Modul praktikum - Minggu 09 - *Functions* bagian 1

Dosen pengampu: **Vika Fitratunanny Insanittaqwa S.Kom., M.Kom**
Asisten mata kuliah: **Bayu Adhitya Wibisana - (10191015)**

Tujuan:

- Mampu memahami terminologi dasar terkait *functions*.
- Mampu mendefinisikan suatu *function* di JavaScript
- Mampu melakukan pemanggilan (*invocation*) suatu *function*.

> Tips belajar bahasa pemrograman adalah mengetik ulang perintah yang kita temukan di buku atau di internet, lalu kita ubah-ubah untuk menguji pemahaman kita sudah tepat atau belum. Faktor bermain-main dan eksplorasi sangat diperlukan untuk memahami setiap perintah bahasa pemrograman yang kita pelajari. Setiap potongan kode di bawah dapat ditulis dalam berkas `.js` lalu dapat di-*running* dengan Node.js.

Fungsi merupakan suatu bentuk abstraksi untuk mewakili proses yang menghubungkan input dan output. Beberapa bahas pemrograman lain yang cukup lama seperti C, C++, dan Fortran, menyebuh fungsi sebagai *subroutine* atau *procedure*.

Maksud dari proses abstraksi tersebut adalah, dengan menggunakan fungsi kita bisa mewakili suatu prosedur yang panjang dan dipakai berulang dan menamakan prosedur tersebut sebagai suatu fungsi. Hal ini tentunya akan mempermudah pekerjaan kita, ketika kita dihadapkan dengan masalah yang hampir serupa, dan kita cukup menggunakan (memanggil) fungsi tersebut tanpa harus mengetik berulang-ulang proses yang diwakili fungsi tersebut.

Kemampuan berabstraski sangat diperlukan untuk bisa menyusun fungsi yang benar dan efisien. Pembuatan fungsi dimulai dengan menuliskan prosedur secara modular (terpisah-pisah), setelah terlihat strukturnya kita bisa menggabungkan prosedur tersebut ke dalam suatu fungsi.

Dengan fungsi, kita juga dapat mengimplementasi berbagai macam algoritma-algoritma pemecahan masalah. Di akhir pertemuan tentang fungsi ini, kita akan mencoba untuk menyusun algoritma sederhana

Beberapa hal yang akan kita pelajari di sesi ini adalah:

1. Istilah-istilah dasar dalam fungsi
2. Cara mendeklarasikan fungsi
3. Pemanggilan fungsi

Istilah-istilah dasar dalam fungsi

- *Invoked*: merupakan peristiwa saat kita memanggil fungsi yang telah didefinisikan
- *parameterized*: keadaan suatu variable input atau variabel yang dimuat oleh suatu fungsi bersifat lokal (telah terparameterisasi)
- *parameter*: variable input yang dimiliki oleh fungsi yang berfungsi sebagai penanda (*identifier*)

- *argument*: merupakan nilai yang akan diinputkan ke dalam fungsi atau nilai yang diberikan ke variabel input. Proses pemberian *argument* terjadi saat fungsi dipanggil (*function invocation*)
- *method*: fungsi yang secara khusus didefinisikan sebagai *property* suatu *object*
- *constructor*: fungsi yang secara khusus didefinisikan untuk melakukan pendeklarasian suatu *object*.
- *closure*: merupakan sifat dari fungsi di JavaScript yang mana suatu definisi fungsi dapat disisipkan dalam suatu fungsi lain dan fungsi yang didefinisikan tersebut dapat mengakses variable di fungsi yang disisipi.

Function declaration

- menggunakan kata kunci `function`

  func-def-with-function.js

```
// Mencetak `property name` dan `property value` dari suatu object o
function printops(o) {
  for (let p in o) {
    console.log(`${p}: ${o[p]}`);
  }
}


// Melakukan penghitungan jarak antara dua titik (x1, y1) dan (x2, y2)
function distance(x1, y1, x2, y2) {
  let dx = x2 - x1;
  let dy = y2 - y1;
  return Math.sqrt(dx*dx + dy*dy);
}

// FUngsi rekursif yang memanggil dirisnya sendiri dan digunakan untuk
// menghitung faktorial
function factorial(x) {
  if (x <= 1) return 1;
  return x * factorial(x-1);
}

let o = {"apple": 5000, "mango": 7000, "pineapple": 15000};
printops(o);

console.log();
let dist = distance(0, 0, 5, 12);  // => 13 = sqrt(5^2 + 12^2)
console.log("dist: ", dist);


console.log();
console.log("factorial(5): ", factorial(5));   // => 120 = 5 x 4 x 3 x 2 x 1
```

- variable yang bernilai suatu *function expression*

func-def-with-func-expression.js

```
// Function expression berikut mendefinisikan nilai kuadrat dari argument
// yang diberikan kepada fungsi tersebut
const square = function(x) { return x*x; };
console.log(square(5));

// Function expression dapat memuat nama yang dapt digunakan
// untuk mendefinisikan fungsi rekursif
const f = function fact(x) { if  (x <= 1) return 1; else return x*fact(x-1);
};
console.log(f(5));


// Function expression dapat digunakan sebagai suatu argumen untuk fungsi
// yang lain
let result = [3, 2, 1].sort(function(a, b) { return a - b; });
console.log(result);

// Function expression dapat juga didefinisikan dan langsung di panggil
// dengan nilai argument
let tensquared = (function(x) { return x*x; }(10));
console.log(tensquared);
```

