# JavaScript

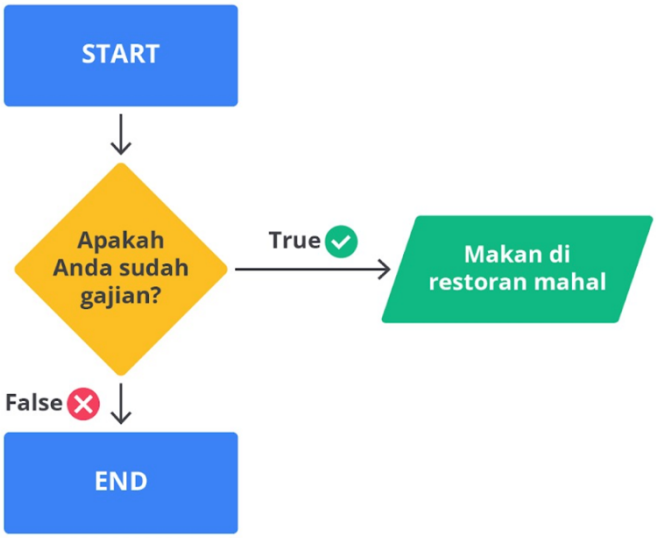
## A Game Changer of Complex Applications

Kevin Perdana

# JavaScript

## Conditional

Conditional flow adalah cara untuk menentukan apakah kode dieksekusi atau dilewatkan. Jika suatu kondisi terpenuhi, kode akan dieksekusi dan kode yang lainnya akan diabaikan. Kondisi ini ditentukan dari inputan yang diberikan oleh pengguna.



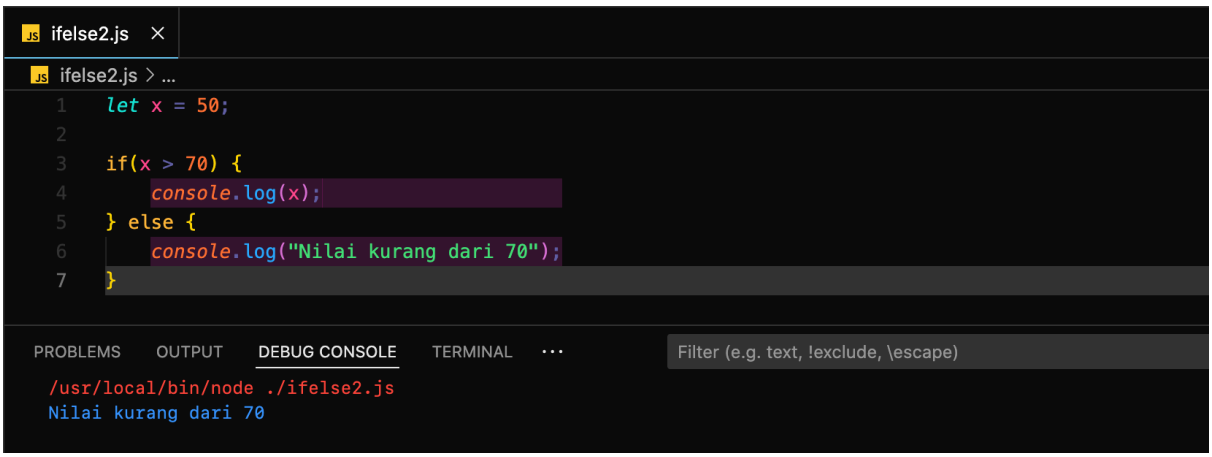

### If Statement

If statement merupakan fundamental statement yang memungkinkan JavaScript untuk membuat keputusan apakah mengeksekusi program atau tidak.

