JavaScript

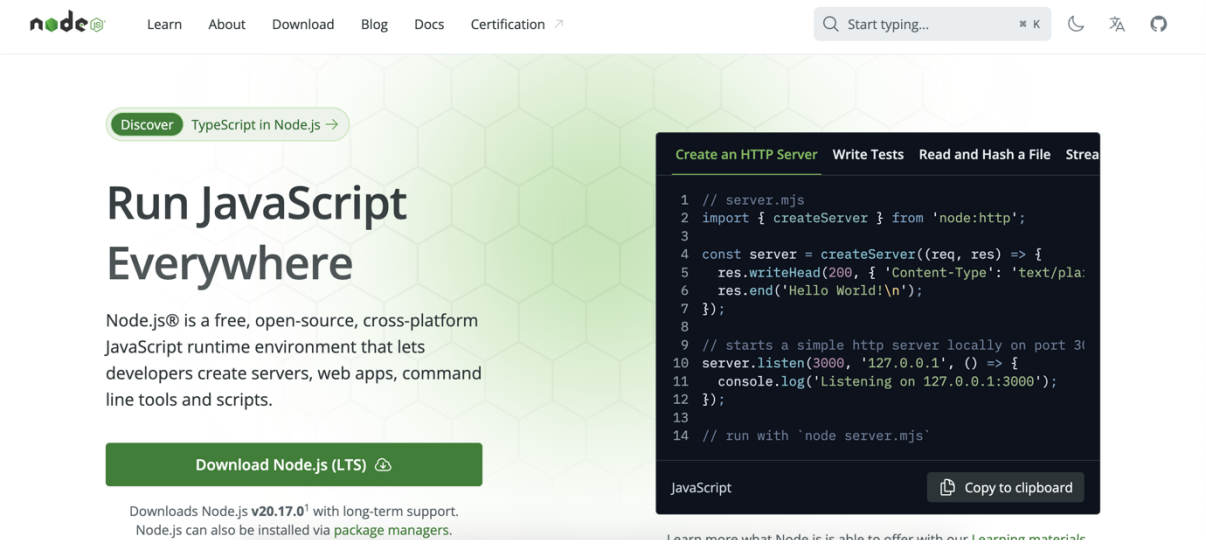
A Game Changer of Complex Applications

Kevin Perdana

JavaScript

Node.js

Download Node.js pada halaman berikut : **https://nodejs.org/**

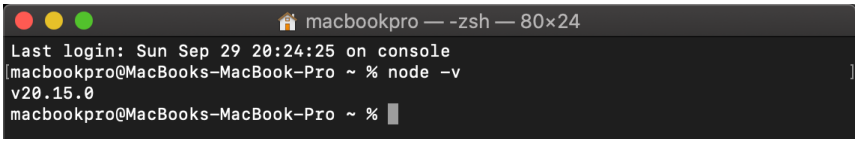


Petunjuk Instalasi
Windows :

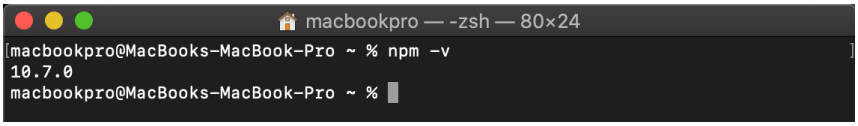
- Unduh file installer .msi.
- Jalankan file installer dan ikuti petunjuk di layar.
- Pastikan untuk mencentang opsi "Add to PATH" saat diminta.

macOS :
Gunakan Homebrew dengan cara ketik **brew install node** pada terminal.

Verifikasi Instalasi
Setelah instalasi selesai, buka terminal atau command prompt dan ketik : **node -v**



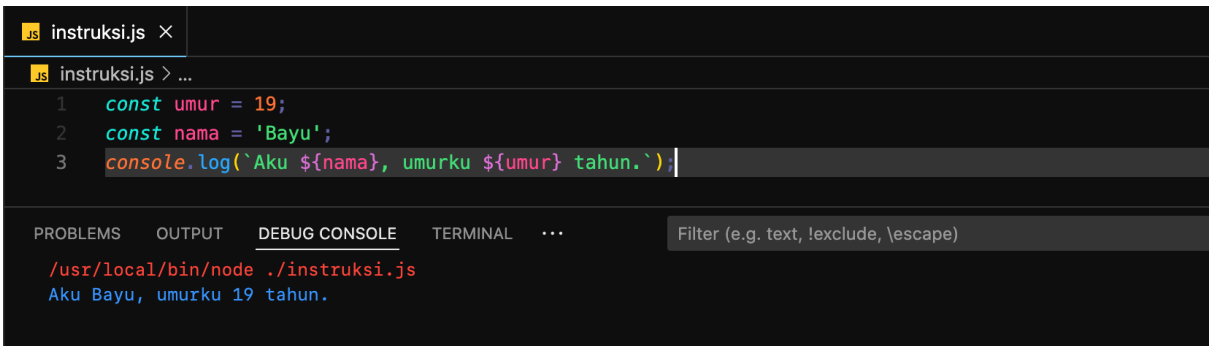
Dan **npm -v**



npm (Node Package Manager) adalah alat manajemen paket yang digunakan untuk mengelola dependensi dalam proyek Node.js.

