NAMA : Muhammad Ibnu Prayogi

NIM : 120140152

Link Github : https://github.com/IbnuPrayogi/PAM.git

1. Closure
   Variabel dalam javascript dapat berlaku dalam global maupun local scope. Global variable merupakan variable yang dapat diakses pada keseluruhan kode javascript tanpa terkecuali. Local variable merupakan variable yang hanya dapat diakses pada fungsi tertentu tempat variable tersebut didefinsikan. Hal ini memunculkan konsep bernama closure. Closure adalah kondisi dimana terjadi kombinasi antara fungsi dimana terdapat outer dan inner fungsi dimana kondisi 'closure' terjadi apabila fungsi inner dapat mengakses variable yang di definisikan di fungsi outernya atau parentnya.

```
function parentFunction(){
    var parentVar = "Variabel fungsi Parent";
    function childFunction (){
        console.log ("Fungsi ini dapa mengakses " + parentVar);
    }
    childFunction();
}

parentFunction();
```

   Dari contoh diatas dapat dilihat bahwa outer function bernama parentFunction memiliki variable bernama "parentVar" yang mana variable tersebut dapat diakses oleh inner function bernama childfunction dan di cetak keluar melalui console.
2. Imidietly Invoke Function Execution
   immediately-Invoked Function Expressions (IIFE) merupakan suatu cara untuk mengeksekusi sebuah fungsi secara instant tepat setelah sebuah fungsi didefinisikan. Fungsi biasanya di definisikan dan dieksekusi secara terpisah seperti berikut:

```
//Definisi
function greeting (callback,name){
    console.log( "Hai "+name );
    callback();
}
function callbackFunction(){
    console.log ("Ini adalah contoh fungsi callback");
}

//Eksekusi
greeting(callbackFunction,"Ibnu");
```

   Akan tetapi, dengan menggunakan IIFE ,sebuah fungsi dapat dengan segera dieksekusi dengan menggunakan syntax umum sebagai berikut:

```
(function() {
    // Code that runs in your function
})()
```

   Berikut ini contoh penggunaanya:

```
(
    function ImmideteInvoke(greeting, name){
        console.log( greeting + name + " Ini merupakan contoh IIFE");
    }
)("Selamat Pagi ","Ibnu");
```

ImmideteInvoke merupakan sebuah fungsi yang membutuhkan argumen berupa sapaan dan nama yang mana argumen tersebut langsung ditambahkan pada akhir definisi fungsi sehingga proses eksekusi langsung dilakukan setelah proses definisi dilakukan. Hal ini dapat meringkas kode sehingga kode terlihat lebih bersih dan efesien.

3. First Class Function.
   Sebuah fungsi dapat dikatakan sebagai first class function apabila sebuah fungsi dapat diperlakukan seperti variable pada umumnya dimana fungsi tersebut dapat di assign sebagai nilai dari variable, dapat dipakai sebagai argumen, dan dapat di return dengan fungsi lain
   a) Dapat di assign sebagai variable

```
//Fungsi dapat assign pada variabel
const contohFungsi = ()=> {
    return (" Ini contoh fungsi sebagai assign variabel");

}
console.log(contohFungsi());
```

   Dapat dilihat bahwa fungsi tersebut di assign sebagai variable dengan tipe const

   b) Dapat digunakan sebagai argumen pada fungsi lainnya

```
//Fungsi dapat dipakai sebagai argumen di fungsi lain
const contohFungsi2=(func1)=>{
    return func1() + " dan sebagai argumen";
}

console.log(contohFungsi2(contohFungsi));
```

   c) Dapat di return oleh fungsi lainnya:

```
//Fungsi dapat di retur oleh fungsi lainnya
const contohFungsi3 = () =>{
    function cetak(){
        return ("Fungsi dapat di return oleh fungsi lainnya");
    }
    return cetak();
}
```

4. High-Order Function
   Hiigher order function merupakan dungsi yang bekerja pada fungsi lain baik digunakan sebagai argumen maupun sebagai nilai return. Dengan kata lain, higher order function adalah fungsi yag menerima fungsii sebagai argumen maupun mengembalikannya sebagai output.

```
const array = [2,5,6];
const array2 = array.map(function(num) {
  return num * 2;
});
console.log(array2);
```

Berikut ini merrupakan penggunaan fungsi map() dimana fungsi ini berfungsi membuat array baru dengan memanggi fungsi callback yang disediakan sebagai argument paa tiap elemen inpu array. Fungsi map() akan mengambil seluruh nilai return dari fungsi callback dan membuata array baru dari nilai tersebut.

5. Execution Context

   Konsep dari execution conteks digunakan untuk mendeskripsikan bagaimana sebuah kode dijalankan di bagian belakang. Execution context menentukan akses fungsi, variable, dan object pada setiap objek. Setiap kode akan dibagi menjadi line – line terpisah dimana setiap variable dan fungsi yang ada akan disimpan kedalam memori untuk kemudian dieksekusi. Execution context dibuat pada saat awal halaman HTML di muat dan setiap terdapat fungsi yang akan dijalankan. Terdapat dua tipe dari execution context.

   a) Global Execution Context

      Global execution context merupakan default execution context yang akan terbentuk saat pertama kali halaman dimuat. Pada tahap ini akan terjadi pembentukan objek window dan this dalam lingkungan Global execution. Pada tahap ini program juga akan mencari setiap variable dan fungsi yang terdapat pada code dan akan mengassign value nya dengan default value,yaitu “undefined”.

   b) Local/Funcition Execution context.

      Ketika sebuah fungsi dipanggil dan dieksekusi, sebuah execution context baru dibuat yang disebut dengan Fucntion Execution context. Secara default, Javascript membuat objek argumen dan sebuah objek this dalam LEC. Key dan value dari sebuah parameter juga diimuat dalam objek argumen dengan memiliki property berupa length.