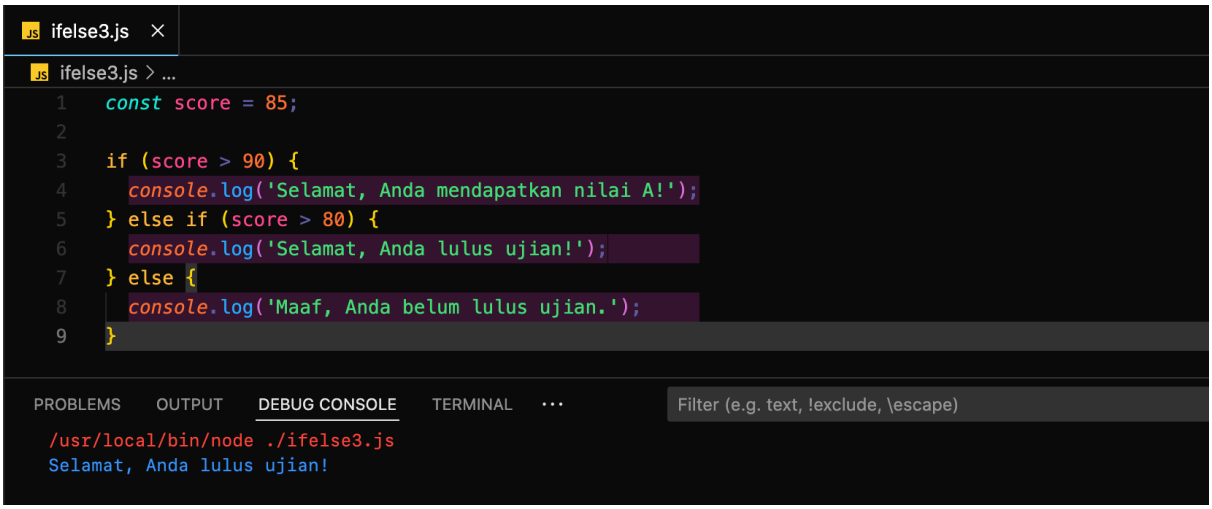


Kevin Perdana
https://medium.com/@kevinperdana

Apakah if statement hanya bisa menangani satu cabang kondisi saja? Jawabannya adalah tidak. Kita bisa menggunakan keyword else untuk menambah pengecekan kondisi lainnya. Perhatikan contoh berikut ini.




Lalu, jika memiliki cabang kondisi lebih dari dua, Anda dapat mengecek beberapa kondisi sekaligus dengan menggabungkan else dan if.

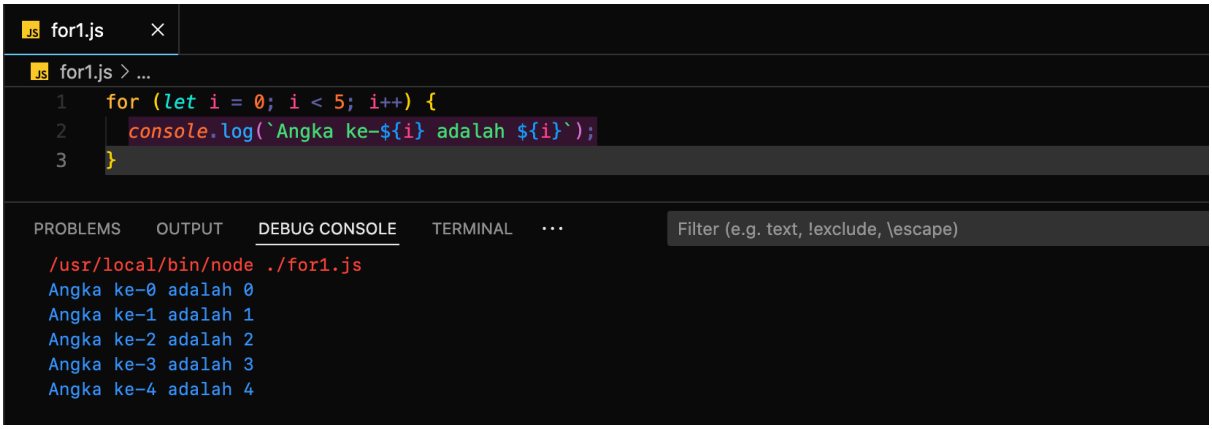


## Looping

Ketika Anda memprogram, ada banyak sekali instruksi yang Anda tulis untuk dieksekusi oleh komputer.