Expression dan Statement

Pada pemrograman, satu instruksi umumnya ditulis dalam satu baris kode. Jadi, tiga instruksi umumnya ditulis dalam tiga baris kode. Istilah yang digunakan untuk "instruksi" dalam bahasa pemrograman adalah *"statement"*.



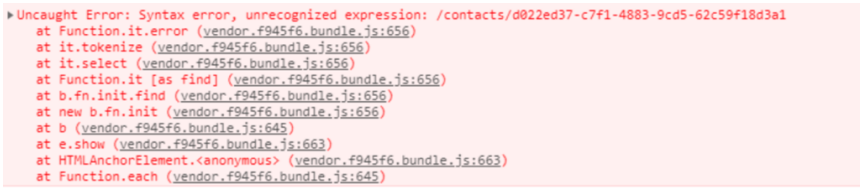
Bisa disimpulkan bahwa statement adalah instruksi yang akan dieksekusi oleh komputer.

Selain statement, ada istilah lain yang penting untuk diketahui, yaitu expression. Ini merupakan bagian dari sebuah statement yang menghasilkan nilai. Setiap statement biasanya mengandung setidaknya satu expression. Contoh, expression dalam kode di atas adalah nilai umur 19, teks “Bayu”, dan teks “Aku Bayu, umurku 19 tahun.”.

Berikut adalah ilustrasi yang bisa Anda lihat untuk memudahkan pemahaman mengenai bagian expression pada statement.

Expression dapat berupa hal-hal yang menghasilkan nilai, tidak hanya nilai statis, seperti angka 19 atau teks “Bayu”. Namun, hasil operasi matematika seperti 4 + 4 yang akan menghasilkan nilai 8 atau penggabungan teks seperti “Kevin” + “ “ + “Perdana” yang akan menghasilkan teks “Kevin Perdana” juga merupakan sebuah expression.

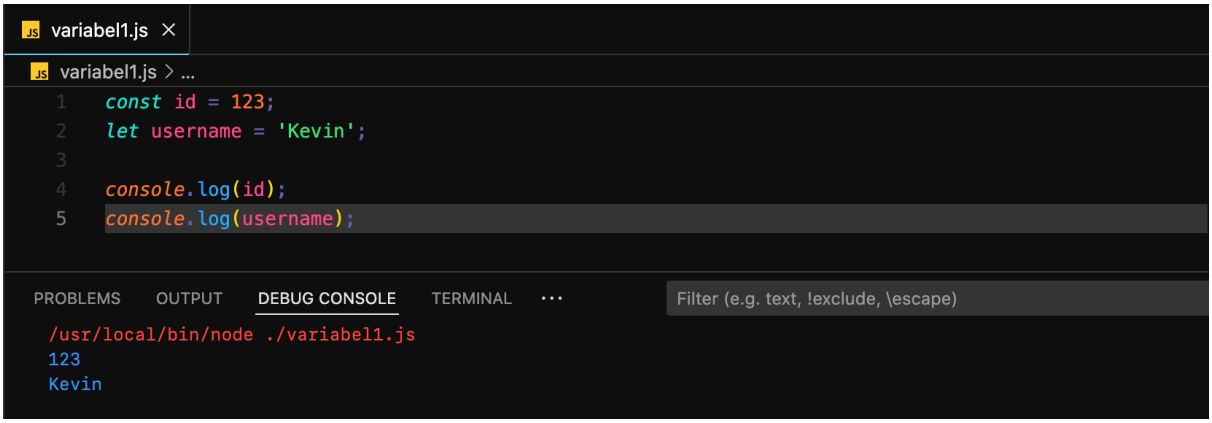
Memahami istilah-istilah ini dalam struktur kode JavaScript sangatlah penting, baik dalam proses pembelajaran maupun pengembangan aplikasi nyata. Salah satu manfaat utamanya adalah ketika Anda menghadapi pesan error dari interpreter JavaScript yang sering menggunakan istilah statement atau expression dalam pesan error-nya.



Variabel

Variabel adalah wadah untuk menampung sebuah nilai. Nilai yang ditampung dapat berupa angka, teks, atau apa pun yang menghasilkan nilai (expression). Pada JavaScript, kita bisa membuat variabel melalui dua cara, yakni dengan sintaksis **const** dan **let**.

Berikut adalah contoh cara membuat variabel dengan menggunakan **const** dan **let**.

