Nama : Raja Saputera

NIM : 120140228

Kelas : Pengembangan Aplikasi Mobile RA

Link github : https://github.com/rajastra/Tugas-PAM

Tugas Individu 2

1.Closure

fungsi yang dieksekusi oleh fungsi lainnya(nested functions) sehingga fungsi tersebut memiliki akses ke variabel tempat fungsi tersebut lahir atau di buat.

contoh :

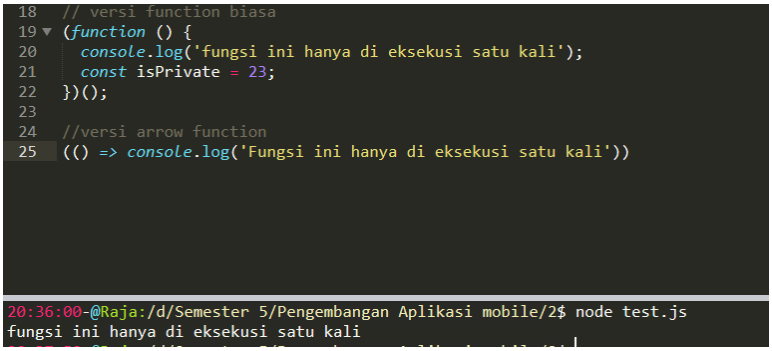
```
1  const tambahAngka = function () {
2    let count = 0;
3
4    return function () {
5      count++;
6      console.log(count);
7    };
8  };
9
10 const tambah = tambahAngka();
11
12 tambah();
13 tambah();
14 tambah();
```

```
20:26:51-@Raja:/d/Semester 5/Pengembangan Aplikasi mobile/2$ node test.js
1
2
3
```

Terlihat bawasannya fungsi tambah angka sebagai fungsi yang mengembalikan fungsi baru kemudian fungsi tersebut di letakkan ke dalam suatu variabel ketika variabel itu di panggil atau fungsi, fungsi yang di kembalikan bisa mengakses vairabel count dan juga bisa menambahnya hal ini bisa di lakukan karena adanya closure walaupun fungsi tambahAngka sudah selesai di eksekusi.

2.Immediately Invoked Function Expression

.Immediately Invoked Function Expression adalah fungsi yang akan menghilang apabila sudah di eksekusi satu kali. Contoh :



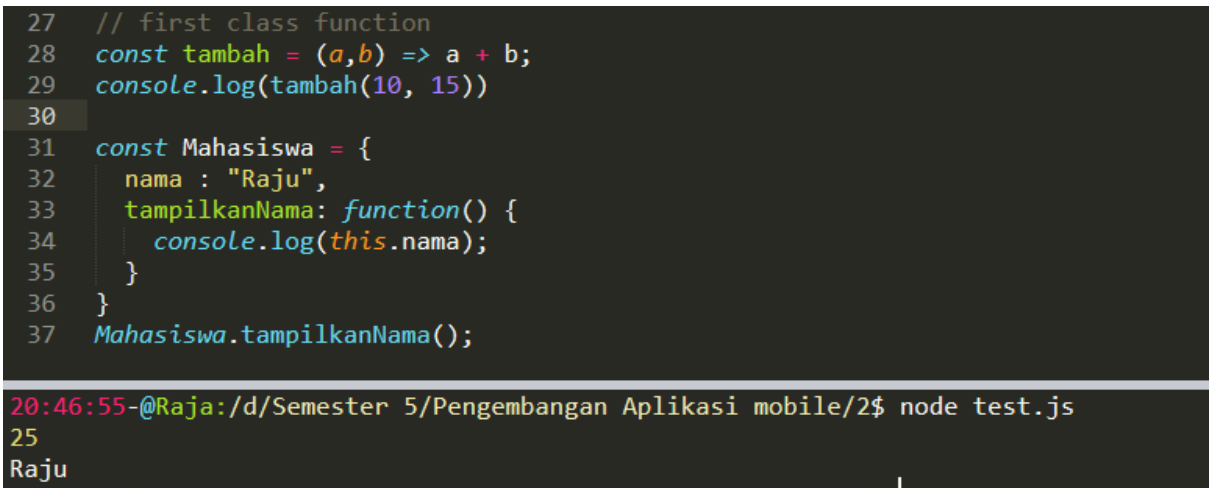
```
// versi function biasa
(function () {
  console.log('fungsi ini hanya di eksekusi satu kali');
  const isPrivate = 23;
})();

//versi arrow function
(() => console.log('Fungsi ini hanya di eksekusi satu kali'))
```

```
20:36:00-@Raja:/d/Semester 5/Pengembangan Aplikasi mobile/2$ node test.js
fungsi ini hanya di eksekusi satu kali
```

3.First-class function

First class function berararti bawassanya bahasa javascript memperlakukan fungsi hanya lah sebagai suatu nilai atau value dan fungsi juga hanya suatu tipe dari objek. Contoh :



```
// first class function
const tambah = (a,b) => a + b;
console.log(tambah(10, 15))

const Mahasiswa = {
  nama : "Raju",
  tampilkanNama: function() {
    console.log(this.nama);
  }
}
Mahasiswa.tampilkanNama();
```

```
20:46:55-@Raja:/d/Semester 5/Pengembangan Aplikasi mobile/2$ node test.js
25
Raju
```

Terlihat kita bisa menyimpan function atau fungsi kedalam suatu variabel dan propertu dari suatu objek.

