**Membuat Rekursif**

Seperti yang kita ketahui, rekursif adalah pemanggilan fungsi yang berulang. *Function* yang menerapkan cara rekursif disebut juga dengan *recursive function*. Jika *recursive function* tersebut memanggil dirinya sendiri, akan terjadi *infinity recursion* (rekursi tak hingga). Maka dari itu ada beberapa hal yang harus diperhatikan dalam membuat *recursive function*.

Algoritma rekursif mempunyai 2 komponen utama, yaitu:

1. **Base Case**  
   Kasus dasar untuk menyelesaikan permasalahan. *Base case* akan dikunjungi jika rekursi berakhir (kondisi untuk berhenti terpenuhi), serta mengembalikan nilai tanpa melakukan rekursi kembali.
2. **Recursion Call**  
   Permasalahan yang ada tentunya akan diperkecil dengan melakukan pemanggilan *function* itu sendiri (*recursion call*). Permasalahan dapat diperkecil dengan mengurangi atau memecahkan data *input* pada setiap pemanggilannya hingga mencapai *base case*.

Struktur dasar *function* rekursif adalah sebagai berikut:

function namaFuncRekursif() {

if (condition) {

// Base case

} else {

// Recursion call

namaFuncRekursif();

}

}

Untuk memahami hal tersebut menjadi lebih mudah, kita akan pelajari menggunakan contoh kasus sederhana. Misalnya kita akan mencari hasil dari angka faktorial yaitu 4 faktorial.

Sebelumnya kita buat terlebih dahulu *function* yang dengan memiliki sebuah parameter n sebagai angka faktorial yang dicari.

function faktorial(n) {

// kode di sini

}

Jika dijabarkan solusi penyelesaian untuk mencari angka faktorial, maka akan menjadi seperti ini:

4! = 4 \* 3!

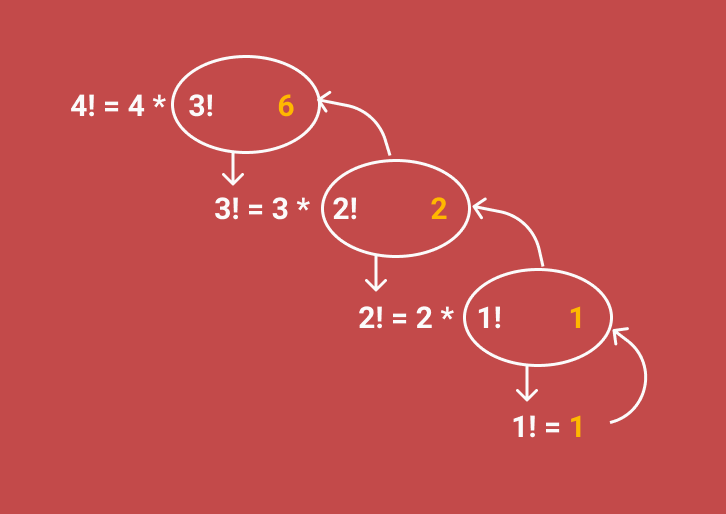
3! = 3 \* 2!

2! = 2 \* 1!

1! = 1

4! = 4 \* 3 \* 2 \* 1

4! = 24



Pada gambar di atas, terlihat sebuah pola yaitu angka faktorial yang dicari akan dikalikan dengan angka faktorial di bawahnya (contohnya 4 \* 3!) dan seterusnya sampai dikalikan dengan 1. Terlihat sama bukan cara penyelesaiannya?

n \* n-1

Maka dari itu kita dapat membuat *recursion call* yang akan mengembalikan suatu nilai seperti di bawah ini:

return n \* faktorial(n - 1);

Tapi ternyata tidak sampai di situ saja. Masalah tersebut harus ditentukan solusi terkecilnya. Jika tidak, maka akan terjadi *infinity recursion*.