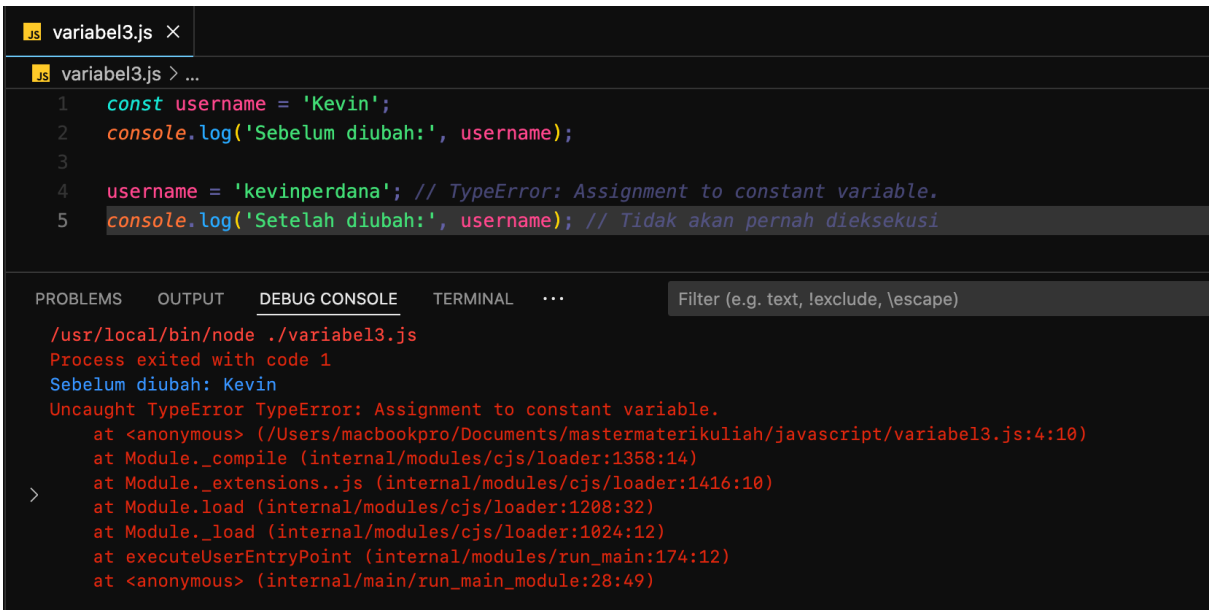


Kode di atas merupakan contoh membuat variabel. Pada contoh tersebut, variabel bernama **id** dibuat menggunakan **const** dan variabel bernama **username** dibuat dengan **let**.

Perbedaan dari variabel yang dibuat dengan **const** dan **let** adalah variabel yang dibuat dengan **const** tidak dapat diinisialisasi ulang (sederhananya, diubah) nilainya, sedangkan jika variabel dibuat dengan **let**, kita bisa menginisialisasi ulang nilainya.

Contoh, ketika membuat variabel dengan **let**, Anda bisa mengubah nilai yang ada di dalamnya setelah variabel tersebut dibuat.

Namun, ketika Anda membuat variabel dengan **const**, nilai yang ditetapkan ketika variabel dibuat, tidak bisa diubah. Jika Anda coba untuk mengubahnya, program akan terhenti dan menghasilkan **error**.



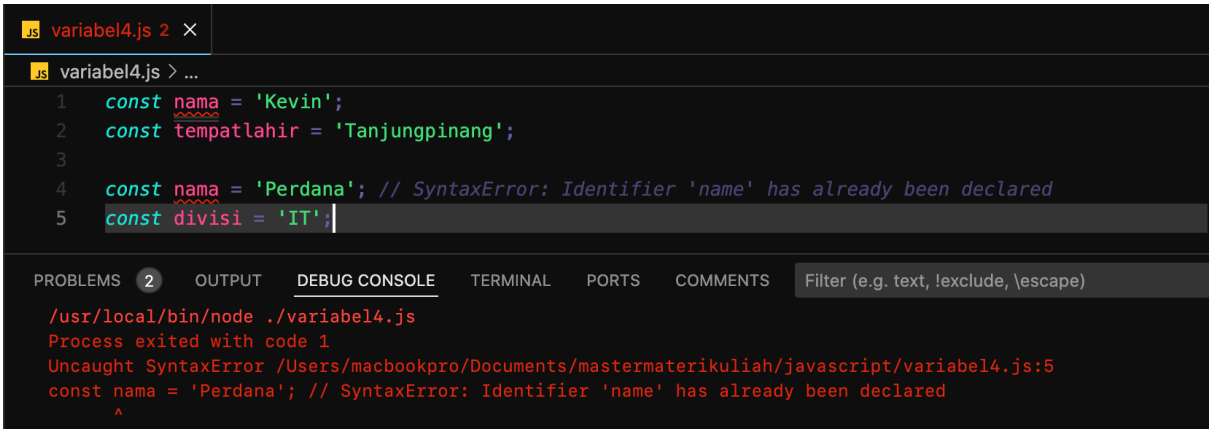
Variabel yang dibuat dengan const juga umum disebut sebagai constant (konstan) karena dalam matematika istilah tersebut memiliki arti tetap atau tidak berubah-ubah.