4.Higher-order function

Higher oreder function adalah suatu fungsi yang menerima fungsi lain sebagai argument atau yang mengembalikan fungsi baru .

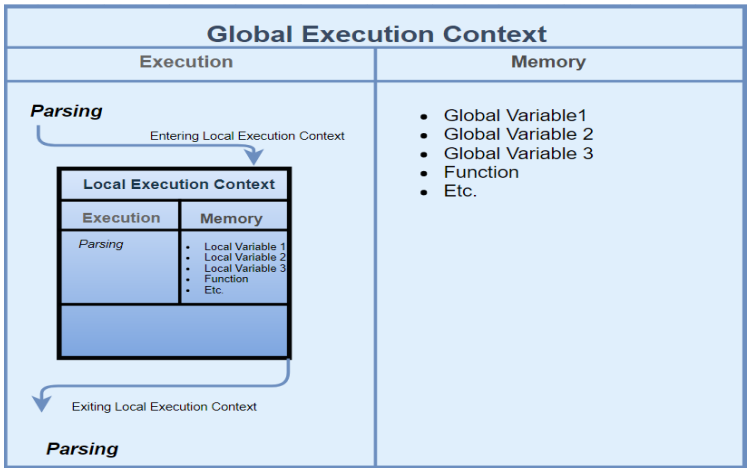
Contoh :

```
38  // higher oreder function
39  const sayHello = () => console.log('Hallo');
40  const btn = document.body.querySelector(".button"); // misalkan ada btn di markup
41  btn.addEventListener('click', sayHello);
```

Terlihat bawasannya yang menjadi higher oreder function yaitu addEventListener, yang menerima argument sayhello yang merupakan function juga.

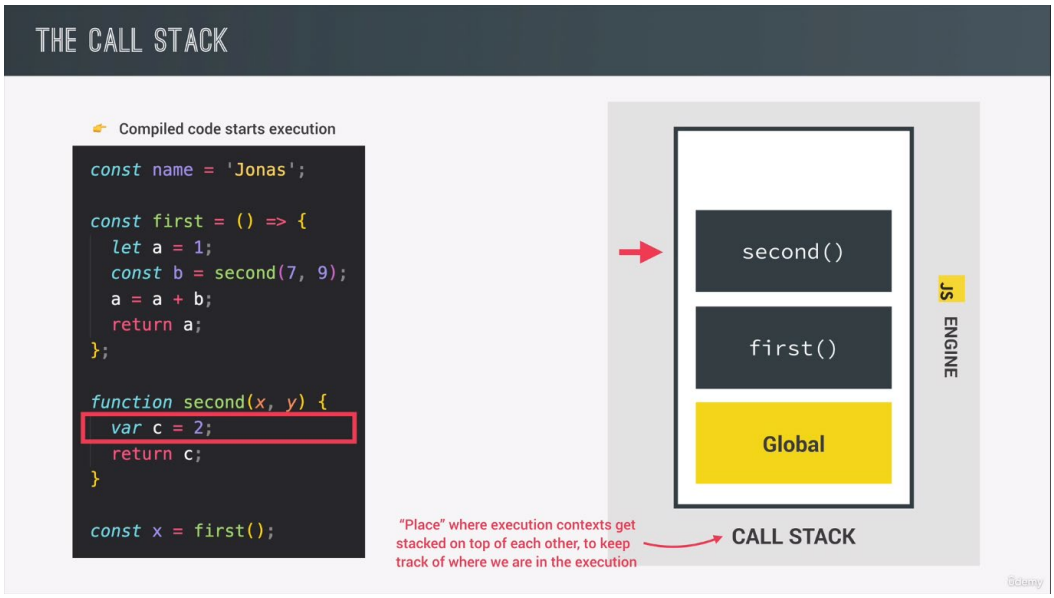
5.Execution Context

Execution Context adalah Environtment yang merupakan bagian dari javascript yang di eksekusi. Menyimpan semua informasi yang diperlukan untuk sutau kode menjadi di eksekusi. Seperti lokal variabel, argument yang di kirim untuk suatu fungsi



