Nama : Fitra Ilyasa

NIM : 120140048

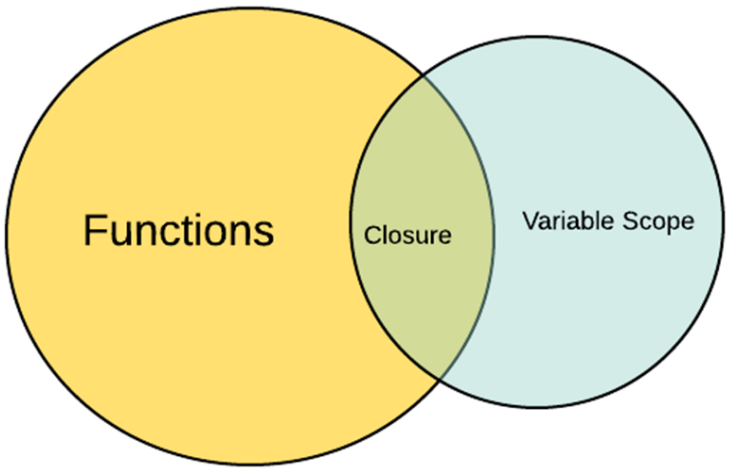
Kelas : Pengembangan Aplikasi Mobile RA

TUGAS 2 | JAVASCRIPT LANJUTAN

1. Closure (Fungsi Penutup)

a. Definisi

Closure pada JavaScript mengadaptasi konsep matematika yaitu lambda calculus. Di JavaScript sendiri, closure merupakan sebuah fungsi yang dieksekusi oleh fungsi lainnya (nested functions) sehingga fungsi tersebut dapat mengakses/merekam variable didalam lexical scope pada fungsi induk(parent function).



https://edward-huang.com/tech/javascript/closure/functional-programming/programming/2020/02/13/what-is-really-so-special-about-javascript-closure/

Closure adalah kombinasi dari fungsi dan lexical environtment dimana fungsi itu dideklarasikan. Berikut beberapa poin penting dari sebuah closure yaitu:

- Closure dibuat ketika ada fungsi yang me-return/mengembalikan nilai fungsi lainnya.
- Nested functions/child function punya akses ke variabel yang di deklarasikan di luar scope (parent function).

b. Contoh Program

```
// global scope
function parentFunction() {
    //local scope untuk fungsi parentFunction
      ....
```

Nama : Fitra Ilyasa

NIM : 120140048

Kelas : Pengembangan Aplikasi Mobile RA

```
    //lexical scope untuk fungsi childFunction
    return childFunction() {
      //local scope untuk fungsi childFunction
    }
  }

// contoh program
function mobil() {
    var merk = "Honda";
    return function() {
        return merk + "Jazz";
    }
}
console.dir(mobil());
```

2. **Immediately Invoked Function Expression (Memanggil Fungsi sebagai Fungsi)**

   a. Definisi

   IIFE adalah bentuk function yang dieksekusi segera setelah function dideklarasikan. IIFE sering ditemui pada pemrograman JavaScript di browser terutama jika kita menggunakan plugin seperti jQuery.

   b. Contoh Program

```
(function () {
    console.log('Hello Fitra');
  })();
```

3. **First-class function**

   a. Definisi

   First-class object itu sendiri adalah sebuah entitas (bisa berbentuk function, atau variable) yang dapat dioperasikan dengan cara yang sama seperti entitas lain. Operasi tersebut biasanya mencakup passing entitas sebagai argument, return entitas dari sebuah function, dan assign entitas kedalam variable, object atau array. Apabila dikaitkan dengan javascript, intinya function pada javascript adalah first-class object, yang berarti function tersebut dapat:

Nama : Fitra Ilyasa

NIM : 120140048

Kelas : Pengembangan Aplikasi Mobile RA

- Di assign ke dalam variable, object, atau array
- Di passing sebagai argument pada function lain
- Di return dari sebuah function

b. Contoh Program

```
const getNama = function () {
    //contoh first class function
    return "Fitra";
  };

  console.log(getNama());

  function alamat() {
    return "Garut";
  }

  getAlamat = alamat(); // contoh first class function
  console.log(getAlamat);
```

4. Higher-order function

a. Definisi

Higher-order function atau fungsi orde tinggi adalah fungsi yang :

- menerima fungsi lain sebagai parameternya, dan/atau
- mengembalikan fungsi lain sebagai keluarannya.

b. Contoh Program

```
function nama(getNamaAkhir) {
    //functin nama adalah higher-order function;
    let namaAwal = "Fitra";
    return namaAwal + " " + getNamaAkhir;
  }

  function namaAkhir() {
    //funtion yang di panggil (*callback)
    return "Ilyasa";
  }
  console.log(nama(namaAkhir()));
```

Nama : Fitra Ilyasa
NIM : 120140048
Kelas : Pengembangan Aplikasi Mobile RA

5. Execution Context

a. Definisi

Sederhananya, **Execution Context** adalah konsep abstrak dari lingkungan di mana kode Javascript dievaluasi dan dieksekusi. Setiap kali kode apa pun dijalankan dalam JavaScript, kode itu dijalankan di dalam konteks eksekusi.

