau perintah yang kita tulis supaya dimengerti oleh mesin.

Itulah kenapa bahasa scripting tidak membutuhkan banyak kode yang perlu ditulis agar sebuah program bisa dijalankan. Hanya dengan satu baris kode berikut Anda sudah bisa membuat program yang menampilkan teks “Hello, World!” ke layar.

1. console.log('Hello World');

Sederhana, bukan? Itulah salah satu ciri utama JavaScript sebagai scripting language. Tentu tidak lengkap rasanya ketika kita sudah mengetahui apa itu JavaScript, tetapi tidak tahu sejarah di baliknya. Sejarah singkat dari JavaScript dapat Anda simak di materi berikutnya.

## Sejarah JavaScript

JavaScript dibuat pada tahun 1995 oleh Brendan Eich, programmer dari Netscape. Bahasa ini awalnya dinamai “Mocha” kemudian berubah menjadi “LiveScript”. Pada saat itu bahasa Java merupakan bahasa pemrograman yang paling populer. Untuk memanfaatkan kepopulerannya, nama LiveScript pun diubah menjadi “JavaScript”. Jadi, meskipun namanya mirip, JavaScript tidak ada hubungannya dengan bahasa pemrograman Java.

Setelah diadopsi di luar Netscape, JavaScript distandarisasi oleh European Computer Manufacturers Association (ECMA). Sejak saat itu JavaScript juga dikenal dengan ECMAScript. Meskipun begitu, masih banyak yang menyebutnya dengan JavaScript hingga saat ini.

Terdapat beberapa versi JavaScript yang sudah distandarisasi oleh ECMAScript. Pada tahun 2000 hingga 2010, ECMAScript 3 merupakan versi yang banyak digunakan ketika JavaScript sedang mendominasi. Selama waktu tersebut, ECMAScript 4 sedang dalam proses pengembangan dengan harapan akan memberikan improvisasi yang cukup signifikan. Namun, ambisi tersebut tidak berjalan mulus sehingga pada tahun 2008 pengembangan ECMAScript dihentikan.

Walaupun begitu, ini bukan akhir dari JavaScript. Pengembangan berganti menjadi ECMAScript 5 dengan mengurangi ambisinya dan hanya melakukan perbaikan pada hal yang tidak jadi kontroversi. Pembaruan tersebut berhasil dan akhirnya ECMAScript 5 rilis pada tahun 2009.

Lalu, pada tahun 2015 ECMAScript 6 rilis dengan membawa perubahan yang cukup besar termasuk ide-ide yang sudah direncanakan untuk versi 4. Sejak saat itu, tiap tahun JavaScript melakukan update bersifat minor.

## Mengenal Statement dan Expression dengan Menulis Kode JavaScript Pertama

Menulis kode dalam suatu bahasa pemrograman sederhananya adalah menuliskan instruksi-instruksi untuk dijalankan oleh komputer. Kode di bawah ini merupakan instruksi bagi terminal atau konsol untuk mencatat (log) kalimat 'Hello, World!'.

1. console.log('Hello World');

console.log adalah kode bawaan JavaScript untuk menampilkan pesan ke konsol, bisa berupa web konsol atau konsol dari terminal/command prompt.

[](https://www.dicoding.com/academies/315/tutorials/16640?from=16635)

Kode atau teks yang berada di dalam tanda kurung adalah pesan yang ingin ditampilkan. Pada contoh kode di atas, kita menggunakan tanda kutip untuk menandakan bahwa pesan yang ingin ditampilkan merupakan sebuah string atau teks. Kita akan membahas mengenai tipe data pada beberapa modul ke depan. Saat ini kita akan fokus untuk menampilkan teks terlebih dulu. Sekarang cobalah untuk menampilkan nama Anda sendiri ke konsol.

console.log('Hello World!');

// Tampilkan nama Anda ke layar pada kode di bawah:

JavaScript membagi instruksi secara umum menjadi dua jenis, yaitu expression dan statement.

Expression adalah unit kode yang dapat dievaluasi menjadi suatu nilai. Untuk lebih mudahnya, perhatikan contoh kode berikut:

1. 5; // Baris kode ini merupakan expression karena interpreter akan membaca kode ini dan menghasilkan nilai 5.
2. 2 + 3; // Baris kode ini juga merupakan expression. Interpreter mengevaluasi kode dan juga akan menghasilkan nilai 5.