- Jalan pintas (*shortcut*) untuk mendefinisikan fungsi: *arrow function*

func-def-with-arrow-func.js

```
let result;

// Mendefinisikan fungsi jumlahan dua buat variabel x dan y menggunakan
arrow
// function ( () => {} )
const sum = (x, y) => { return x + y; };
result = sum(25, 75);
console.log(result);

// Tanpa kurung kurawal dan kata kunci `return`
const sumShort = (x, y) => x + y;
result = sumShort(25, 75);
console.log(result);

// Untuk arrow function dengan satu parameter tidak perlu menggunakan kurung
const polynomial = x => x*x + 2*x + 3;
result = polynomial(1);
console.log(result);


// Untuk arrow function tanpa argumen dapat menggunakan kurung saja
const greeting = () => "Hi!";
```

```
result = greeting();
console.log(result);

// -- Penggunaan arrow function sebagai argument suatu array method
// Digunakan untuk menghapus element yang bernilai null
let filtered = [1, null, 2, 3].filter(x => x !== null);
console.log(filtered);

// Digunakan untuk melakukan kuadrat setiap element array tanpa harus
// melakukan perulangan menggunakan array method `map()`
let squares = [1, 2, 3, 4].map(x => x*x);
console.log(squares);
```

- Fungsi yang memuat fungsi (*nested* function)

  func-def-with-nested-func.js

```
// Fungsi berikut akan menghitung sisi miring dari suatu segitiga siku-siku
// dengan diberikan panjang sisi tegak dan sisi mendatar.
// Fungsi yang memuat definisi suatu fungsi (square). Disini fungsi square
// dapat memanggil variabel tempat fungsi tersebut didefinisikan, yaitu
// dapat mengakses variabel a dan b dari fungsi hypotenuse
function hypotenuse(a, b) {
  function square(x) { return x*x; }
  return Math.sqrt(square(a) + square(b));
}

let result = hypotenuse(5, 12);
console.log(result);
```

Function invocation

- fungsi dipanggil sebagai fungsi
  Pada bagian sebelumnya kita secara tidak langsung sudah melakukan pemanggilan fungsi, yaitu menggunakan nama fungsi dan dikuti kurung buka lalu daftar arguments diakhiri dengan kurung tutup.

- fungsi dipanggil sebagai *method*
  Suatu fungsi yang didefinisikan di dalam suatu object dapat dipanggil sebagai suatu *methods*

  func-invoke-with-method.js

```
// Object calculator yang dapat menyimpan input penjumlahan dan hasil
// penjumlahannya
let calculator = {
  operand1: 1,
  operand2: 1,

  add() {
    this.result = this.operand1 + this.operand2
```

```
  }
}

// Kita berikan input bilangan yang akan dilakukan penjumlahan
calculator.operand1 = 5;
calculator.operand2 = 7;

// Lakukan proses penjumlahan setelah memasukan inpuit
calculator.add();

// Tampilkan hasil penjumlahan
console.log(calculator.result);
```

- pemanggilan fungsi secara tidak langsung menggunakan `call()` dan `apply()`

  func-invoke-with-call-and-apply.js

```
let arrayOfNumbers = [4, 2, 3, 1, 1, 2, 3, 5, 1, 3, 3];
let biggest;

// Pemanggilan fungsi Math.max secara langsung
biggest = Math.max(...arrayOfNumbers)
console.log(biggest);

// Kita menggunakan Math object (object bawaan dari JavaScript)
// yng memuat semua type bilangan. Kita terapkan fungsi Math.max
// ke arrayOfNumber tetapi secara tidak langsung menggunakkan .appyly
// Math.max akan menjadi method dari Math object dan juga arrayNumbers
// akan menjadi argument dari Math.max dan kita tidak perlu melakukan
// unpack menggunakan spread operator
biggest = Math.max.apply(Math, arrayOfNumbers)
console.log(biggest);

// with .call you have to use spread operator to unpack the elements
biggest = Math.max.call(Math, ...arrayOfNumbers);
console.log(biggest);
```

Tugas (Exercise - 06)

> Laporan harus ditulis dan dikumpulkan dalam bentuk berkas *markdown* atau berkas berekstensi `.md`. Apabila laporan memuat lebih dari satu berkas, misal memuat berkas gambar `.png` atau `.jpg`, maka berkas disatukan menjadi berkas `.zip`.
> **PASTIKAN** berkas `md` sudah dilakukan *preview*, sehingga kode *markdown* bisa di-*preview* dengan benar.
> Format penamaan file: `NIM_NAMA.md` atau `NIM_NAMA.zip` (boleh nama lengkap atau nama panggilan).
>
> **Contoh format laporan atau jawaban (`NIM_NAMA.md`)**
> Nama: [NAMA LENGKAP]
> NIM: [NIM]

> 1. (Jawaban nomor 1)
> 2. (Jawaban nomor 2)

1. [30 poin] Sebutkan 3 kegiatan yang kalian lakukan sehari-hari yang dapat diubah menjadi suatu fungsi. Implementasi fungsi tersebut kedalam JavaScript. Berikan juga contoh penggunaannya.

   Berikut contoh salah satu kegiatan yang mungkin.

   ```
   // Berikut adalah fungsi untuk menggambarkan kegiatan makan
   function angkatSendok(n) {
     for (let i = 0; i < n; i++>) {
       console.log(`menangangkat sendork yang ke-${i+1} kali`);
     }
   }

   angkatSendok(4);
   ```

2. [70 poin] Buatlah suatu fungsi untuk mengkonversi suatu bilangan ke bentuk string yang merupakan pembacaan bilangan tersebut:

   - 1, menjadi 'satu',
   - 2, menjadi 'dua',
   - 3, menjadi 'tiga',
   - 4, menjadi 'empat',
   - 5, menjadi 'lima',
   - dst.

   Fungsi yang akan kalian buat tersebut harus mampu mengkonversi hingga bilangan ke-100.