Aturan Penamaan Variabel

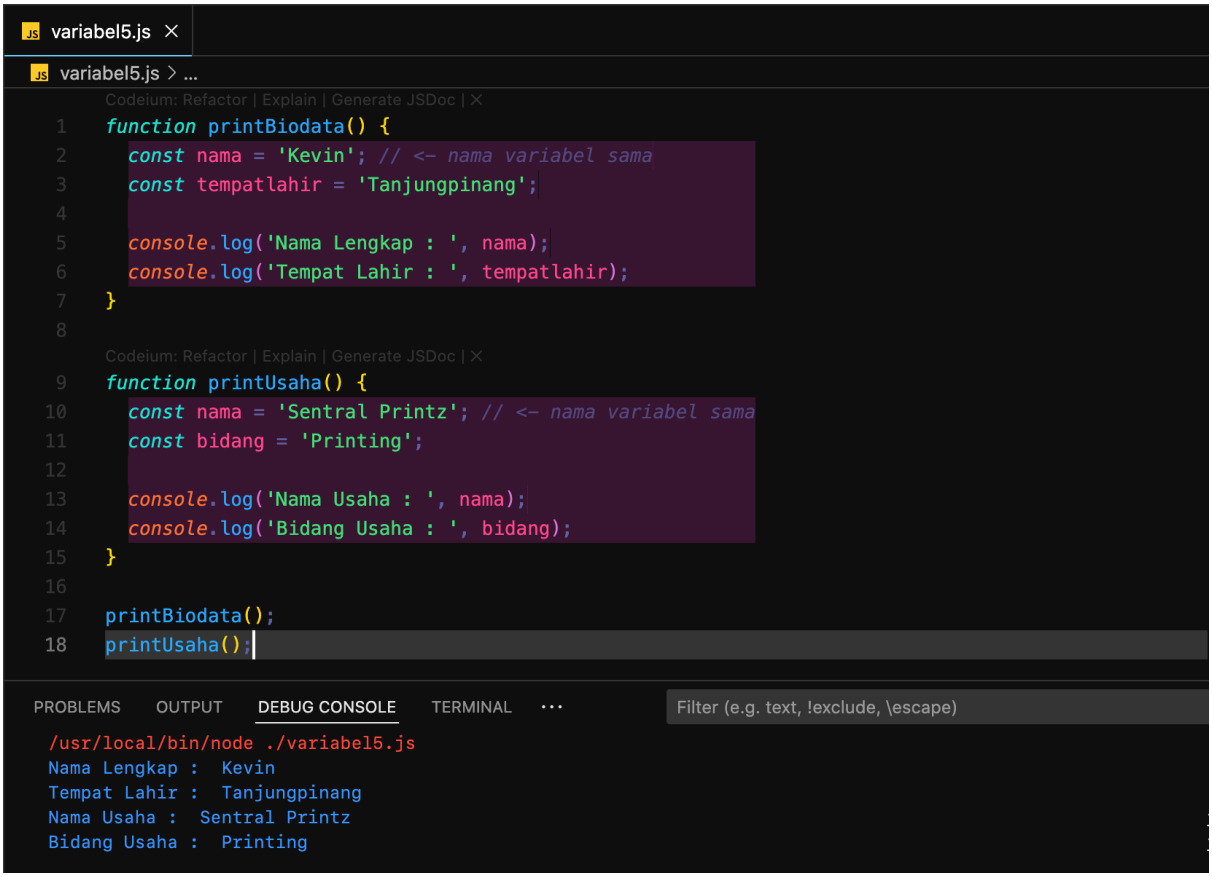
Tidak Boleh Memberikan Nama yang Sama dalam Cakupan yang Sama

Nama variabel haruslah unik dalam cakupannya. Anda tidak bisa menggunakan nama yang sama dengan variabel yang sudah terdefinisi sebelumnya.

Berikut adalah contoh kode yang akan menghasilkan error karena memberikan nama variabel yang sudah terdefinisi sebelumnya.



Anda boleh menggunakan nama variabel yang sama selama cakupannya berbeda, contohnya variabel yang berada dalam sebuah **fungsi** berbeda.



Nama Variabel Hanya Terdiri dari Karakter Tertentu

Nama variabel tidak boleh mengandung karakter **selain** huruf, angka, garis bawah (*underscore*), dan tanda dolar. Berikut adalah contoh penamaan variabel yang benar dan salah.

variabel6.js 7

```
// nama variabel yang benar
const firstName = 'Kevin';
const last_name = 'Perdana';
const $message = 'Hello, World!';
const userId1 = 123;
const userId2 = 456;

// nama variabel yang salah
const first-name = 'Kevin'; // tidak boleh mengandung karakter -
const last name = 'Perdana'; // tidak boleh mengandung spasi
const @message = 'Hello, World!'; // tidak boleh mengandung karakter @
```

Nama Variabel Tidak Boleh Diawali dengan Angka

Walau angka boleh digunakan dalam penamaan variabel, tetapi jika nama variabel diawali dengan angka, nama tersebut dianggap salah. Berikut contohnya.

variabel7.js 6

```
// nama variabel yang benar
const firstName = 'Kevin';
const _secondName = 'Perdana';

// nama variabel yang salah karena diawali dengan angka
const 1stName = 'Kevin';
const 2ndName = 'Perdana';
```