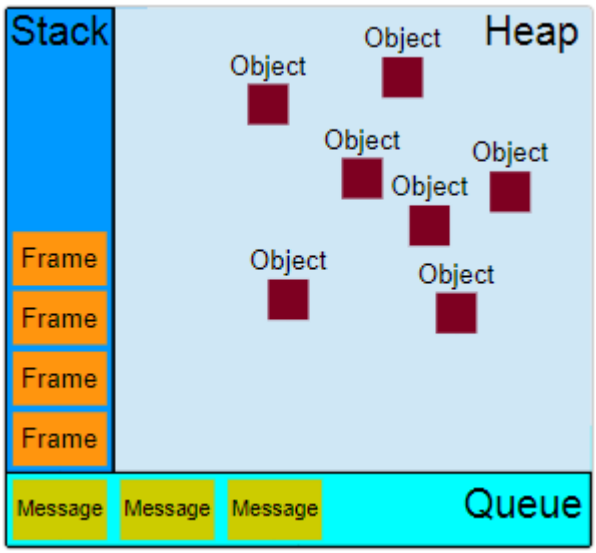
6.Execution Stack

Tempat di mana execution context ditumpuk, agar dapat melacak diamana sekarang di tempat eksekusinya.



7.Event Loop

JavaScript memiliki model runtime berdasarkan event loop, yang bertanggung jawab untuk mengeksekusi kode, mengumpulkan dan memproses peristiwa, dan mengeksekusi sub-tugas yang diantrekan. Model ini sangat berbeda dari model dalam bahasa lain seperti C dan Java.



8.Callbacks

Callback pada dasarnya adalah sebuah function. Callback hanya sebuah istilah untuk function yang di passing ke dalam function lain sebagai argument, yang kemudian di eksekusi oleh function yang membungkus function callback tersebut.

Contoh :

```
45  // callbacks
46  const ubahAwalHuruf = function(str){
47    const hasil = str.charAt(0).toUpperCase() + str.slice(1);
48    return hasil;
49  }
50
51  const satuKata = function(str){
52    return str.replace(/ /g, '').toLowerCase();
53  }
54
55  const ubahKata = function(str,fn){
56    console.log("Kata yang di ubah menjadi :", fn(str));
57  }
58  ubahKata("raja saputera", satuKata);
59  ubahKata("ahmad jamil", ubahAwalHuruf);
60
```

```
21:34:37-@Raja:/d/Semester 5/Pengembangan Aplikasi mobile/2$ node test.js
Kata yang di ubah menjadi : rajasaputera
Kata yang di ubah menjadi : Ahmad jamil
21:34:39-@Raja:/d/Semester 5/Pengembangan Aplikasi mobile/2$
```

9.Promises dan Async/Await

-Promise adalah promise adalah Sebuah mekanisme baru pada fitur javascript / ES6 yang merepresentasikan sebuah object request pengolahan data yang dilakukan secara asynchronous seperti ajax, dan promise ini mewakili sebuah operasi yang belum selesai, tetapi diharapkan di masa mendatang.

Contoh :

```
61  // promise
62  const wait = function (seconds) {
63    return new Promise(function (resolve) {
64      setTimeout(resolve, seconds * 1000);
65    });
66  };
67
68  wait(1)
69    .then(() => {
70      console.log('1 second passed');
71      return wait(1);
72    })
73    .then(() => {
74      console.log('2 second passed');
75      return wait(1);
76    })
```

Jadi disini membuat promise untuk async code yaitu menunggu waktu 1 detik kemudian di eksekusi dan dapat di hasilkan hasilnya menggunakan then.

-Async / await

Async/await adalah fitur yang hadir sejak ES2017. Fitur ini mempermudah kita dalam menangani proses asynchronous.Async/Await merupakan sebuah syntax khusus yang digunakan untuk menangani Promise agar penulisan code lebih efisien dan rapih

Contoh :

```
73  const wait = function (seconds) {
74    return new Promise(function (resolve) {
75      setTimeout(resolve, seconds * 1000);
76    });
77  };
78
79  const waiting = async function(){
80    await wait(1);
81    console.log("1 detik");
82    await wait(1);
83    console.log("2 detik");
84  }
85
86  waiting();
```

```
21:51:10-@Raja:/d/Semester 5/Pengembangan Aplikasi mobile/2$ node test.js
1 detik
2 detik
21:51:15-@Raja:/d/Semester 5/Pengembangan Aplikasi mobile/2$
```

Terlihat dari contoh bawasannya fungsi wait yang merupakan suatu promise kemudian dengan fungsi async kita dapat menggunakan fitur await di dalam fungsi tersebut sehingga dapat menjalan perintah asynchronous

Referensi :

https://www.udemy.com/course/the-complete-javascript-course/

https://www.freecodecamp.org/news/execution-context-how-javascript-works-behind-the-scenes/

https://medium.com/@aryarifqipratama/mengupas-tuntas-mengenai-apa-itu-callback-ce7bafc72173

https://codesyariah122.github.io/javascript-promise/#:~:text=Sooo%20jadi%20promise%20adalah%20Sebuah,tetapi%20diharapkan%20di%20masa%20mendatang.