Sementara itu, statement adalah instruksi untuk melakukan sebuah aksi tertentu. Aksi ini bisa termasuk membuat variabel, melakukan perulangan, pengecekan kondisi, dll.

Jika Anda masih bingung untuk membedakan expression dan statement, tidak perlu khawatir. Saat ini belum banyak kode yang bisa kita jadikan contoh. Namun, konsep ini akan banyak kita gunakan pada materi-materi yang akan datang. Yang perlu Anda ingat adalah:

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

## Komentar

Pada materi sebelumnya kita telah membahas bahwa kode yang kita tulis merupakan kumpulan instruksi yang harus dijalankan oleh komputer. Salah satu instruksi yang penting adalah memberi tahu komputer untuk mengabaikan perintah yang kita tulis. Instruksi yang ditulis dalam suatu program tetapi tidak dijalankan oleh komputer disebut “comments”.

Sebuah komentar akan dilewatkan oleh interpreter atau compiler, sehingga tidak akan memengaruhi alur program yang kita tulis. Komentar ini bisa digunakan sebagai dokumentasi atau penjelasan dari kode yang kita tulis.

Terdapat dua metode untuk memberikan komentar. Pertama, untuk memberikan komentar pada satu baris saja, kita bisa gunakan tanda dua garis miring (//) di awal baris.

// Ini merupakan komentar satu baris, kode di bawah tidak akan dijalankan

// console.log('Halo!');

Lalu, untuk memberikan komentar lebih dari satu baris kita bisa menggunakan tanda /\* sebagai pembuka komentar dan tanda \*/ untuk penutup komentar. Teks apa pun yang berada di antara tanda tersebut akan dijadikan komentar dan tidak akan dieksekusi.

/\*

Ini merupakan komentar dengan lebih dari satu baris

Teks apa pun yang berada di sini akan dijadikan komentar.

Ketika menggunakan ini, jangan lupa untuk menutupnya.

\*/

## Variabel

Ketika menulis sebuah program, kita memberi tahu komputer cara memproses informasi seperti mencetak teks ke layar atau melakukan operasi perhitungan. Untuk lebih mudah dalam penggunaan dan pemanggilan data, kita bisa memanfaatkan variabel. Variabel umumnya digunakan untuk menyimpan informasi atau nilai yang akan dikelola dalam sebuah program.

Pada JavaScript setidaknya ada tiga cara untuk mendeklarasikan sebuah variabel, yaitu menggunakan keyword **var**, **let**, dan **const**. Pada versi ECMAScript 2015 (ES6) dikenalkan deklarasi variabel dengan **let** dan **const**untuk menggantikan **var**yang dinilai kontroversial dan rawan menimbulkan bug.

Dalam kelas ini, kita akan banyak menggunakan keyword **let**dan **const**. Jika Anda penasaran kenapa **var**sudah tidak lagi disarankan, silakan simak diskusi [berikut](https://softwareengineering.stackexchange.com/questions/274342/is-there-any-reason-to-use-the-var-keyword-in-es6).

Sekarang bagaimana caranya membuat sebuah variabel dalam JavaScript? Cukup mudah. Ketikkan keyword let yang diikuti dengan nama variabelnya.

1. let lastName;

Kode untuk mendeklarasikan variabel seperti di atas juga dikenal dengan declaration statement.

Selanjutnya, Anda bisa mengisi nilai variabel di atas menggunakan tanda sama dengan (=).

let lastName;

lastName = 'Skywalker';

console.log(lastName);

/\*

Output:

Skywalker

\*/

Anda juga bisa langsung menginisialisasi nilai variabel setelah mendeklarasikannya seperti berikut:

let lastName = 'Skywalker';

console.log(lastName);

/\*

Output:

Skywalker

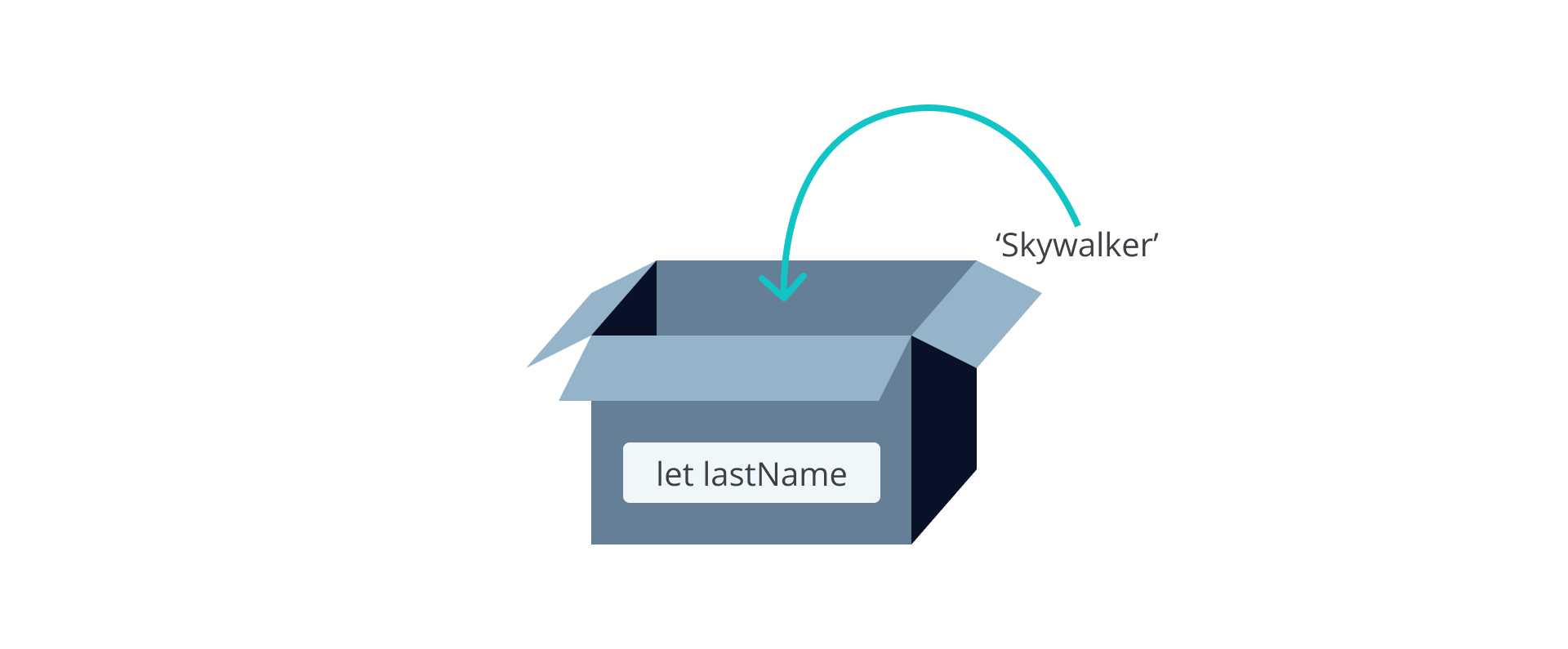
\*/

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' merupakan expression, sehingga kode ini tidak error.

Melalui contoh kode di atas, kita bisa bayangkan variabel sebagai sebuah kotak atau wadah yang menyimpan nilai. Proses inisialisasi atau assignment berarti kita memasukkan nilai ke dalam kotak tersebut.

[](https://www.dicoding.com/academies/315/tutorials/16650?from=16645)

Variabel **lastName**di atas akan tersimpan di dalam memori komputer.

Setiap variabel memiliki nama yang dapat kita panggil dan gunakan. Kita dapat menamai variabel dengan nama apa pun, tetapi pastikan penamaannya masih masuk akal dengan konteksnya supaya kode mudah di-maintenance.

Sebaiknya hindari penamaan variabel dengan istilah umum seperti “data”. Gunakanlah penamaan variabel yang dapat mendeskripsikan nilai dari variabel itu sendiri. Berikut beberapa aturan dalam penamaan variabel yang perlu Anda ketahui:

* Harus dimulai dengan huruf atau underscore (\_).
* Dapat terdiri dari huruf, angka, dan underscore (\_) dengan berbagai kombinasi.
* Tidak boleh mengandung spasi (whitespace). Jika penamaan variabel lebih dari dua kata, tuliskan secara camelCase. Contoh firstName, lastName, catName, dll.
* Tidak boleh mengandung karakter spesial (! . , / \ + \* = dll.)

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.