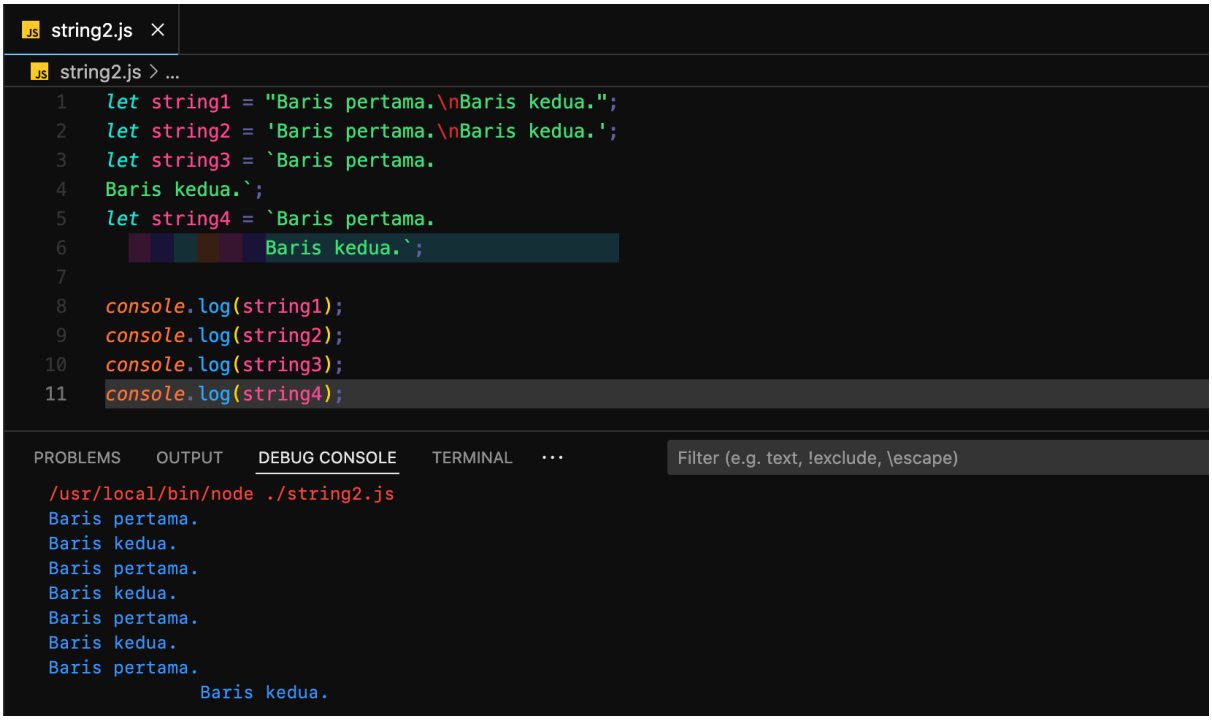
Tipe Data

Dalam JavaScript, ada tipe data primitif yang penting untuk diketahui, yaitu **string**, **number**, **boolean** dan **nilai kosong** (**null** dan **undefined**).

String

String adalah tipe data yang merepresentasikan teks. Data seperti nama, alamat, atau email adalah contoh data yang dikelola dalam bentuk string. Dalam JavaScript, nilai string diapit oleh tanda kutip. Ada tiga jenis tanda kutip yang dapat digunakan untuk membuat nilai string, yaitu **petik tunggal** (*single quote*), **petik ganda** (*double quote*), dan ***backticks*** (tanda backtick).

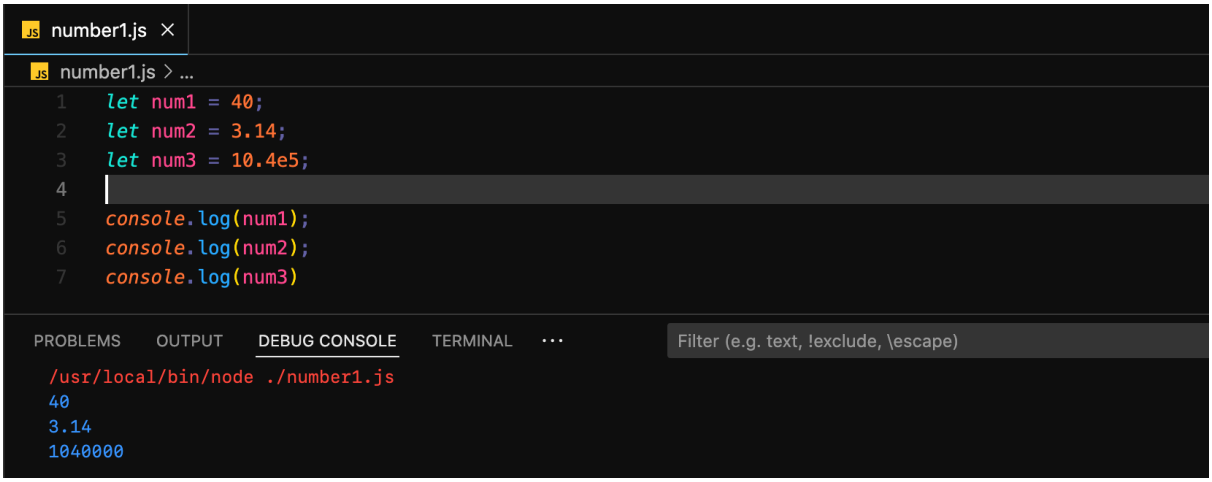
Hampir semua karakter dapat ditulis langsung di antara tanda kutip. Namun, ada beberapa karakter yang memerlukan penanganan khusus, seperti baris baru (karakter yang terbentuk ketika Anda menekan tombol Enter). Baris baru hanya bisa dituliskan secara langsung ketika Anda menggunakan ***backticks***. Untuk tanda **kutip tunggal** atau **ganda**, kita harus menggunakan notasi **\n**.



Backticks sering disebut juga sebagai template literals karena memungkinkan kita menyisipkan JavaScript expressions untuk membentuk nilai string menggunakan notasi **${}**.