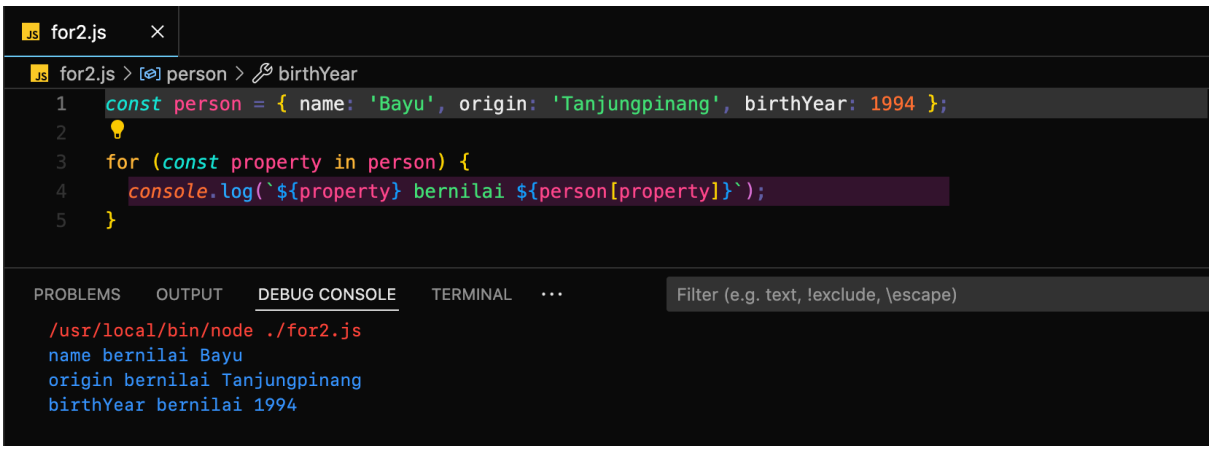
### For Loop

Struktur dari for loop tampak seperti berikut ini.




### For In

For in banyak digunakan untuk pengulangan pada object karena ia dapat melakukan iterasi ke seluruh data di dalam objek. Bahkan, ia juga dapat melakukan iterasi ke properti *inheritance* dari object seperti *length*. Berikut contoh penggunaan for in.




### For Of

Kehadiran for of dimulai pada ECMAScript 2015 (ES6). For of berbeda dengan for in. For of lebih sederhana karena kita tidak perlu memikirkan *property* dan *key*. Perhatikan contoh berikut.

for3.js

```
const names = ['Bebek', 'Ayam', 'Telor', 'Tempe'];

for (const item of names) {
    console.log(item);
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ··· Filter (e.g. text, !exclude, \escape)

```
/usr/local/bin/node ./for3.js
Bebek
Ayam
Telor
Tempe
```

## While

Perulangan di JavaScript tak hanya menggunakan for, tetapi ada cara lainnya yaitu *while statement*.

while1.js

```
let i = 0;

while (i < 5) {
    console.log(`Angka ke-${i} adalah ${i}.`);
    i++;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ··· Filter (e.g. text, !exclude, \escape)