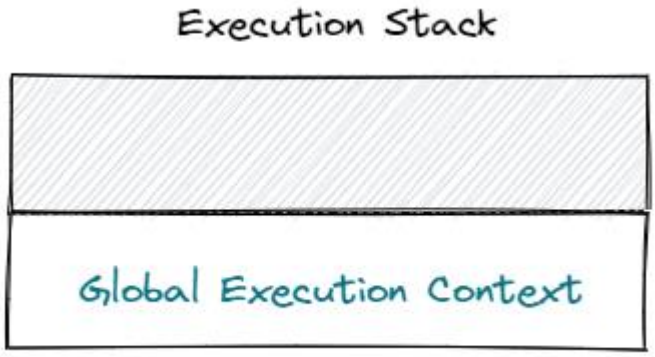
   c) Eval Function Execution Context

      Execition context yang satu ini perlu untuk dihindari sebisa mungkin. Execution context dibangun dan dimasukan ke sebuah call stack setiap kali fungsi eval() digunakan. Hal ini digunakan untuk mengecek string yang di masukan sebagai argumen. Maka dari itu, jika programmer secara tidak sengaja mengirimkan kode yang berbahayam hal ini juga membahayakan website.Oleh karena itu, fungsi ini tidak disarankan karena terdapat banyak alternative lain yang lebih baik.

6. Execution Stack

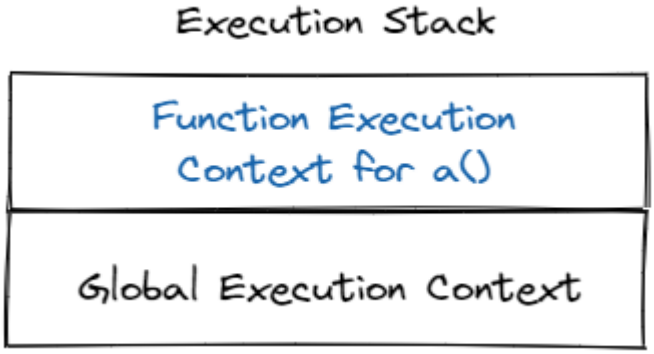
   Stack merupakan sebuah struktur data yang melakukan store data secara LIFO (Last in First Out) dimana data yang terakhir kali di masukan dalam stack akan menjadi data pertama yang dieksekusi. Dalam javascript, execution stack berfungsi untuk menjaga setiap execution context agar tetap berada pada jalur yang sama secara terurut sehingga dapat dieksekusi. Hal ini juga berkaitan dengan javascript yang bekerja secara single -threaded dimana hal ini berarti bahwa hanya ada satu operasi yang dapat dieksekusi dalam satu satuan waktu. Antrian execution context yang berisi operasi dan fungsi yang belum dieksekusi itulah yang disebut dengan execution stack.

Global execution context merupakan execution context default yang dibuat pada awal program sehingga posisinya berada pada bagian bottom dari stack.



Src : https://www.javatpoint.com

Kemudian, execution context yang terbetuk setelahnya akan diletakan diatas GOC yang sudah terbentuk.



Src : https://www.javatpoint.com

Sesuai dengan mekanisme nya, FEC akan dieksekusi terlebih dahulu untuk kemudian di pop out setelah selesai, kemudian program akan menuju pada execution context dibawah nya untuk segera dieksekusi

7. Event Loop
   Model runtime dari javascript berbasis pada event loop yang bertanggung jawab pada tiap fase eksekusi kode, mengumpulkan dan memproses operasi, dan mengeksekusi antriian tugas.

8. Callback function
   Callback adalah sebuah fungsi yang dijalankan pada fungsi lain dan bertindak sebagai argument dan akal dieksekusi setelah fungsi yang membungkusnya selesai dijalankan. Fungsi callback sering kali digunakan untuk melanjutkan eksekusi kode setelah operasi asyncronus selesai. Hal ini disebut dengan asyncronus callback.

```
function callbackFunction(){
    console.log ("Ini adalah contoh fungsi callback");
}

function greeting (callback,name){
    console.log( "Hai "+name );
    callback();
}
greeting(callbackFunction,"Ibnu");
```

Pada gambar diatas dapat dilihat bahwa fungsi greeting memanggil fungsi callback sebagai argument dan kemudian di eksekusi setelah seluruh kode pada fungsi greeting selesai dieksekusi.

9. Async-Await-Promise

Promise adalah sebuah mekanisme yang merepresentasikan sebuah permintaan objek dan pengolahan data yang dilakukan secara asynchronous. Promise tersebut mewakili sebuah operasi tugas yang belum selesai tetapi diharapakan akan diselesaikan pada masa mendatang. Ada tiga kemungkinan dalam promise, yaitu sedang dalam proses(pending), proses selesai(fulfilled), dan proses dibatalkan(rejected).
Ascyn await adalah sebuah syntax pada javascript yang berfungsi untuk mempermudah efesiensi promise. Async adalah fungsi yang mengembalika sebuah Promise. Sedangkan await adalah fungsi yang berjalan dalam Async. Await berfungsi untuk menunda proses Async hingga proses await selesai di eksekusi

```
function testPromise(){
    return new Promise(resolve=>{
        setTimeout(()=>{
            resolve('selesai')

        },3000)
    });

}
async function testAsync(){
    const test = await testPromise();
    console.log(test);
}

testAsync();
```

Dapat dilihat bahwa pada program diatas terdapat fungsi test async yang berjalan secara asyncronus dan kemudian dipanggil syntax await sehingga operasi pada fungsi test async harus ditunda selama 3 detik sampai proses pada fungsi testPromise selesai dijalankan