

hacktiv8

/ phase-0-activities

Watch

41

Star

74

Fork

175

<> Code

Issues 0

Pull requests 0

Projects 0

Insights

Join GitHub today

GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

Branch: master

phase-0-activities / modules / js-function-recursive.md

Find file

Copy path

 adhywiranata Update js-function-recursive.md

6e19a8a on Oct 30, 2017

1 contributor


182 lines (138 sloc)


7.47 KB

Raw

Blame

History





# JavaScript Recursive Function

## Objectives

- ☐ Mengetahui pembuatan dan cara kerja fungsi Rekursif
- ☐ Memahami cara membuat fungsi rekursif sederhana

## Learnings

### Fungsi (Function) Review

Sedikit kita review apa yang telah kita pelajari minggu lalu, fungsi atau metode (metode) tentunya kita sudah pakai beberapa kali saat berlatih dengna tipe data dan objek. Ia bekerja baik dengan statement kondisi, sehingga kkita tidak perlu repot berulang kali memanggil/menulis beberapa baris kode yang sama atau berpola berulang kali.

Sebuah fungsi dibuat dengan ekspresi yang dimulai dengan kata kunci `function` . Fungsi dapat memiliki beberapa parameter (or argumen), atau bahkan tanpa parameter sama sekali. Fungsi dalam JS juga merupakan pendefinisian variabel umum yang mana nilainya adalah `function` .

```
var printPrint = function() { // without parameter
  console.log("Print")
};
printPrint() // call it

var printStarred = function(text) { // with parameter
  console.log("*** " + text + " ***")
};
printStarred("Super Star")

function printCircle(text) { // shorthand to create function
  console.log("ooo " + text + " ooo")
}
printCircle("Circling Circle")
```

Wajarnya adalah fungsi perlu mengembalikan nilai ( `return a value`) yang bisa kita ambil berikutnya. Seringkali kita ingin untuk mendapatkan nilai yang dikembalikan tersebut, maka kita assign ke sebuah variabel, di luar `function` .

```
var timesTwo = function(x) {
  return x * 2;
};
var result = timesTwo(8) // 16
console.log(result)
```

Nah, berdasarkan apa yang telah kita review mengenai function, saatnya kita masuk ke dalam rekursi, atau fungsi rekursif.

### Rekursi (Recursion)

Kita bisa membuat function yang memanggil dirinya sendiri, disebut recursion. Memungkinkan kita untuk menulis fungsi dengan ekspektasi hasil yang sama namun cara/gayanya berbeda. Ini akan berhubungan nanti tentang bagaimana kita menstrukturkan langkah logika kita, atau algorithm.

Mungkin rekursif bagi sebagian orang menjadi satu hal yang dianggap mengerikan atau membingungkan. Tapi apabila kita sudah mengenal bagaimana cara atau alur dari fungsi rekursif, ketakutan atau kebingungan itu pasti bisa hilang.

Berikut ini contoh paling sederhana dalam implementasi fungsi rekursif:

```
function numberSum(num) {
  if(num == 1) {
    return 1;
  }
  else {
    return num + numberSum(num - 1);
  }
}

console.log(numberSum(5)); // 5 + 4 + 3 + 2 + 1 = 15
```

Fungsi di atas akan mengembalikan nilai berupa hasil penambahan dari 5 + 4 + 3 + 2 + 1. Bagaimana cara kita melakukannya dengan fungsi di atas, hanya berbekal satu parameter yaitu angka 5? Jika diperhatikan, fungsi numberSum akan memanggil kondisi. Jika angka yang diinput adalah 1, maka fungsi akan selesai dan mengembalikan angka 1. Namun, jika lebih dari satu, fungsi akan mengembalikan nilai berupa value dari variabel `num` , dan menambahkannya dengan `numberSum(num - 1)` . Mungkin beberapa akan bingung sejenak tentang apa yang terjadi. Jadi, inilah kenapa fungsi ini disebut rekursif, atau sebuah fungsi yang memanggil dirinya sendiri. Untuk dapat mendapatkan hasil yang dibutuhkan, maka pemanggilan ulang fungsi tersebut wajib dilakukan, namun jangan lupa untuk mengubah nilai parameternya.

Jika kita pecah fungsi di atas secara proses, maka fungsi yang berjalan adalah `numberSum(5)` , kemudian mengembalikan nilai `5 + numberSum(4)` , yang akan mengembalikan nilai `4 + numberSum(3)` , dan seterusnya hingga mengembalikan nilai `1` dan keluar dari rekursif, sehingga pada akhirnya akan menghasilkan nilai 5 + 4 + 3 + 2 + 1.

⚠ seperti layaknya looping, waspadai juga pemanggilan rekursif yang tidak akan pernah selesai!

Kamu bisa mencoba kode diatas [disini](#).

*Kenapa kita menggunakan fungsi rekursif? Bukankah hal ini bisa kita selesaikan dengan looping for atau while saja?*

Tentu, kamu bisa menyelesaikan kasus di atas dengan looping. Namun, banyak kasus yang sangat membutuhkan rekursif, atau beberapa kasus akan menjadi lebih efisien dari segi jumlah baris kode apabila menggunakan kode rekursif.

Dibawah ini ada contoh fungsi rekursif yang lebih advanced, kamu bisa coba dua contoh dibawah untuk memperkuat pemahaman kamu tentang rekursif!

### Contoh Fungsi Rekursif Untuk Kasus Perpangkatan

```
// Perpangkatan
function power(base, exponent) {
  if (exponent == 0)
    return 1;
  else
    return base * power(base, exponent - 1);
}
console.log(power(3, 3)); // 3 ** 3 = 27
```