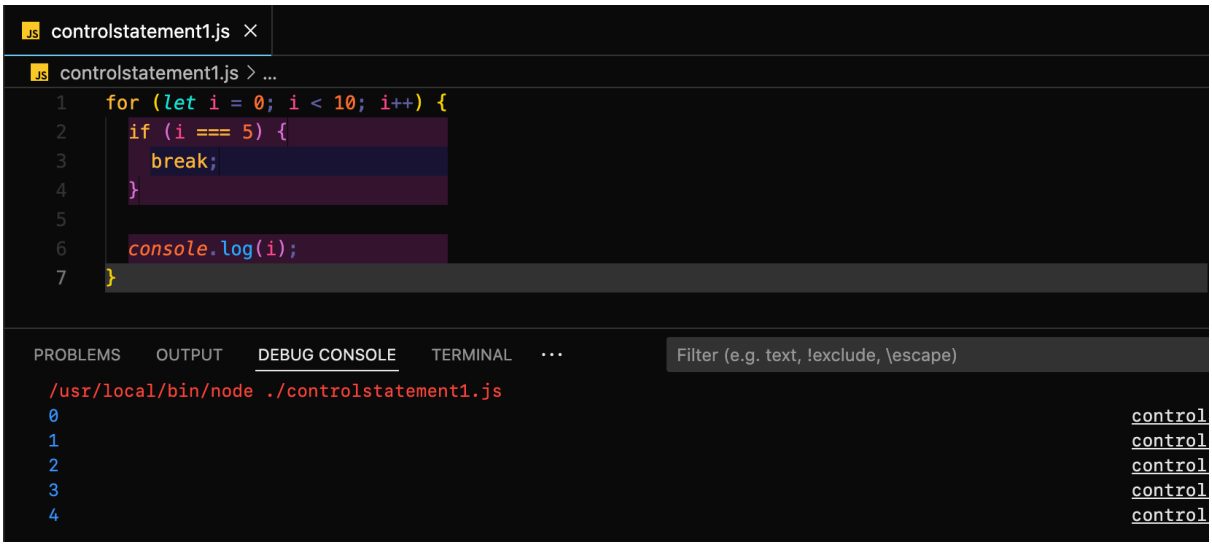
```
/usr/local/bin/node ./while1.js
Angka ke-0 adalah 0.
Angka ke-1 adalah 1.
Angka ke-2 adalah 2.
Angka ke-3 adalah 3.
Angka ke-4 adalah 4.
```

## Control Statement

Ketika melakukan perulangan, ada yang namanya control statement. Control statement berfungsi untuk menghentikan eksekusi kode. Beberapa statement yang masuk ke dalam kategori control statement adalah *break* dan *continue*.

### Break

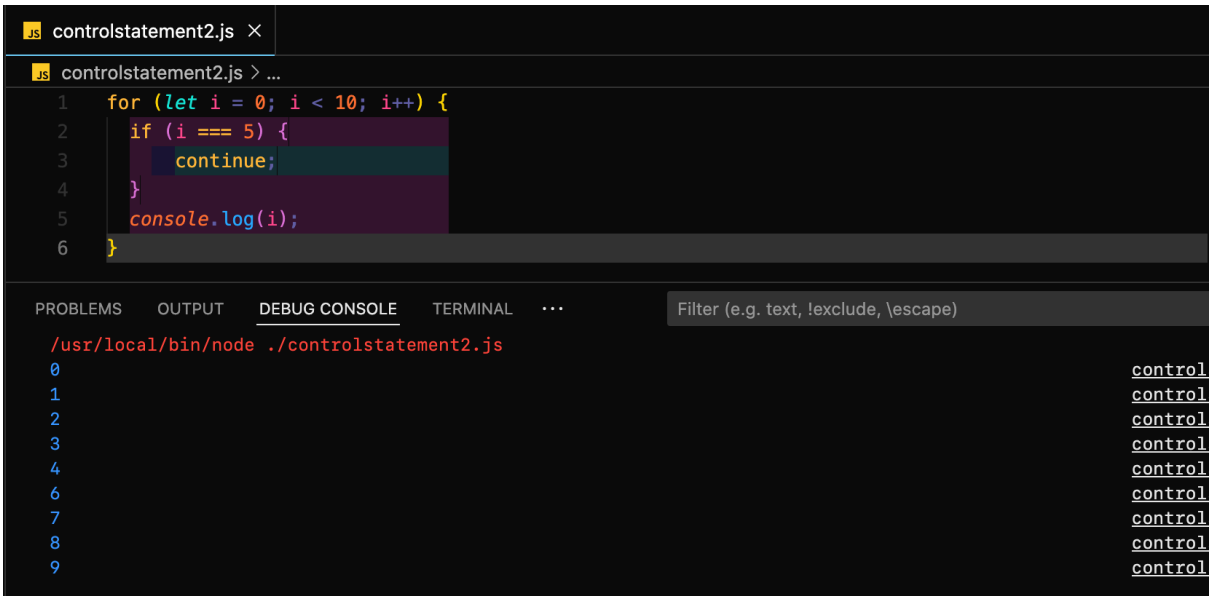
Break statement adalah cara kita untuk memberitahukan interpreter yang sedang mengeksekusi kode untuk berhenti dan langsung berpindah ke akhir dari percabangan atau perulangan.