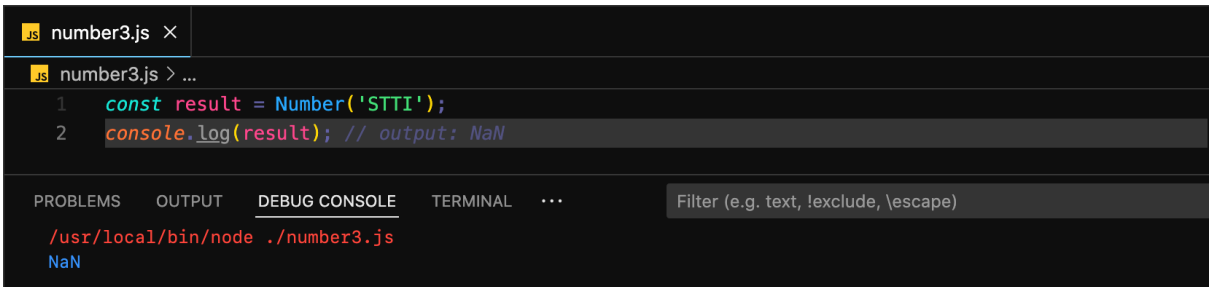
Number

Semua data berupa angka direpresentasikan dalam tipe data number, baik itu bilangan bulat maupun pecahan. Untuk membuat nilai number, kita cukup menuliskan angkanya secara langsung tanpa menggunakan tanda khusus. Berikut adalah contoh nilai number dalam JavaScript.



Selain angka normal, tipe data number juga memiliki nilai spesial, yaitu **Infinity** dan **NaN**. Nilai **Infinity** dihasilkan ketika kita melakukan operasi aritmatika yang tidak terdefinisi, seperti membagi sebuah nilai dengan nol. Contohnya kode di bawah ini.

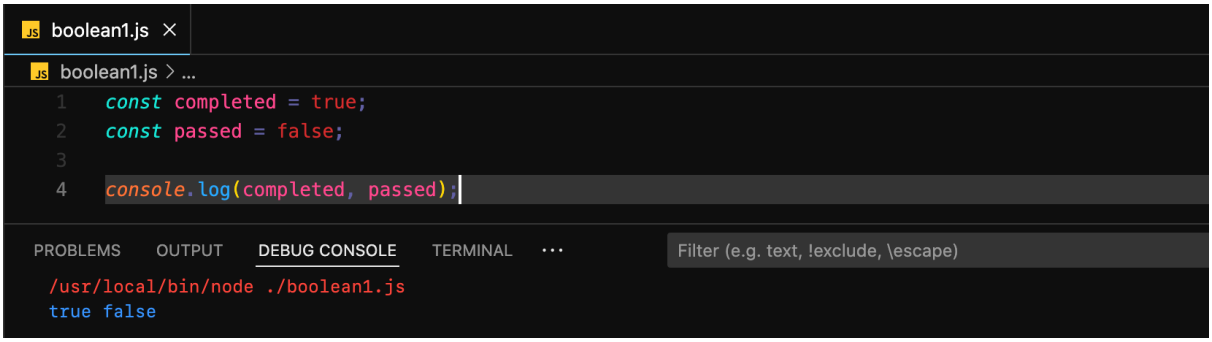
Adapun nilai **NaN (Not-a-Number)** dihasilkan ketika nilai non-numerik diubah ke tipe data number. Contohnya ketika kita mencoba mengonversi string yang bukan angka menjadi number.



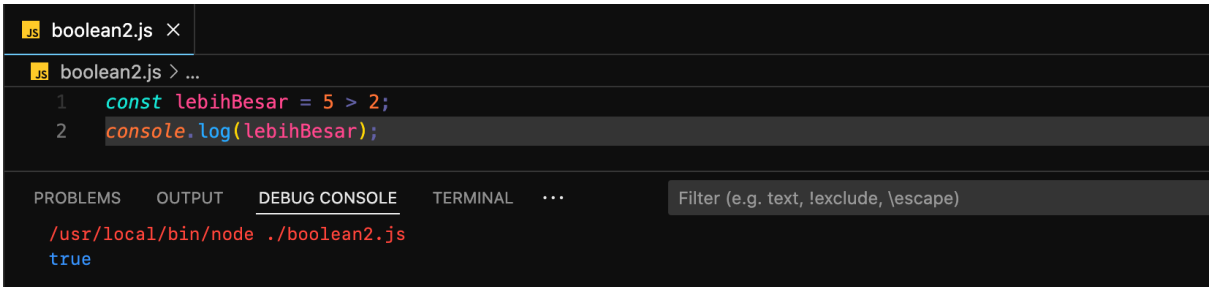
Boolean

Boolean adalah tipe data yang hanya memiliki dua nilai : **true** dan **false**. Boolean umumnya digunakan untuk merepresentasikan “ya” atau “tidak”, “ya” adalah true dan “tidak” adalah false.

Untuk membuat nilai boolean, kita bisa menuliskan **true** atau **false** secara langsung. Contohnya seperti kode di bawah ini.