Jika menginisialisasi kembali nilai variabel yang menggunakan const, kita akan mendapati eror “TypeError: Assignment to constant variable.”

const z = 100;

console.log(z);

z = 200;

console.log(z);

/\* TypeError: Assignment to constant variable. \*/

## Tipe Data

Pada materi sebelumnya kita telah belajar tentang variabel untuk menyimpan nilai. Nilai yang kita tetapkan pada variabel pasti memiliki tipe data. Tipe data merupakan pengklasifikasian data berdasarkan jenisnya. Pada JavaScript terdapat beberapa tipe data sebagai berikut:

### Undefined

Tipe data ini terbentuk ketika sebuah variabel tidak memiliki nilai. Artinya, ketika kita mendeklarasikan variabel tanpa menginisialisasikan nilainya, variabel tersebut menjadi undefined. Contoh:

1. let x;
2. console.log(typeof(x));
4. /\* Output: undefined \*/

Pada contoh kode di atas, kita mendeklarasikan variabel x, tetapi kita tidak menginisialisasikan dengan nilai apa pun. Ketika kita memastikan tipe data dengan menggunakan fungsi typeof(), ia akan menghasilkan output undefined.

**Catatan:** Fungsi typeof() digunakan untuk memastikan tipe data pada variabel dengan mengembalikan tipe data tersebut dalam bentuk teks.

### Numbers

Nilai dari tipe data number adalah angka. Variabel bertipe data number dituliskan seperti angka pada umumnya:

1. let x = 10;

Jika angka tersebut merupakan sebuah bilangan desimal, maka kita bisa gunakan tanda titik pada pecahan bilangannya.

let x = 10;

console.log(typeof(x));

/\* Output: number \*/

let y = 17.25;

console.log(typeof(y));

/\* Output: number \*/

Pada tipe data number, kita juga dapat melakukan perhitungan aritmatika seperti penjumlahan, pengurangan, perkalian, dsb. Berikut operator yang dapat kita gunakan dalam perhitungan aritmatika pada tipe data number:

| **Operator** | **Fungsi** | **Contoh** |
| --- | --- | --- |
| + | Penjumlahan | 10 + 10 = 20 |
| - | Pengurangan | 15 - 7 = 8 |
| / | Pembagian | 21 / 7 = 3 |
| \* | Perkalian | 9 \* 9 = 81 |
| % | Sisa hasil bagi | 5 % 2 = 1 |

let a = 12;

let b = 9;

console.log(a + b);

console.log(a - b);

console.log(a \* b);

console.log(a / b);

console.log(a % b);

/\* Output:

21

3

108

1.3333333333333333

3

\*/

Pada operator aritmatika juga terdapat operator increment (++) dan decrement (--). Operator increment dan decrement digunakan untuk menambahkan atau mengurangi nilai 1 pada nilai variabel yang ada sekarang.

Operator ini dapat dituliskan sebelum atau sesudah variabel, tetapi hal tersebut bukan berarti sama. Berikut ketentuannya:

* 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.

Untuk lebih jelasnya, berikut adalah contoh kode penerapan operator tersebut, perhatikan hasil yang didapat.

/\* Increment dan Decrement \*/

let postfix = 5;

console.log(postfix++); /\* Output: 5 \*/

console.log(postfix); /\* Output: 6 \*/

let prefix = 5;

console.log(++prefix); /\* Output: 6 \*/

### BigInt

Pada JavaScript, tipe data “Number” hanya mencakup nilai dari -(253 - 1) hingga (253 - 1). Untuk kebutuhan umum, sebenarnya nilai tersebut sudah sangat cukup. Namun, akan ada kebutuhan tertentu di mana kita membutuhkan cakupan nilai yang lebih besar, seperti untuk kriptografi atau menentukan waktu hingga presisi microsecond.

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.

const bigNumber = 1234567890123456789012345678901234567890n;

const myInt = 1234567890123456789012345678901234567890;

console.log(bigNumber);

console.log(myInt);

/\* Output:

1234567890123456789012345678901234567890n

1.2345678901234568e+39

\*/

Meskipun digunakan untuk menyimpan angka dengan nilai besar, namun BigInt tetap bisa digunakan untuk nilai yang lebih kecil.