Kode di atas akan mencetak angka 0 hingga sepuluh kali, tetapi akan terhenti ketika nilai variabel i sama dengan 5. Hal ini disebabkan oleh adanya statement break. Break akan menghentikan proses perulangan.

**Continue**

Continue statement sama seperti break statement. Namun, alih-alih menghentikan eksekusi program, continue akan melanjutkan iterasi ke iterasi berikutnya. Continue statement hanya dapat digunakan di dalam body looping. Perhatikan contoh berikut ini.



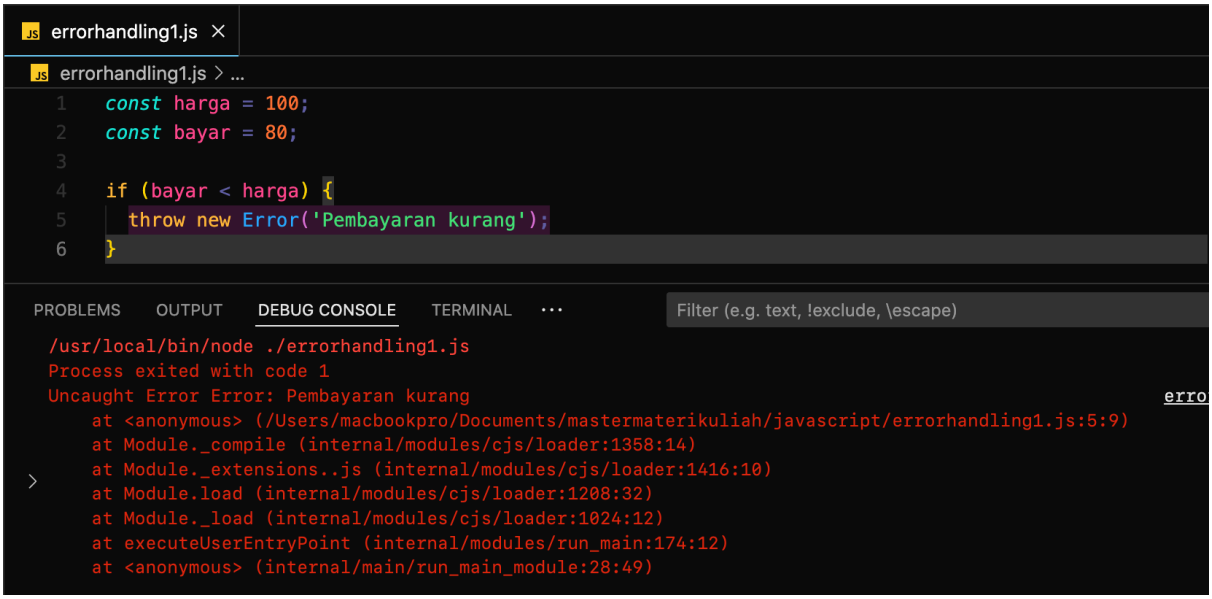

Looping akan berjalan seperti biasa. Namun, ketika nilai variabel i sama dengan 5, iterasi akan dihentikan dan lanjut ke iterasi berikutnya sehingga di terminal/console tidak akan menampilkan angka 5.