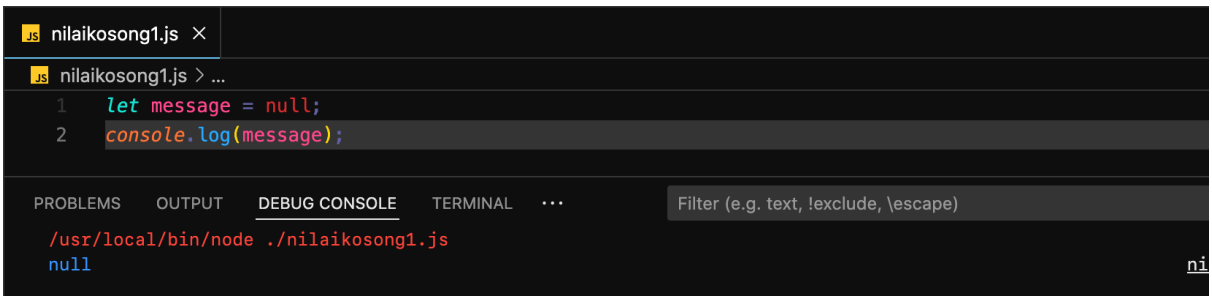
Nilai boolean juga biasa diperoleh dari hasil penggunaan operator perbandingan.



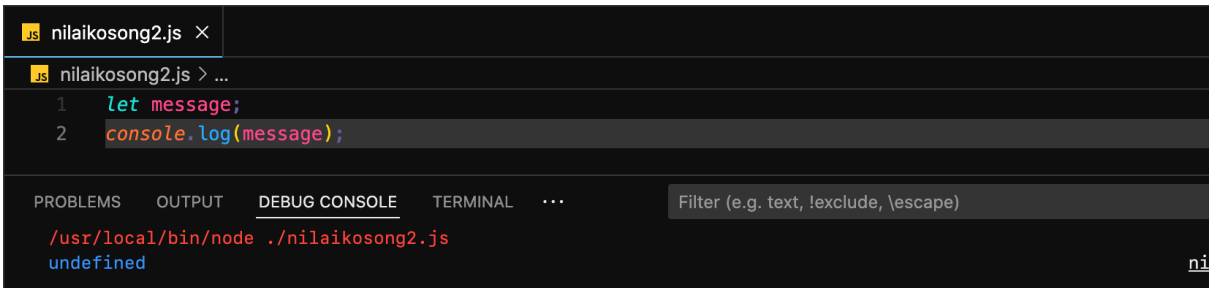
Nilai Kosong

JavaScript memiliki dua nilai spesial yang merepresentasikan nilai kosong, yaitu **null** dan **undefined**. Keduanya digunakan untuk menunjukkan ketiadaan nilai (*the absence of something*).

Null banyak diadopsi dalam berbagai bahasa pemrograman sebagai tipe data standar untuk menunjukkan nilai yang tidak ada. Untuk membuat **null**, kita cukup menulis sintaksis **null**.



Adapun **undefined** hadir dalam JavaScript sebagai nilai implisit ketika kita mendeklarasikan variabel tanpa menginisialisasi dengan nilai apa pun.



Sekilas, null dan undefined terlihat sama, tetapi sebenarnya mereka berbeda. Perbedaan ini dapat terlihat lebih jelas ketika kita membandingkan objek yang propertinya bernilai null dan undefined dalam format **JSON**.

Properti yang diberi nilai **undefined** tidak akan tampak ketika diubah ke **JSON** karena JSON tidak mendukung tipe data undefined. Oleh karena itu, **null** lebih standar untuk menunjukkan nilai kosong.

Mengubah Nilai Antar Tipe Data

Saat mengelola sebuah data dalam JavaScript, seringkali kita perlu mengubah nilai dari satu tipe data ke tipe data lain. Proses ini sangat penting karena dalam berbagai situasi, kita perlu menyesuaikan tipe data dengan konteks atau kebutuhan tertentu, baik untuk keperluan perhitungan, perbandingan, maupun manipulasi data.

JavaScript adalah bahasa pemrograman yang dinamis dan fleksibel, ia menyediakan berbagai cara untuk mengonversi tipe data. Konversi tipe data dapat dilakukan secara eksplisit oleh developer atau secara implisit oleh interpreter. Memahami cara konversi tipe data penting agar dapat menulis kode yang efisien, efektif, dan bebas dari kesalahan.

Konversi Eksplisit

Konversi eksplisit adalah cara yang paling dapat diandalkan untuk mengubah tipe data karena dilakukan dengan instruksi yang jelas, alias eksplisit dari programmer. Berikut adalah beberapa metode umum yang digunakan untuk konversi tipe data secara eksplisit.

1. **Mengubah ke String**

   Untuk mengubah suatu tipe data ke bentuk string umumnya dapat dilakukan dengan dua cara, yaitu menggunakan fungsi **String()** dan method **.toString()**. Berikut adalah contoh dari penggunaan kedua cara tersebut.

Kevin Perdana
https://medium.com/@kevinperdana

2. Mengubah ke Number

Secara umum, untuk mengubah bentuk numerik, seperti “10”, “3.14” dapat dilakukan dengan menggunakan fungsi **Number()**. Berikut contoh penggunaannya.

Selain dengan fungsi **Number()**, ada juga cara yang lebih spesifik, seperti fungsi **parseInt()** dan **parseFloat()**.