Berdasarkan kasus di atas, masalah terkecilnya adalah mendapatkan bilangan 1 setelah bilangan lainnya diperkecil. Maka kita dapat membuat *base case* dengan mengembalikan angka 1.

return 1;

Untuk mencapai *base case*, tentunya harus memenuhi suatu kondisi tertentu. Angka 1 akan dikembalikan jika angka faktorialnya adalah 1. Sehingga keseluruhan *function*-nya menjadi seperti ini:

function faktorial(n) {

if (n == 1) {

return 1;

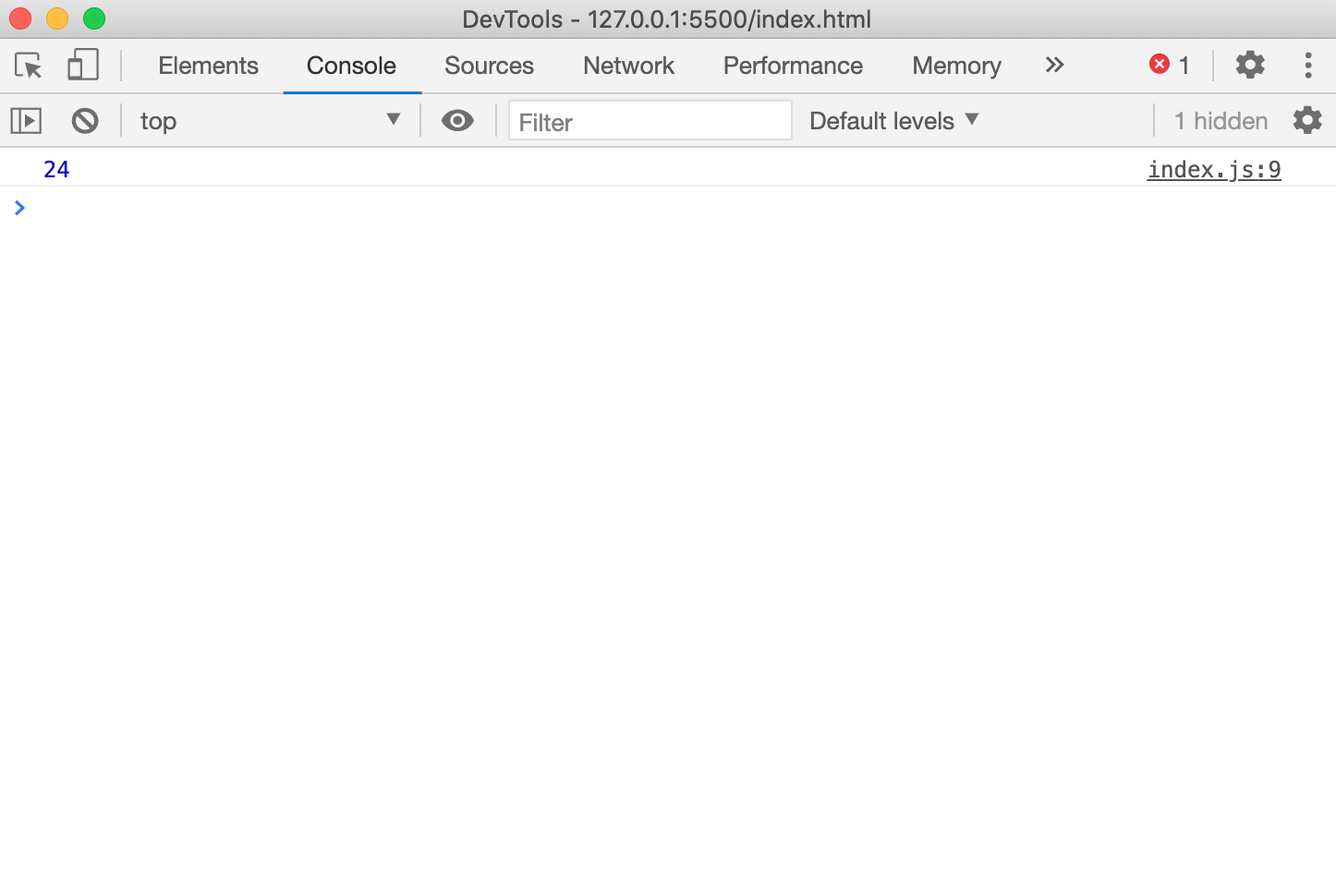
} else {

return n \* faktorial(n - 1);

}

}

console.log(faktorial(4))



Maka jika kita jalankan fungsi faktorial(4), proses fungsinya akan berjalan seperti ini:

4\*faktorial(3) // faktorial(3) = 3\*faktorial(2)

4\*3\*faktorial(2) // faktorial(2)= 2\*faktorial(1)

4\*3\*2\*faktorial(1) //faktorial(1) = 1 --> masuk ke kondisi n==1

sehingga didapatkan hasil

4\*3\*2\*1=24