1. const bigIntButSmall = 7n;

Kita juga bisa menggunakan BigInt untuk operasi aritmatika pada umumnya. Yang membedakan adalah pada operasi pembagian, hasilnya akan dibulatkan ke bawah dan tanpa mengandung nilai desimal. Contohnya adalah seperti ini:

console.log(5n + 2n);

console.log(5n - 2n);

console.log(5n \* 2n);

console.log(5n / 2n);

console.log(5n % 2n);

/\* Output:

7n

3n

10n

2n (Bukan 2.5n)

1n

\*/

### Strings

Tipe data selanjutnya adalah string yang merupakan sebuah teks. Untuk menetapkan nilai sebagai string pada variabel gunakan tanda petik satu (‘) atau petik dua (“) di antara teksnya. Contohnya:

let greet = 'Hello';

console.log(typeof(greet));

/\* Output: string \*/

Tidak ada perbedaan antara menggunakan petik satu atau petik dua. Anda dapat menggunakan tanda petik secara bergantian, khususnya jika Anda memiliki teks yang mengandung tanda petik.

const question = '"What do you think of JavaScript?" I asked';

console.log(question);

/\* Output: "What do you think of JavaScript?" I asked \*/

Lalu bagaimana

const answer = '"I think it's awesome!" he answered confidently'; // Invalid

console.log(answer);

Tentunya kode di atas akan menghasilkan eror. Solusinya, 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'; // Valid

Backslash sebelum tanda petik akan memberitahukan interpreter bahwa itu hanyalah tanda petik dan tidak boleh ditafsirkan sebagai pembatas string. Selain tanda petik, backslash juga berguna untuk mengabaikan simbol lain yang menimbulkan ambigu di dalam string, contohnya seperti backslash itu sendiri.

1. console.log("Windows path: C:\\Program Files\\MyProject");

Pada String, kita juga dapat menggunakan operator plus (+). Operator tersebut berfungsi untuk menggabungkan dua teks yang terpisah menjadi satu buah teks. Contohnya seperti ini:

let greet = 'Hello';

let moreGreet = greet + greet;

console.log(moreGreet);

/\* Output: HelloHello \*/

Ingat, string concatenation seperti di atas akan menggabungkan string apa adanya sehingga jika Anda ingin menggabungkan dua kata atau lebih, maka perlu menambahkan spasi sendiri.

let greet = 'Hello';

let moreGreet = greet + " " + greet;

console.log(moreGreet);

/\* Output: Hello Hello \*/

let x = true;

let y = false;

console.log(typeof(x))

console.log(typeof(y))

/\* Output:

boolean

boolean

\*/

Kita juga bisa menggunakan operator komparasi seperti lebih dari (>) atau kurang dari (<) untuk menghasilkan output boolean. Contohnya:

const a = 10;

const b = 12;

let isGreater = a > b;

let isLess = a < b;

console.log(isGreater);

console.log(isLess);

/\* Output:

false

true

\*/

### Null

Tipe berikutnya adalah null. Serupa dengan undefined, namun null perlu diinisialisasikan pada variabel. Null biasa digunakan sebagai nilai sementara pada variabel, tapi sebenarnya nilai tersebut “tidak ada”.

Terkadang kita perlu membuat sebuah variabel, namun kita belum memerlukan nilai apa-apa dan tidak ingin terikat oleh tipe data apa pun. Nah, daripada kita tidak menetapkan nilai apa pun (variabel akan undefined) sebaiknya kita beri nilai null pada variabel tersebut dan ubah nanti ketika kita membutuhkannya.

Untuk menetapkan null pada variabel, kita dapat gunakan keyword null ketika variabel tersebut diinisialisasi.

1. let someLaterData = null;
2. console.log(someLaterData);
4. /\* Output:
5. null
6. \*/

### Symbol

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:

const id = Symbol('id');

console.log(id);

/\* Output:

Symbol(id)

\*/

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:

const id1 = Symbol('id');

const id2 = Symbol('id');

console.log(id1 == id2);

/\* Output:

false

\*/

Symbol ini umumnya digunakan sebagai nama properti dari Object. Object sendiri merupakan tipe data kompleks untuk menyimpan berbagai struktur data. Kita akan segera bertemu dan mempelajari tentang object nanti.

/\*\*

\* TODO:

\* Buatlah variabel firstName, lastName, age, isMarried dengan ketentuan:

\* - firstName: bertipe data string, dengan nilai nama depan Anda.

\* - lastName: bertipe data string, dengan nilai nama belakang Anda.

\* - age: bertipe data number, dengan nilai umur Anda.

\* - isMarried: bertipe data boolean, dengan bebas Anda tentukan.

\*/

// TODO

let firstName = "Aditya Imam";

let lastName = "Zuhdi";

let age = 21;

let isMarried = false;

console.log(firstName,lastName,age,isMarried);

/\*\*

\* Jangan hapus kode di bawah ini

\*/

module.exports = {

firstName, lastName, age, isMarried,

};