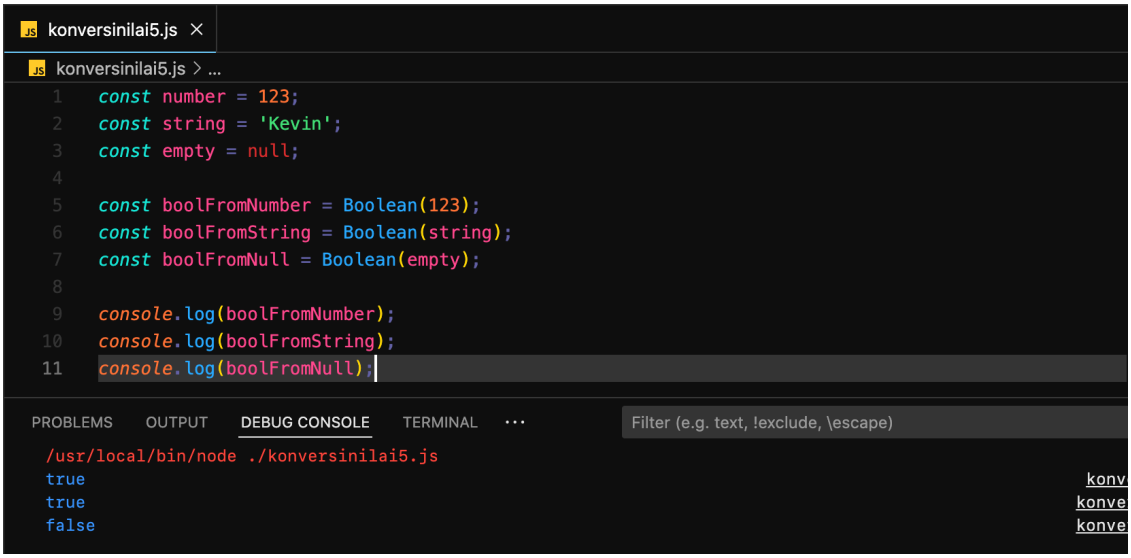
Fungsi **parseInt()** digunakan untuk mengonversi string menjadi bilangan bulat (integer). Fungsi ini memiliki kemampuan untuk membaca karakter satu per satu. Ketika menemukan karakter yang tidak bisa diubah menjadi angka, proses konversi terhenti di sana. Dengan kemampuan ini, **parseInt()** dapat digunakan untuk mengubah nilai string, seperti **"20CM"**, **"64px"**, atau angka dengan satuan lainnya.

Adapun fungsi **parseFloat()** digunakan untuk mengonversi string menjadi angka desimal (floating-point number). Sama seperti **parseInt()**, fungsi ini juga memiliki

kemampuan membaca karakter string satu per satu sehingga dapat mengubah numerik yang mengandung satuan.

3. Mengubah ke Boolean

Untuk mengubah suatu nilai ke tipe data boolean, kita bisa gunakan fungsi **Boolean()**. Sama seperti fungsi sebelumnya, kita cukup memberikan nilai yang akan diubah di antara tanda kurung. Berikut adalah contoh penggunaan fungsi **Boolean()**.

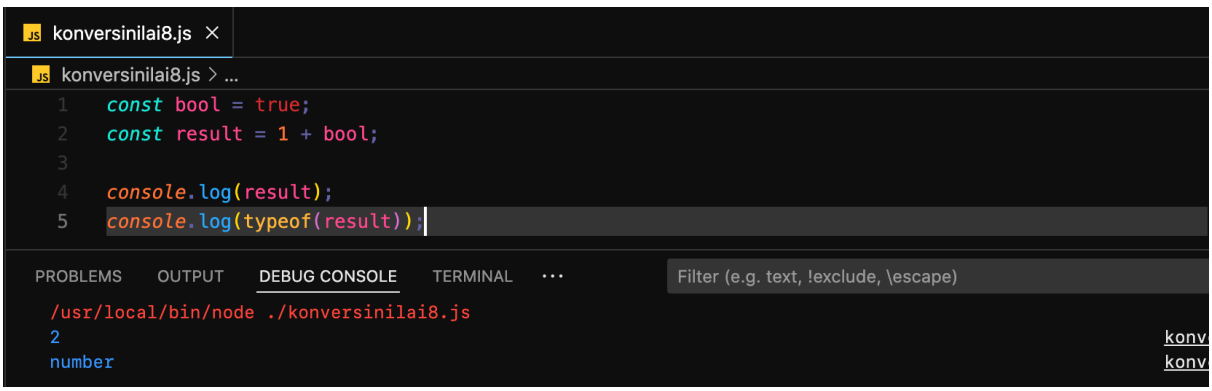


Konversi Implisit

Konversi implisit terjadi ketika JavaScript secara otomatis mengubah tipe data tanpa instruksi eksplisit dari programmer. Ini biasanya terjadi dalam konteks tertentu, seperti operasi aritmatika, perbandingan, dan konteks logika.

Dalam contoh ini, tipe data number (age) secara otomatis dikonversi menjadi string karena operator + digunakan untuk penggabungan string.

Dalam contoh ini, strNumber (yang merupakan string) dikonversi menjadi number karena operator * digunakan untuk operasi aritmatika.



Contoh lain dalam penggunaan operasi aritmatika yang mengubah nilai boolean menjadi number.