## Error Handling

Error yang dibiarkan dan tidak ditangani akan menyebabkan *crash* pada program yang dibangun. JavaScript memiliki cara untuk menangani error tersebut yang disebut dengan error handling. Error handling dapat mencegah *crash* pada program ketika terjadi error yang disebabkan oleh kesalahan syntax atau error lainnya.

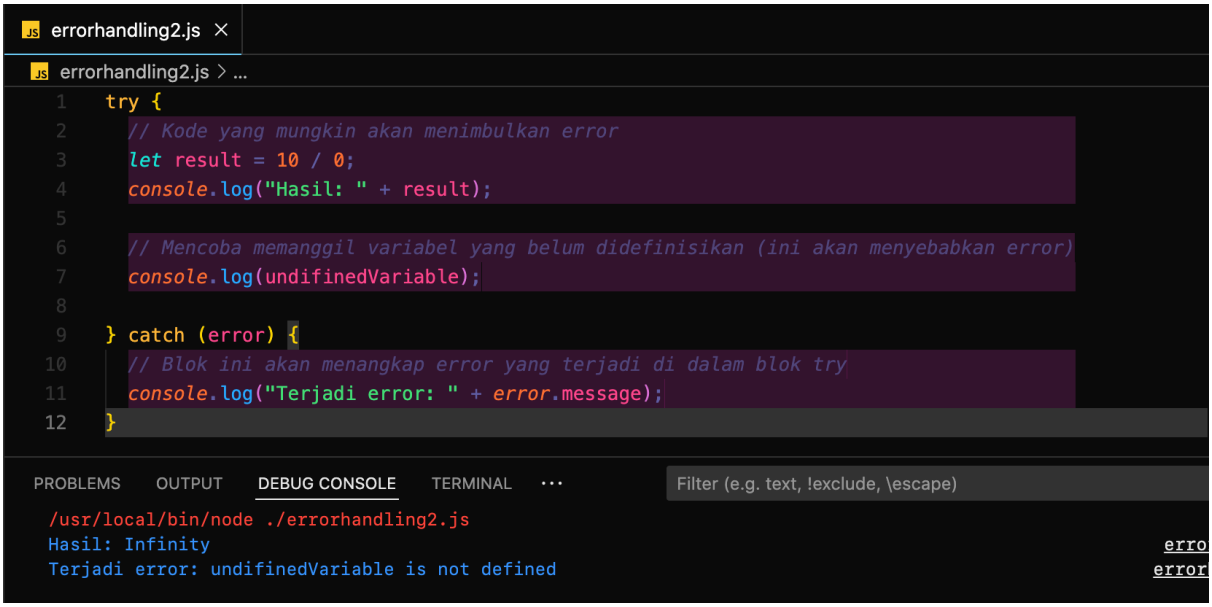
### Throwing Error

"Throwing error" adalah istilah yang sering digunakan dalam pemrograman untuk merujuk pada proses "melempar" atau memunculkan suatu kesalahan (error) ketika ada masalah yang terdeteksi dalam kode program.