Jenis Konteks Eksekusi, Ada tiga jenis konteks eksekusi dalam JavaScript:

- **Execution Context Global** — Ini adalah konteks eksekusi default atau dasar. Kode yang tidak ada di dalam fungsi apa pun berada dalam konteks eksekusi global. Ia melakukan dua hal: ia menciptakan objek global yang merupakan objek jendela (dalam kasus browser) dan menetapkan nilai ini sama dengan objek global. Hanya ada satu konteks eksekusi global dalam suatu program.
- **Execution Context Fungsional** — Setiap kali suatu fungsi dipanggil, konteks eksekusi baru dibuat untuk fungsi itu. Setiap fungsi memiliki konteks eksekusinya sendiri, tetapi dibuat saat fungsi dipanggil atau dipanggil. Ada sejumlah konteks eksekusi fungsi. Setiap kali konteks eksekusi baru dibuat, ia melewati serangkaian langkah dalam urutan yang ditentukan yang akan saya bahas nanti di artikel ini.
- **Execution Context Fungsi Eval** — Kode yang dieksekusi di dalam fungsi eval juga mendapatkan konteks eksekusinya sendiri, tetapi karena eval biasanya tidak digunakan oleh pengembang JavaScript, jadi saya tidak akan membahasnya di sini.

6. Execution Stack

a. Definisi

Execution Stack juga dikenal sebagai "tumpukan panggilan" dalam bahasa pemrograman lain, adalah tumpukan dengan struktur LIFO (Last in, First out), yang digunakan untuk menyimpan semua konteks eksekusi yang dibuat selama eksekusi kode. Membuat konteks eksekusi global dan mendorongnya ke tumpukan eksekusi saat ini. Setiap kali mesin menemukan pemanggilan fungsi, itu membuat konteks eksekusi baru untuk fungsi itu dan mendorongnya ke atas tumpukan.

Nama : Fitra Ilyasa

NIM : 120140048

Kelas : Pengembangan Aplikasi Mobile RA

b. Contoh Program

```
let a = 'Halo Fitra!';
function first() {
  console.log('Di dalam fungsi pertama');
  second();
  console.log('Masih di dalam fungsi pertama');
}
function second() {
  console.log('Di dalam fungsi kedua');
}
first();
console.log('Di dalam Konteks Eksekusi Global');
```

7. Event Loop

a. Definisi

Merupakan tempat kode JavaScript dijalankan, berjalan dalam satu baris pada satu waktu dan tidak ada kemungkinan menjalankan kode secara paralel.

b. Contoh Program

```
// Event Loop
console.log("Sebelum delay");

function delayBySeconds(sec) {
    let start = now = Date.now()
    while(now-start < (sec*1000)) {
        now = Date.now();
    }
}

delayBySeconds(5);

// Dieksekusi setelah penundaan 5 detik
console.log("Setelah delay");
```

8. Callbacks

a. Definisi

Nama : Fitra Ilyasa

NIM : 120140048

Kelas : Pengembangan Aplikasi Mobile RA

Callbacks adalah fungsi yang diteruskan sebagai argumen ke fungsi lain. Menggunakan Callbacks, Anda dapat memanggil fungsi kalkulator (myCalculator) dengan Callbacks, dan membiarkan fungsi kalkulator menjalankan Callbacks setelah penghitungan selesai:

b. Contoh Program

```
function myDisplayer(some) {
    document.getElementById("demo").innerHTML = some;
  }

function myCalculator(num1, num2, myCallback) {
    let sum = num1 + num2;
    myCallback(sum);
  }

  myCalculator(5, 5, myDisplayer);
```

9. Promises dan Async/Await

a. Definisi

1) **Promises** merupakan Objek JavaScript Promise berisi kode produksi dan panggilan ke kode dapat berupa:
    - Tertunda
    - Terpenuhi
    - Ditolak

Objek Promise mendukung dua properti: **state** dan **result** .

- Sementara objek Promise "tertunda" (berfungsi), hasilnya tidak ditentukan.
- Ketika objek Janji "terpenuhi", hasilnya adalah nilai.
- Ketika objek Promise "ditolak", hasilnya adalah objek error.

2) **Async/Await** merupakan dua argumen (menyelesaikan dan menolak) telah ditentukan sebelumnya oleh JavaScript, tetapi memanggil salah satunya ketika fungsi eksekutor sudah siap, sangat sering kita tidak membutuhkan fungsi penolakan.

b. Contoh Program

Nama : Fitra Ilyasa

NIM : 120140048

Kelas : Pengembangan Aplikasi Mobile RA

```
// Promises
let myPromise = new Promise(function(myResolve, myReject) {
    // "Memproduksi Kode" (Mungkin perlu waktu)

      myResolve(); // ketika berhasil
      myReject();  // ketika error
    });

    // "Mengkonsumsi Kode" (Harus menunggu Janji terpenuhi)
    myPromise.then(
      function(value) { /* kode jika berhasil */ },
      function(error) { /* kode jika gagal */ }
    );

// Async/Await
async function myDisplay() {
    let myPromise = new Promise(function(resolve) {
      resolve("I love You !!");
    });
    document.getElementById("demo").innerHTML = await myPromise;
  }

  myDisplay();
```

Link Github : https://github.com/fitrailyasa/pam