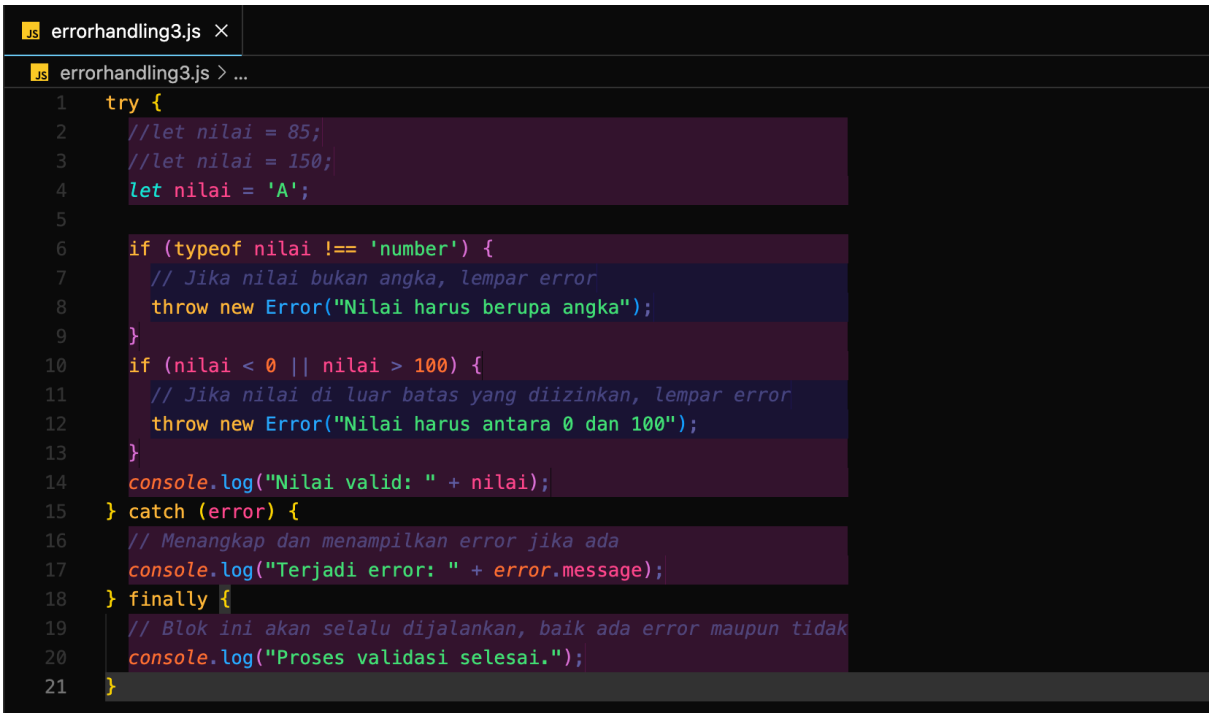
## Try-Catch

Try-catch merupakan cara yang dimiliki JavaScript untuk menangani error. Try-catch memiliki dua blok utama yaitu try dan catch. Try merupakan blok kode yang akan menangani error, sedangkan catch merupakan blok kode yang dibangkitkan ketika terjadi error di dalam blok try.

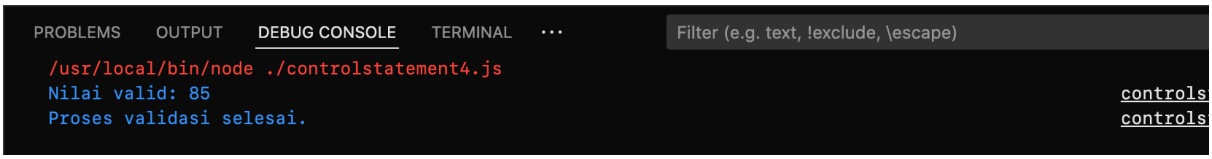



## Finally

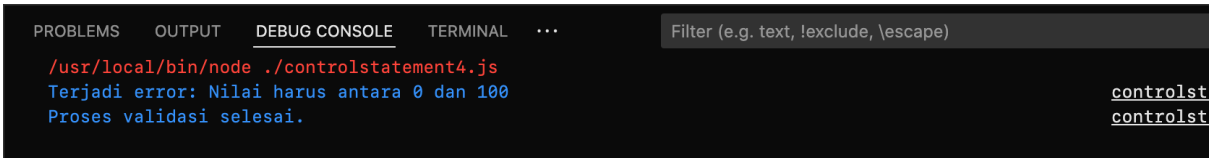
Finally adalah blok kode yang berada di akhir try-catch. Bilamana catch dieksekusi hanya ketika ada error di dalam blok try, blok yang ada di finally akan selalu dieksekusi. Simak contoh di bawah ini.



Output :
let nilai = 85;




let nilai = 150;




let nilai = 'A';